# Verification of component-based systems with recursive architectures 

Marius Bozga, Radu Iosif, Joseph Sifakis

## To cite this version:

Marius Bozga, Radu Iosif, Joseph Sifakis. Verification of component-based systems with recursive architectures. Theoretical Computer Science, 2023, 940, Part B, pp.146-175. 10.1016/j.tcs.2022.10.022 . hal-04106065

## HAL Id: hal-04106065

## https://hal.science/hal-04106065

Submitted on 25 May 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Verification of Component-based Systems with Recursive Architectures 

Marius Bozga*, Radu Iosif, Joseph Sifakis<br>Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, 38000 Grenoble, France


#### Abstract

We study a sound verification method for parametric component-based systems. The method uses a resource logic, a new formal specification language for distributed systems consisting of a finite yet unbounded number of components. The logic allows the description of architecture configurations coordinating instances of a finite number of types of components, by means of inductive definitions similar to the ones used to describe algebraic data types or recursive data structures. For parametric systems specified in this logic, we show that decision problems such as reaching deadlock or violating critical section are undecidable, in general. Despite this negative result, we provide for these decision problems practical semi-algorithms relying on the automatic synthesis of structural invariants allowing the proof of general safety properties. The invariants are defined using the $\mathrm{WS} \kappa S$ fragment of the monadic second order logic, known to be decidable by a classical automata-logic connection, thus reducing a verification problem to checking satisfiability of a $\mathrm{WS} \kappa S$ formula.


Keywords: Resource Logic, Component-based Distributed Systems, Parameterized Verification

## 1. Introduction

Mastering the complexity of a distributed system requires a deep understanding of its coordination mechanisms. We distinguish between endogenous coordination, that explicitly uses synchronization primitives in the code describ5 ing the behavior of the components (e.g. semaphores, monitors, compare-andswap, etc.) and exogenous coordination, that defines global rules describing how the components interact. These two orthogonal paradigms play different roles in the design of a system: exogenous coordination is used during high-level

[^0]model building, whereas endogenous coordination is considered at a later stage of development, to implement the model using low-level synchronization.

In this paper we focus on (high-level) exogenous coordination of distributed systems, consisting of a finite yet unbounded number of interconnected components. Communication is assumed to be correct, i.e. we abstract from packet losses and corruptions. Components behave according to a small set of finitestate abstractions of sequential programs, whose transitions are labeled with events. They communicate via interactions (handshaking) modeled as sets of events that occur simultaneously in multiple components. For instance, Figure 1 (a) shows a token-ring system, whose components are depicted by gray boxes containing state machines that represent their behavior and whose architecture is the set of connectors between components depicted by solid lines.

The separation between behavior and coordination is a fundamental principle in the design of large-scale distributed systems [1]. This modular view of a distributed system, in which the internal computation and the state changes of each component are encapsulated in a well defined interface, is key to scalable design methods that exploit a conceptual hierarchy. For instance, a ring is a chain whose final output port is connected to the initial input port, whereas a chain consists of a (head) component linked to a separate (tail) chain as shown in Figure 1(b). Moreover, system designers are accustomed to using architectural patterns, such as rings, pipelines, stars, trees, etc., that define interactions
${ }_{30}$ between (unboundedly large) sets of components. Such high-level models of reallife distributed systems are suitable for reasoning about correctness in the early stages of system design, when details related to network reliability or the implementation of coordination by means of low-level synchronization mechanisms (e.g. semaphores, monitors, compare-and-swap, etc.) are abstracted away.

### 1.1. Running Example

For starters, we consider the following recursive definition of a token-ring architecture, composed of a finite but unbounded number of stations that are instances of the same component type $S$, connected via transfer connectors, of interaction type T . The behavior of each station is a machine with two states, indicating whether the station has a token $(\mathfrak{t})$ or not $(\mathfrak{n})$. The number of tokens is constant and equal to the number of stations that are initially in state $\mathfrak{t}$. Each station not having a token may receive a token from the left, via the in port (that triggers the transition $\mathfrak{n} \xrightarrow{i n} \mathfrak{t}$ ) and send its token to the right, via the out port (that triggers the transition $\mathfrak{t} \xrightarrow{\text { out }} \mathfrak{n}$ ). We refer to Figure 1(a) for a depiction of a token-ring system, defined by the rules:

$$
\begin{align*}
\operatorname{Ring}() & \leftarrow \exists y_{1} \exists y_{2} \cdot \operatorname{Chain}\left(y_{1}, y_{2}\right) * \mathrm{~T}\left(y_{2}, y_{1}\right)  \tag{1}\\
\operatorname{Chain}\left(x_{1}, x_{2}\right) & \leftarrow \operatorname{Comp}\left(x_{1}\right) * \mathrm{~T}\left(x_{1}, x_{2}\right) * \operatorname{Comp}\left(x_{2}\right)  \tag{2}\\
\operatorname{Chain}\left(x_{1}, x_{2}\right) & \leftarrow \exists y_{1} \cdot \operatorname{Comp}\left(x_{1}\right) * \mathrm{~T}\left(x_{1}, y_{1}\right) * \operatorname{Chain}\left(y_{1}, x_{2}\right)  \tag{3}\\
\operatorname{Comp}\left(x_{1}\right) & \leftarrow \mathrm{S}^{\mathfrak{n}}\left(x_{1}\right)  \tag{4}\\
\operatorname{Comp}\left(x_{1}\right) & \leftarrow \mathrm{S}^{\mathfrak{t}}\left(x_{1}\right) \tag{5}
\end{align*}
$$



Figure 1: Recursive Specification of a Token-Ring System
Intuitively, rule 1 states that a token-ring consists of a chain of components and an interaction between the out port of the last and the in port of the first component in the chain. A chain consists of either two components and an interaction between their out and in ports, respectively (rule 2), or a component and an interaction between its out port and the in port of the first component of a distinct chain (3). The rules (4) and (5) intuitively say that a component has type $S$ and can be in state $\mathfrak{n}$ or in state $\mathfrak{t}$, respectively. The star symbol * used in the rules 115 is a commutative and associative logical connective that composes sub-systems with disjoint sets of components and interactions; for are necessarily different, meaning that $x_{1}$ and $x_{2}$ cannot be mapped to the same value. The rules 2 and 3 correspond to the base and the inductive case of a recursive definition of finite chains of length at least two. We refer to Figure 1 (b) for a depiction of the recursive unfolding of the above rules.

Note that no transfer of tokens is possible if the number of tokens in the system is either zero (there is no token to be transfered) or equals the number of components (there is no room to place a token). In this case, we say that a system is in a deadlock. The decision problem "can a token-ring started in a nondeadlock actually reach a deadlock?" is challenging, because it requires a proof
65 that holds for systems of any size, i.e. number of components and interactions. We show that, even if such problems are undecidable, in general, a large number of instances of these problems can be handled by the methods developed in this paper.

### 1.2. Contributions of this Work

The parameterized verification method presented in this paper relies on an idea reported in 2]. In addition to a complete development of the technical results, previously omitted for space reasons, here we apply the verification method to a specification language based on a resource logic, that resembles Separation Logic 3], instead of the recursive term algebra introduced in [2].
reasoning about the safety properties of programmed reconfiguration in dis tributed systems [4, to which the work presented here provides a push-button verification back-end. In combination with the proof system reported in 4], the verification method presented in this paper can automatically prove the correctness of a distributed system after the reconfiguration of its coordinating architecture. Since various dialects of (Concurrent) Separation Logic are being commonly used to specify and reason about concurrent systems [5, 6, 7], we expect this new logic to be easily accepted by the research and development community. The contribution of this paper is three-fold:

1. We introduce a logic-based language, called Configuration Logic (CL), to describe the sets of configurations (i.e. architectures and local states of components) of distributed systems parameterized by (i) the number of components of each type that are active in the system, e.g. a system with $n$ readers and $m$ writers, in which $n$ and $m$ are not known a priori and (ii) the pattern in which the interactions occur (e.g. a pipeline, ring, star or more general hypergraph structures). The language uses predicate symbols to hierarchically break the architecture into specific patterns. The interpretation of these predicate symbols is defined inductively by rewriting rules consisting of formulæ that contain predicate atoms, in a way that recalls the usual definitions of algebraic datatypes [8] or heaps [3].
2. We tackle a parametric safety problem concerning systems described in this language, which is essentially checking that the reachable states of every instance stays clear of a set of global error configurations, such as deadlocks or critical section violations. In particular, we show that both the parametric deadlock freedom and critical section violation problems are undecidable, even for architectures as simple as a chain of components.
3. We develop a verification method that synthesizes parametric invariants from the syntactic description of the architecture (in CL ) and from the behavior of its components (finite-state machines). The invariants and the set of error configurations are both described using a decidable fragment of Monadic Second Order Logic (MSO), that enables the use of off-the-shelf solvers for checking the resulting verification conditions.

The stages of the synthesis of verification conditions are depicted in Figure 2 The starting point is a set of inductive definitions (SID) and a CL formula describing the initial configurations of the system, together with a description

Finite-state Machines
Monadic Second Order Logic


Figure 2: The Synthesis of Verification Conditions
of component behavior by finite-state machines. The CL formula uses inductively defined predicate symbols defined by sets of rewriting trees, describing the unfoldings of their inductive definitions. In particular, these rewriting trees describe architectures, up to a permutation of the indices of components of each type.

We use rewriting trees as the backbones of architecture encoding, from which we derive a MSO flow formula, that essentially describes the operational semantics induced by the interactions of the architecture, relying on (i) a static description of the profile of the interactions (how they glue component types), and (ii) the transition systems describing the behavior of component types in the architecture. The main technical problem in building flow formulæ is tracking the identities and values of the variables that occur within predicate symbols, which, in turn, are recursively replaced by their definitions, in a rewriting tree. For instance, the $y_{2}$ variable introduced by rule (1) using existential quantification substitutes the $x_{2}$ variable from the definition of a Chain predicate, in rules (2) and (3) several times, before the same variable $\left(y_{2}\right)$ substitutes the $x_{1}$ variable in rule (4) or (5) (\$1.1). This is achieved by defining a path automaton that traverses the rewriting tree downwards tracking the introduction of a variable by existential quantification in a rule to the rule where it is instantiated in a component atom.

The flow formula is subsequently used to define invariants, i.e. over-approximations of the sets of states reachable from some initial configuration, in a system defined by some unfolding (rewriting) of the predicate symbols in CL specifications. In contrast with the classical approach of invariant inference using, e.g. abstract interpretation [9, our technique generates invariants as MSO formulæ obtained directly from the flow formulæ, by a syntactic translation. Analogously, the set of error configurations (deadlocks and critical section violations) is obtained directly from the flow formula of the system. The verification condition asks that the conjunction of the MSO formulæ describing the invariant and error configurations, respectively, is unsatisfiable; a satisfiable verification condition might indicate the presence of a spurious error caused by the overapproximation introduced by the invariant synthesis. Since we represent the configurations of a system by rewriting trees, we use a decidable fragment of MSO, interpreted over trees to answer the verification condition automatically.

The paper is structured as follows. Section 2 introduces the definitions of component-based systems and their operational semantics, Section 3 describes the resource logic used to define sets of configurations, Section 4 describes the parametric decision problems and gives the general undecidability results, Section 5 defines rewriting trees formally, together with a notion of a canonical model, induced by a rewriting tree of a logical formula, Section 6 describes the invariant synthesis and the encoding of the verification conditions in MSO, Section 7 reports on related work and Section 8 gives concluding remarks and sketches directions for future work.

## 2. Component-based Systems

### 2.1. Definitions

This section introduces the definitions needed to formally describe our model of component-based systems. We denote by $\mathbb{N}$ the set of natural numbers. Given integers $i$ and $j$, we write $[i, j]$ for the set $\{i, i+1, \ldots, j\}$, assumed to be empty if $i>j$. For a set $A$ and an integer $i \geq 1$, we denote by $A^{i}$ the $i$-times Cartesian product of $A$ with itself. $\operatorname{By} \operatorname{pow}(A)$ we denote the powerset of $A$. For a finite set $A$, we denote by $\|A\|$ its cardinality.

Let $\mathbb{P}$ be a countable set of ports. We consider classes of component-based systems that share the same signature $\Sigma=(\mathfrak{C}, \mathfrak{I}, \mathfrak{P})$, where:

- $\mathfrak{C}=\left\{\mathrm{C}_{1}, \ldots, \mathrm{C}_{N}\right\}$ is a finite set of relation symbols of arity one, called component types,
- $\mathfrak{I}=\left\{I_{1}, \ldots, \mathfrak{I}_{M}\right\}$ is a finite set of relation symbols of arity $\#\left(I_{j}\right) \geq 1$, called interaction types,
- $\mathfrak{P}: \mathfrak{C} \cup \mathfrak{I} \rightarrow \operatorname{pow}(\mathbb{P}) \cup \bigcup_{i \geq 1} \mathbb{P}^{i}$ is an interface associating each component type $C \in \mathfrak{C}$ a finite set of ports $\mathfrak{P}(\mathrm{C}) \in \operatorname{pow}(\mathbb{P})$ and each interaction type $I \in \mathfrak{I}$ a finite tuple of ports $\mathfrak{P}(I) \in \mathbb{P}^{\#(I)}$.
W.l.o.g., we assume that $\bigcup_{i=1}^{N} \mathfrak{P}\left(\mathrm{C}_{i}\right)=\mathbb{P}$ and $\mathfrak{P}\left(\mathrm{C}_{i}\right) \cap \mathfrak{P}\left(\mathrm{C}_{j}\right)=\emptyset$, for all $1 \leq$ $i<j \leq N$, i.e. each port $p$ belongs to the interface of exactly one component type, denoted by $\operatorname{comp}(p)$. We denote by $\mathfrak{I}^{(k)}$ the subset of interaction types of arity $k$, formally $\mathfrak{I}^{(k)} \stackrel{\text { def }}{=}\{\mathbf{I} \in \mathfrak{I} \mid \#(I)=k\}$, for any $k \geq 1$.

Example 1 (contd. from 81.1). The signature for the token-ring example from Figure 1 a is $\Sigma=\langle\{\mathrm{S}\},\{\mathrm{T}\}, \mathfrak{P}\rangle$, where $\mathfrak{P}(\mathrm{S})=\{$ in, out $\}$ and $\mathfrak{P}(\mathrm{T})=\langle$ out, in $\rangle$, i.e. the interaction type T connects an out port to an in port.

The component and interaction types are interpreted as sets and relations over a countably infinite universe $\mathbb{U}$ of indices. The particular nature of indices is not important at this point; we assume that two indices can only be compared for equality, with no other associated relation or function. An architecture $\alpha$ over the signature $\Sigma=(\mathfrak{C}, \mathfrak{I}, \mathfrak{P})$ associates each component type $\mathrm{C}_{i}$ a set $\alpha\left(\mathrm{C}_{i}\right) \subseteq \mathbb{U}$ and each interaction type $\mathbf{I}_{j}$ a relation $\alpha\left(\mathbf{I}_{j}\right) \subseteq \mathbb{U}^{\#\left(\mathbf{I}_{j}\right)}$, defining:

- components $\mathrm{C}_{i}[u]$, for some $u \in \alpha\left(\mathrm{C}_{i}\right)$, and
- interactions $\mathbf{I}_{j}\left[u_{1}, \ldots, u_{\#\left(\mathbf{I}_{j}\right)}\right]$, for some $\left\langle u_{1}, \ldots, u_{\#\left(\mathbf{I}_{j}\right)}\right\rangle \in \alpha\left(\mathbf{I}_{j}\right)$.

Intuitively, an architecture is a description of the static structure of a system. Note that an index $u \in \mathbb{U}$ can refer two (or more) different components $\mathrm{C}_{1}[u]$ and $\mathrm{C}_{2}[u]$ belonging to different component types and similarly, a tuple of indices $\mathbf{u}$ can refer two (or more) different interactions $\mathbf{I}_{1}[\mathbf{u}]$ and $\mathbf{I}_{2}[\mathbf{u}]$ of different $T) \cup(T \times S)$ is a set of edges. Given $x, y \in S \cup T$, we write $E(x, y) \stackrel{\text { def }}{=} 1$ if $(x, y) \in E$ and $E(x, y) \stackrel{\text { def }}{=} 0$, otherwise. Let $\cdot x \stackrel{\text { def }}{=}\{y \in S \cup T \mid E(y, x)=1\}$, $x \stackrel{\text { def }}{=}\{y \in S \cup T \mid E(x, y)=1\}$ and lift these definitions to sets of nodes. A marking of N is a function $\mathrm{m}: S \rightarrow \mathbb{N}$. A transition $t$ is enabled in m if and
225 only if $\mathrm{m}(s)>0$ for each place $s \in{ }^{\bullet} t$. We write $\mathrm{m} \xrightarrow{t} \mathrm{~m}^{\prime}$ whenever $t$ is enabled in m and $\mathrm{m}^{\prime}(s)=\mathrm{m}(s)-E(s, t)+E(t, s)$, for all $s \in S$ and $t \in T$. A sequence
of transitions $\sigma=t_{1}, \ldots, t_{n}$ is a firing sequence, written $\mathrm{m} \xrightarrow{\sigma} \mathrm{m}^{\prime}$ if and only if either (i) $n=0$ and $\mathrm{m}=\mathrm{m}^{\prime}$, or (ii) $n \geq 1$ and there exist markings $\mathrm{m}_{1}, \ldots, \mathrm{~m}_{n-1}$ such that $\mathrm{m} \xrightarrow{t_{1}} \mathrm{~m}_{1} \ldots \mathrm{~m}_{n-1} \xrightarrow{t_{n}} \mathrm{~m}^{\prime}$. A marking m is a deadlock of a Petri net $\mathrm{N}=(S, T, E)$ if and only if no transition $t \in T$ is enabled in m and let $\operatorname{Dead}(\mathbf{N})$ denote the set of deadlocks of N .

A marked Petri net is a pair $\mathcal{N}=\left(\mathrm{N}, \mathrm{m}_{0}\right)$, where $\mathrm{m}_{0}$ is the initial marking of N . A firing sequence is initial if it starts in $\mathrm{m}_{0}$. A marking m is reachable in $\mathcal{N}$ if there exists an initial firing sequence ending in m. Let $\operatorname{Reach}(\mathcal{N})$ (resp. Step $(\mathcal{N})$ ) be the set of markings reachable (in one step) in $\mathcal{N}$. For simplicity, we write $\operatorname{Reach}\left(\mathrm{N}, \mathrm{m}_{0}\right)$ and $\operatorname{Step}\left(\mathrm{N}, \mathrm{m}_{0}\right)$ instead of $\operatorname{Reach}\left(\left(\mathrm{N}, \mathrm{m}_{0}\right)\right)$ and $\operatorname{Step}\left(\left(\mathrm{N}, \mathrm{m}_{0}\right)\right)$, respectively. A marked Petri net $\mathcal{N}$ is boolean ${ }^{2}$ if $\mathrm{m}(s) \leq 1$, for each $s \in S$ and $\mathrm{m} \in \operatorname{Reach}(\mathcal{N})$. All marked Petri nets considered in the following will be boolean and we blur the distinction between a marking $\mathrm{m}: S \rightarrow\{0,1\}$
240 and the set $\{s \in S \mid \mathrm{m}(s)=1\}$, by writing $s \in \mathrm{~m}$ (resp. $s \notin \mathrm{~m}$ ) instead of $\mathrm{m}(s)=1$ (resp. $\mathrm{m}(s)=0)$.
Definition 1. The operational semantics of a system $\mathfrak{S}=(\Sigma, \alpha, \beta)$ with signature $\Sigma=(\mathfrak{C}, \mathfrak{I}, \mathfrak{P})$ is represented by the Petri net $\mathrm{N}(\mathfrak{S})=(S, T, E)$, where:

$$
\begin{aligned}
& S \stackrel{\text { def }}{=} \bigcup_{\mathbb{C} \in \mathfrak{C}}\left\{q[u] \mid q \in \operatorname{states}_{\beta}(\mathrm{C}), u \in \alpha(\mathrm{C})\right\} \\
& T \stackrel{\text { def }}{=} \bigcup_{\mathrm{I} \in \mathfrak{I}}\{ \left(\mathrm{I}\left[u_{1}, \ldots, u_{\#(\mathrm{I})}\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right) \mid\left\langle p_{1}, \ldots, p_{\#(\mathrm{I})}\right\rangle=\mathfrak{P}(\mathrm{I}), \\
&\left\langle u_{1}, \ldots, u_{\#(\mathrm{I})}\right\rangle \in \alpha(\mathrm{I}),\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle \in T_{\beta}(\mathrm{I}), \\
&\left.\forall i, j \in[1, \#(\mathrm{I})] . i \neq j \Rightarrow u_{i} \neq u_{j} \text { or } \operatorname{comp}\left(p_{i}\right) \neq \operatorname{comp}\left(p_{j}\right)\right\} \\
& E \stackrel{\text { def }}{=} \bigcup_{\mathrm{I} \in \mathfrak{I}}\{ \left(q\left[u_{i}\right],\left(\mathrm{I}\left[u_{1}, \ldots, u_{\#(\mathrm{I})}\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right)\right), \\
&\left(\left(\mathrm{I}\left[u_{1}, \ldots, u_{\#(\mathrm{I})}\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right), q^{\prime}\left[u_{i}\right]\right) \mid \\
&\left.t_{i}=\left(q \xrightarrow{p_{i}} q^{\prime}\right), i \in[1, \#(\mathrm{I})]\right\}
\end{aligned}
$$

The places, transitions and edges of $N(\mathfrak{S})$ are defined jointly by the architecture $\alpha$ and the behavior $\beta$. For every component $\mathrm{C}[u]$ in $\alpha$, the Petri net contains the places $q[u]$, for each state $q$ in states $_{\beta}(\mathrm{C})$. For every interaction $\mathrm{I}\left[u_{1}, \ldots, u_{\#(\mathrm{I})}\right]$ in $\alpha$, the Petri net contains one transition for every tuple $\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle$ of transitions of component types behavior, which are synchronizing according to $\boldsymbol{I}$, that is, where their labeling ports $\left\langle p_{1}, \ldots, p_{\#(\mathrm{I})}\right\rangle$ form the tuple $\mathfrak{P}(\mathrm{I})$. Moreover, a transition corresponding to an interaction $\mathrm{I}\left[u_{1}, \ldots, u_{\#(\mathrm{I})}\right]$ involves pairwise distinct components; the last condition in the definition of $T$ 250 above requires that $u_{i} \neq u_{j}$ or $\operatorname{comp}\left(p_{i}\right) \neq \operatorname{comp}\left(p_{j}\right)$. Finally, edges are defined according to the tuple of synchronizing transitions $\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle$ and connect to pre- and post- places, respectively $q\left[u_{i}\right]$ and $q^{\prime}\left[u_{i}\right]$, for each involved component $\mathrm{C}\left[u_{i}\right]$. For the sake of clarity we omit writing the tuple $\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle$ when it is determined by I, namely for those behaviors where a port labels exactly one 255 transition, in each state machine, as it is the case in the example below:

[^1]

Figure 3: The Execution Semantics of a Token-Ring System

Example 4 (contd. from Example 3). The Petri net describing the execution semantics of the token-ring system from Figure 1(a) is given in Figure 3. Consider the marking of this Petri net depicted in dashed lines. From this marking, the sequence of interactions $\mathrm{T}[1,2], \mathrm{T}[2,3], \ldots, \mathrm{T}[n-1, n], \mathrm{T}[n, 1]$ can be fired any number of times, in this order. These interactions correspond to the joint execution of the transitions labeled by the ports out and in, from any two adjacent components $i$ and $(i \bmod n)+1$ of the ring, respectively.

The executions of a system $\mathfrak{S}=(\Sigma, \alpha, \beta)$ can be represented by the firing sequences of a marked Petri net involving only those boolean markings of $\mathrm{N}(\mathfrak{S})$, $u \in \alpha(\mathrm{C})$. We formalize and prove this fact below:

Definition 2. Given a system $\mathfrak{S}=(\Sigma, \alpha, \beta)$ with signature $\Sigma=(\mathfrak{C}, \mathfrak{I}, \mathfrak{P})$, a marking m of the Petri net $\mathrm{N}(\mathfrak{S})=(S, T, E)$ is precise if and only if, for each $\mathrm{C} \in \mathfrak{C}$ and each $u \in \alpha(\mathrm{C})$, we have $\| \mathrm{m} \cap\left\{q[u] \mid q \in\right.$ states $\left._{\beta}(\mathrm{C})\right\} \|=1$.
${ }_{270}$ Proposition 1. Given a system $\mathfrak{S}$ and a precise initial marking $\mathrm{m}_{0}$ of $\mathrm{N}(\mathfrak{S})$, every marking $\mathrm{m} \in \operatorname{Reach}\left(\mathrm{N}(\mathfrak{S}), \mathrm{m}_{0}\right)$ is precise.

Proof. Let $\mathfrak{S}=(\Sigma, \alpha)$ and $\mathrm{m}^{\prime} \in \operatorname{Reach}\left(\mathrm{N}(\mathfrak{S}), \mathrm{m}_{0}\right)$ be a marking. The proof goes by the length $n \geq 0$ of the firing sequence taking $\mathrm{m}_{0}$ into $\mathrm{m}^{\prime}$. For the base case $n=0$, we have that $\mathrm{m}^{\prime}=\mathrm{m}_{0}$ is precise, by hypothesis. For the inductive step $n \geq 1$, let $m$ be the predecessor of $\mathrm{m}^{\prime}$ in the sequence, hence m is precise for $\mathfrak{S}$, by the inductive hypothesis. Then, by Def. 11, there exists a transition $\left(\mathrm{I}\left[u_{1}, \ldots, u_{\#(\mathrm{I})}\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right)$ in $T$, such that

$$
\begin{equation*}
\mathrm{m}^{\prime}=\left(\mathrm{m} \backslash\left\{q_{1}\left[u_{1}\right], \ldots, q_{\#(\mathrm{I})}\left[u_{\#(\mathrm{I})}\right]\right\}\right) \cup\left\{q_{1}^{\prime}\left[u_{1}\right], \ldots, q_{\#(\mathrm{I})}^{\prime}\left[u_{\#(\mathrm{I})}\right]\right\} \tag{6}
\end{equation*}
$$

where $\mathfrak{P}(\mathrm{I})=\left\langle p_{1}, \ldots, p_{\#(\mathrm{I})}\right\rangle$ and $t_{i}=\left(q_{i} \xrightarrow{p_{i}} q_{i}^{\prime}\right) \in \rightarrow_{\operatorname{comp}\left(p_{i}\right)}$ for all $i \in[1, \#(\mathrm{I})]$ and moreover $u_{i} \neq u_{j}$ or $\operatorname{comp}\left(p_{i}\right) \neq \operatorname{comp}\left(p_{j}\right)$ for all distinct $i, j \in[1, \#(\mathrm{I})]$.

Let $\mathrm{C} \in \mathfrak{C}$ be a component type and let $u \in \alpha(\mathrm{C})$ be an index. Since m is ${ }_{25}$ precise, we have $\mathrm{m} \cap\left\{q[u] \mid q \in\right.$ states $\left._{\beta}(\mathrm{C})\right\}=\left\{q^{*}[u]\right\}$, for some state $q^{*}$. We distinguish the following cases:

- $u=u_{i}$ for some $i \in[1, \#(\mathrm{I})]$ and $q^{*}=q_{i}$ In this case, we have:

$$
\mathrm{m}^{\prime} \cap\left\{q[u] \mid q \in \operatorname{states}_{\beta}(\mathrm{C})\right\}=\left\{q_{i}^{\prime}[u]\right\} \text {, by definition of } \mathrm{m}^{\prime} \text { in (6) }
$$

- $u=u_{i}$ for some $i \in[1, \#(\mathrm{I})]$ and $q^{*} \neq q_{i}$. In this case, as the transition $\left(\mathrm{I}\left[u_{1}, \ldots, u_{\#(\mathrm{I})}\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right)$ fires in m it follows that $q_{i}[u] \in \mathrm{m}$. But

In all non-contradictory cases, we obtain that $\| \mathrm{m} \cap\left\{q[u] \mid q \in\right.$ states $\left._{\beta}(\mathrm{C})\right\} \|=1$. Because $\mathbb{C} \in \mathfrak{C}$ and $u \in \alpha(\mathrm{C})$ are arbitrary, $m$ is precise for $\mathfrak{S}$.

## 3. The Configuration Logic

By configuration of a system, we understand the architecture describing the components and interactions from the system, together with a snapshot of its current state. Configurations are used to reason about parametric systems, that share a common architectural pattern (style) and differ in the number of instances of a certain component type. For instance, a token-ring (Figure 1) applies the same architectural pattern (the output of each component is connected to its right neighbor in a round-robin fashion) to any number of components, greater or equal to two.

We introduce the configuration logic (CL) to describe (possibly infinite) sets of configurations, via inductive definitions. Let $\Sigma=(\mathfrak{C}, \mathfrak{I}, \mathfrak{P})$ be a signature and $\beta: \mathfrak{C} \rightarrow \mathbb{B}$ be a behavior map, fixed for the rest of this section.
${ }_{295}$ Definition 3. A configuration of a system $\mathfrak{S}=(\Sigma, \alpha, \beta)$ is a pair $(\alpha, \mathrm{m})$, where $\alpha$ is an architecture over the signature $\Sigma$ and m is a precise marking of $\mathrm{N}(\mathfrak{S})$.

We aim at describing sets of configurations recursively (i.e. configurations with more complex structure being obtained by composing simpler ones), using the following definition of composition:
soo Definition 4. Two architectures $\alpha_{1}$ and $\alpha_{2}$, over the signature $\Sigma=(\mathfrak{C}, \mathfrak{I}, \mathfrak{P})$, are disjoint if and only if $\alpha_{1}(\mathrm{C}) \cap \alpha_{2}(\mathrm{C})=\emptyset$, for all $\mathrm{C} \in \mathfrak{C}$ and $\alpha_{1}(\mathrm{I}) \cap \alpha_{2}(\mathrm{I})=\emptyset$, for all $\boldsymbol{I} \in \mathfrak{I}$. If $\alpha_{1}$ and $\alpha_{2}$ are disjoint, the composition of configurations $\left(\alpha_{1}, \mathrm{~m}_{1}\right)$ and $\left(\alpha_{2}, \mathrm{~m}_{2}\right)$ is $\left(\alpha_{1}, \mathrm{~m}_{1}\right) \bullet\left(\alpha_{2}, \mathrm{~m}_{2}\right) \stackrel{\text { def }}{=}\left(\alpha_{1} \cup \alpha_{2}, \mathrm{~m}_{1} \cup \mathrm{~m}_{2}\right)$, where $\alpha_{1} \cup \alpha_{2}$ denotes the pointwise union of the architectures $\alpha_{1}$ and $\alpha_{2}$ :

- $\left(\alpha_{1} \cup \alpha_{2}\right)(\mathrm{C})=\alpha_{1}(\mathrm{C}) \cup \alpha_{2}(\mathrm{C})$, for each $\mathrm{C} \in \mathfrak{C}$,
- $\left(\alpha_{1} \cup \alpha_{2}\right)(\mathbb{I})=\alpha_{1}(\mathbb{I}) \cup \alpha_{2}(\mathbb{I})$, for each $\mathbf{I} \in \mathfrak{I}$.

The composition $\left(\alpha_{1}, \mathrm{~m}_{1}\right) \bullet\left(\alpha_{2}, \mathrm{~m}_{2}\right)$ is undefined if $\alpha_{1}$ and $\alpha_{2}$ are not disjoint.
It is easy to check that the composition of configurations $\left(\alpha_{1}, \mathrm{~m}_{1}\right)$ and $\left(\alpha_{2}, \mathrm{~m}_{2}\right)$ with disjoint architectures is again a configuration, in particular $\mathrm{m}_{1} \cap$ $\mathrm{m}_{2}=\emptyset$ and $\mathrm{m}_{1} \cup \mathrm{~m}_{2}$ is a precise marking of the Petri net $\mathrm{N}\left(\Sigma, \alpha_{1} \cup \alpha_{2}, \beta\right)$.

Let $\mathbb{X}^{1}$ be a set of first-order variables and $\mathbb{A}$ be a countably infinite set of predicate symbols, where $\#(\mathcal{A}) \geq 0$ denotes the arity of a predicate symbol $\mathcal{A} \in \mathbb{A}$. The formulæ of $C L$ are inductively defined by the following syntax:

$$
\phi::=\operatorname{emp}|\mathrm{C}(x)| \mathrm{C}^{q}(x)\left|\mathrm{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)\right| \mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right)|\phi * \phi| \exists x \cdot \phi
$$

where $C \in \mathfrak{C}$ are component types, $q \in$ states $_{\beta}(C)$ are states, $I \in \mathfrak{I}$ are interaction types, $\mathcal{A} \in \mathbb{A}$ are predicate symbols and $x_{1}, x_{2}, \ldots \in \mathbb{X}^{1}$ denote firstorder variables. Atomic formulæ of the form $\mathrm{C}(x)$ or $\mathrm{C}^{q}(x), \mathbf{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)$ and $\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right)$ are called component, interaction and predicate atoms, respec- tively. The logical connective $*$ is an associative and commutative separating conjunction operator.
$\operatorname{By} \operatorname{fv}(\phi)$ we denote the set of free variables that do not occur within the scope of a quantifier; $\phi$ is called quantifier-free (q.f.) if and only if it has no quantifiers and a sentence if and only if $\operatorname{fv}(\phi)=\emptyset$, respectively. A substitution is a partial mapping $\theta: \mathbb{X}^{1} \rightarrow \mathbb{X}^{1}$ and $\phi \theta$ is the formula obtained by replacing each variable $x \in \operatorname{fv}(\phi) \cap \operatorname{dom}(\theta)$ by $\theta(x)$ in $\phi$, where $\operatorname{dom}(\theta) \stackrel{\text { def }}{=}\left\{x \in \mathbb{X}^{1} \mid \theta\right.$ is defined at $\left.x\right\}$. We denote by $\left[x_{1} / y_{1}, \ldots, x_{k} / y_{k}\right]$ the substitution that replaces $x_{i}$ with $y_{i}$, for all $i \in[1, k]$ and is undefined everywhere else. This notation is extended to tuples of variables of equal length as $[\mathbf{x} / \mathbf{y}]$, where $\mathbf{x}=\left\langle x_{1}, \ldots, x_{k}\right\rangle$ and $\mathbf{y}=\left\langle y_{1}, \ldots, y_{k}\right\rangle$.

Intuitively, a formula emp describes configurations with empty architecture, $\mathrm{C}(x)$ (resp. $\left.\mathrm{C}^{q}(x)\right)$ describes configurations with architectures consisting of a single instance of the component type C , indexed by the value of $x$ (resp. in state $q$ ), and $\mathbf{I}\left(x_{1}, \ldots, x_{k}\right)$ describes a single interaction of type $\mathbf{I}$, between components indexed by $x_{1}, \ldots, x_{k}$, respectively. The formula $\mathrm{C}^{q_{1}}\left(x_{1}\right) * \ldots * \mathrm{C}^{q_{k}}\left(x_{k}\right) *$ $\mathrm{I}\left(x_{1}, \ldots, x_{k}\right)$ describes an architecture consisting of $k$ pairwise distinct instances of the component type C , in states $q_{1}, \ldots, q_{k}$, respectively, joined by an interaction of type $\mathbf{I}$. The formula $\mathbf{I}\left(x_{1}, \ldots, x_{k}\right) * \mathbf{I}\left(x_{1}^{\prime}, \ldots, x_{k}^{\prime}\right)$ states the existence of two interactions of type I, with distinct tuples of indices, given by the values of $\left\langle x_{1}, \ldots, x_{k}\right\rangle$ and $\left\langle x_{1}^{\prime}, \ldots, x_{k}^{\prime}\right\rangle$, respectively, i.e. the values of $x_{i}$ and $x_{i}^{\prime}$ must differ for at least one $i \in[1, k]$.

The semantics of CL formulæ is given by a satisfaction relation $\models_{\Delta}^{\mathfrak{s}}$ between configurations and formulæ. This relation is parameterized by:

- a store $\mathfrak{s}: \mathbb{X}^{1} \rightarrow \mathbb{U}$, i.e. a function mapping variables to indices, and
- a set of inductive definitions (SID) $\Delta$ consisting of rules of the form $\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \phi$, where $\phi$ is a CL formula such that $\operatorname{fv}(\phi)=\left\{x_{1}\right.$, $\left.\ldots, x_{\#(\mathcal{A})}\right\}$.
The satisfaction relation is defined inductively on the structure of the formulæ:

| $(\alpha, \mathrm{m}) \models_{\Delta}^{\mathfrak{s}} \mathrm{emp}$ | $\Longleftrightarrow \alpha(\mathbf{C})=\emptyset$, for all $\mathbb{C} \in \mathfrak{C}, \alpha(\mathbf{I})=\emptyset$, for all $\mathrm{I} \in \mathfrak{I}$ and $\mathrm{m}=\emptyset$ |
| :---: | :---: |
| $(\alpha, \mathrm{m}) \models^{\mathfrak{s}}{ }_{\Delta} \mathrm{C}(x)$ | $\Longleftrightarrow \alpha(\mathrm{C})=\{\mathfrak{s}(x)\}, \alpha\left(\mathrm{C}^{\prime}\right)=\emptyset$, for all $\mathrm{C}^{\prime} \in$ $\mathfrak{C} \backslash\{\mathrm{C}\}, \alpha(\mathrm{I})=\emptyset$, for all $\mathrm{I} \in \mathfrak{I}$ and $\mathrm{m}=$ $\{q[\mathfrak{s}(x)]\}$, for some $q \in$ states $_{\beta}(\mathrm{C})$ |
| $(\alpha, \mathrm{m}) \models{ }_{\Delta}^{\mathfrak{s}} \mathrm{C}^{q}(x)$ | $\Longleftrightarrow \alpha(\mathbf{C})=\{\mathfrak{s}(x)\}, \alpha\left(\mathbf{C}^{\prime}\right)=\emptyset$, for all $\mathrm{C}^{\prime} \in$ $\mathfrak{C} \backslash\{\mathrm{C}\}, \alpha(\mathrm{I})=\emptyset$, for all $\mathrm{I} \in \mathfrak{I}$ and $\mathrm{m}=$ $\{q[\mathfrak{s}(x)]\}$ |
| $(\alpha, \mathrm{m}) \models_{\Delta}^{\mathfrak{s}} \mathrm{I}\left(x_{1}, \ldots, x_{\#\left(\mathrm{I}_{k}\right)}\right)$ | $\begin{aligned} \Longleftrightarrow & \alpha(\mathrm{C})=\emptyset, \text { for all } \mathrm{C} \in \mathfrak{C}, \alpha(\mathrm{I})=\left\{\left\langle\mathfrak{s}\left(x_{1}\right), \ldots,\right.\right. \\ & \left.\left.\mathfrak{s}\left(x_{\#\left(I_{k}\right)}\right)\right\rangle\right\}, \alpha\left(\mathrm{I}^{\prime}\right)=\emptyset, \text { for all } \mathrm{I}^{\prime} \in \mathfrak{I} \backslash\{\mathrm{I}\} \text { and } \\ & \mathrm{m}=\emptyset \end{aligned}$ |

$$
\begin{aligned}
(\alpha, \mathrm{m}) \models_{\Delta}^{\mathfrak{s}} \mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A}))}\right) \Longleftrightarrow & (\alpha, \mathrm{m}) \models_{\Delta}^{\mathfrak{s}} \phi\left[y_{1} / x_{1}, \ldots, y_{\#(\mathcal{A})} / x_{\#(\mathcal{A})}\right], \text { for } \\
& \text { some rule } \mathcal{A}\left(y_{1}, \ldots, y_{\#(\mathcal{A})}\right) \leftarrow \phi \in \Delta \\
(\alpha, \mathrm{m}) \models_{\Delta}^{\mathfrak{s}} \phi_{1} * \phi_{2} \Longleftrightarrow & \text { there exist configurations }\left(\alpha_{1}, \mathrm{~m}_{1}\right),\left(\alpha_{2}, \mathrm{~m}_{2}\right) \\
& \text { such that }(\alpha, \mathrm{m})=\left(\alpha_{1}, \mathrm{~m}_{1}\right) \bullet\left(\alpha_{2}, \mathrm{~m}_{2}\right) \text { and } \\
& \left(\alpha_{i}, \mathrm{~m}_{i}\right) \models_{\Delta}^{\mathfrak{s}} \phi_{i}, \text { for } i=1,2 . \\
(\alpha, \mathrm{m}) \models_{\Delta}^{\mathfrak{s}} \exists x \cdot \phi_{1} \Longleftrightarrow & (\alpha, \mathrm{~m}) \models_{\Delta}^{\mathfrak{s}^{\prime}} \phi_{1}, \text { for some store } \mathfrak{s}^{\prime} \text { that agrees } \\
& \text { with } \mathfrak{s} \text { on all variables from } \mathbb{X}^{1} \backslash\{x\}
\end{aligned}
$$

To simplify the notation, we consider that $\Delta$ is fixed and omit the $\Delta$ subscript in the following. If the formula $\phi$ is a sentence, we can omit the store $\mathfrak{s}$ from the satisfaction relation $\models^{\mathfrak{s}}$ and write $(\alpha, \mathrm{m}) \models \phi$. In this case, $(\alpha, \mathrm{m})$ is said to be a model of $\phi$ and denote by $\llbracket \phi \rrbracket$ the set of models of the sentence $\phi$. For two sentences $\phi$ and $\psi$, we say that $\phi$ entails $\psi$ if $\llbracket \phi \rrbracket \subseteq \llbracket \psi \rrbracket$, written $\phi \models \psi$.

Example 5 (contd. from \$1.1). The SID consisting of the rules 1.5 (\$1.1) defines systems with token-ring architectures. On the other hand, the rules below define chains of S and T components, with at least $n, t \in \mathbb{N}$ components in state $\mathfrak{n}$ and $\mathfrak{t}$, respectively:

$$
\begin{array}{lc}
\text { Chain }_{0,0}(x, x) & \leftarrow \mathrm{S}(x) \\
\text { Chain }_{0,1}(x, x) & \leftarrow \mathrm{S}^{\mathfrak{t}}(x) \\
\text { Chain }_{1,0}(x, x) & \leftarrow \mathrm{S}^{\mathfrak{n}}(x) \\
\text { Chain }_{n, t}(x, z) & \leftarrow \exists y . \mathrm{S}^{\mathfrak{t}}(x) * \mathrm{~T}(x, y) * \text { Chain }_{n, t-1}(y, z) \\
\text { Chain }_{n, t}(x, z) & \leftarrow \exists y . \mathrm{S}^{\mathfrak{n}}(x) * \mathrm{~T}(x, y) * \text { Chain }_{n-1, t}(y, z)
\end{array}
$$

where $k \dot{-} 1 \stackrel{\text { def }}{=} \max (k-1,0), k \in \mathbb{N}$.
Below we show that CL sentences define indeed system configurations.
Proposition 2. Given a sentence $\phi$ of $C L$, if $(\alpha, \mathrm{m}) \in \llbracket \phi \rrbracket$ then m is a precise marking of the Petri net $\mathrm{N}(\Sigma, \alpha, \beta)$.

Proof. We prove a more general statement: for each formula $\phi$ and each store $\mathfrak{s}$, if $(\alpha, \mathrm{m}) \not \models^{\mathfrak{s}} \phi$ then m is a precise marking of $\mathrm{N}(\Sigma, \alpha, \beta)$. The proof is by induction on the definition of the $\models^{\mathfrak{s}}$ relation, by distinguishing the following cases:

- $\phi \in\left\{\mathrm{emp}, \mathrm{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)\right\}: \mathrm{m}=\emptyset$ and $\alpha(\mathrm{C})=\emptyset$, for all $\mathrm{C} \in \mathfrak{C}$, thus m is trivially precise for $\mathrm{N}(\Sigma, \alpha, \beta)$.
- $\phi=\mathrm{C}(x): \mathrm{m}=\{q[\mathfrak{s}(x)]\}$, for some $q \in$ states $_{\beta}(\mathrm{C}), \alpha(\mathrm{C})=\{\mathfrak{s}(x)\}$ and $\alpha\left(\mathrm{C}^{\prime}\right)=\emptyset$, for all $\mathrm{C}^{\prime} \in \mathfrak{C} \backslash\{\mathrm{C}\}$, hence m is precise for $\mathrm{N}(\Sigma, \alpha, \beta)$.
- $\phi=\mathrm{C}^{q}(x): \mathrm{m}=\{q[\mathfrak{s}(x)]\}, \alpha(\mathrm{C})=\{\mathfrak{s}(x)\}$ and $\alpha\left(\mathrm{C}^{\prime}\right)=\emptyset$, for all $\mathrm{C}^{\prime} \in$ $\mathfrak{C} \backslash\{\mathrm{C}\}$, hence m is precise for $\mathrm{N}(\Sigma, \alpha, \beta)$.
- $\phi=\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right):(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \psi\left[y_{1} / x_{1}, \ldots, y_{\#(\mathcal{A})} / x_{\#(\mathcal{A})}\right]$, for some rule $\mathcal{A}\left(y_{1}, \ldots, y_{\#(\mathcal{A})}\right) \leftarrow \psi \in \Delta$ and apply the inductive hypothesis.
- $\phi=\phi_{1} * \phi_{2}$ : if $(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \phi_{1} * \phi_{2}$ then there exists configurations $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s}}$ $\phi_{i}$, for $i=1,2$, such that $\alpha_{1}$ and $\alpha_{2}$ are disjoint, $\alpha=\alpha_{1} \cup \alpha_{2}$ and $\mathrm{m}=\mathrm{m}_{1} \cup \mathrm{~m}_{2}$. By the inductive hypothesis, $\mathrm{m}_{i}$ is a precise marking of $\mathrm{N}\left(\Sigma, \alpha_{i}, \beta\right)$, for $i=1,2$. Let $\mathrm{C} \in \mathfrak{C}$ be a component type and let $u \in \alpha(\mathrm{C})$ be an index. Then either $u \in \alpha_{1}(\mathrm{C})$ or $u \in \alpha_{2}(\mathrm{C})$, but not both. Let us consider the case $u \in \alpha_{1}(\mathrm{C}) \backslash \alpha_{2}(\mathrm{C})$, the other case being symmetric. Then $\mathrm{m} \cap\{q[u] \mid q \in \operatorname{states}(\mathrm{C})\}=\mathrm{m}_{1} \cap\{q[u] \mid q \in \operatorname{states}(\mathrm{C})\}$, hence $\|\mathrm{m} \cap\{q[u] \mid q \in \operatorname{states}(\mathrm{C})\}\|=\left\|\mathrm{m}_{1} \cap\{q[u] \mid q \in \operatorname{states}(\mathrm{C})\}\right\|=1$. Since the choices of C and $u$ were arbitrary, we obtain that m is precise.
- $\phi=\exists x . \phi_{1}$ : by an application of the inductive hypothesis.


## 4. Decision Problems

A decision problem is a class of yes/no queries of the same kind, that differ only in their input. All decision problems considered in this paper are parameterized by a given signature $\Sigma=(\mathfrak{C}, \mathfrak{I}, \mathfrak{P})$. The queries take as input a sentence $\phi$, a SID $\Delta$, a behavior map $\beta: \mathfrak{C} \rightarrow \mathbb{B}$ and a tuple of states $\left\langle q_{1}, \ldots, q_{n}\right\rangle \in \mathcal{Q}_{\beta}^{n}$ for some $n \in \mathbb{N}$. They are defined as follows:

- deadlock $(\phi, \Delta, \beta)$ : is there a configuration $(\alpha, \mathrm{m})$, such that $(\alpha, \mathrm{m}) \models_{\Delta} \phi$ and $\operatorname{Reach}(\mathrm{N}(\Sigma, \alpha, \beta), \mathrm{m}) \cap \operatorname{Dead}(\mathrm{N}(\Sigma, \alpha, \beta)) \neq \emptyset$ ?
- $\operatorname{reach}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right)$ : are there configurations $(\alpha, \mathrm{m}),\left(\alpha, \mathrm{m}^{\prime}\right)$ and indices $u_{1}, \ldots, u_{k} \in \mathbb{U}$, such that $(\alpha, \mathrm{m}) \models \Delta \phi, \mathrm{m}^{\prime} \in \operatorname{Reach}(\mathrm{N}(\Sigma, \alpha, \beta), \mathrm{m})$ and $\left\{q_{i}\left[u_{i}\right] \mid i \in[1, k]\right\} \subseteq \mathrm{m}^{\prime}$, with $q_{1}\left[u_{1}\right], \ldots, q_{k}\left[u_{k}\right]$ pairwise distinct?

The above queries occur typically as correctness conditions in system verification. For instance, proving that, in every system described by a formula $\phi$, no freeze configuration is reachable means showing that $\operatorname{deadlock}(\phi, \Delta, \beta)$ does not hold. Similarly, proving that each system described by $\phi$ stays clear of a set of error configurations e.g., at most one component is in some critical state $q_{\text {crit }}$ at any time amounts to proving that $\operatorname{reach}\left(\phi,\left\langle q_{\text {crit }}, q_{\text {crit }}\right\rangle, \Delta, \beta\right)$ does not hold.

### 4.1. Undecidability Results for Linear Systems

We show that the problems defined by the sets of queries above, taken over all inputs, but parameterized by a fixed given signature, are undecidable. In fact, we shall prove stronger results, in which the input formulæ define sets of configurations with linear architectures.

Definition 5. Given a signature $\Sigma=(\{\mathrm{C}\}, \mathfrak{T}, \mathfrak{P})$, a sentence $\phi$, interpreted over a SID $\Delta$, is linear if and only if $\phi=\Delta \cup \Lambda \Lambda \mathcal{L}_{\mathcal{L}}$, where $\Lambda$ consists of the rules:

$$
\begin{aligned}
& \mathcal{L}(x, x) \leftarrow \quad \mathrm{C}(x) * *_{\mathrm{I} \in \mathfrak{I}^{(1)}} \mathbf{I}(x) \\
& \mathcal{L}(x, y) \leftarrow \exists z \cdot \mathrm{C}(x) * *_{\mid \in \mathfrak{I}^{(1)} \mathbf{l}} \mathbf{l}(x) * *_{\mathbf{1} \in \mathfrak{I}^{(2)}} \mathbf{I}(x, z) * \mathcal{L}(z, y)
\end{aligned}
$$

where $\mathcal{L}$ is a binary predicate symbol and $\phi_{\mathcal{L}}$ is the sentence $\exists x \exists y . \mathcal{L}(x, y)$.

For example, the sentences $\exists x \exists y$.Chain $(x, y)(\$ 1.1)$ and $\exists x \exists y$. Chain $_{n, t}(x, y)$ for $n, t \geq 0$ (Example 5) are linear, when taking $\mathfrak{C}=\{\mathrm{S}\}$ and $\mathfrak{I}=\{\mathrm{T}\}$.

Below we show that undecidability occurs for classes of queries taking linear sentences as input, over a fixed signature consisting of only one component type and a fixed set of interaction types, which is not part of the input of a query.

Theorem 1. The following problems are undecidable:

$$
\begin{aligned}
\text { LinearDeadlock } & \stackrel{\text { def }}{=}\left\{\operatorname{deadlock}(\phi, \Delta, \beta) \mid \phi \models_{\Delta \cup \Lambda} \phi_{\mathcal{L}}\right\} \\
\text { LinearReachability } & \stackrel{\text { def }}{=}\left\{\operatorname{reach}\left(\phi,\left\langle q_{1}\right\rangle, \Delta, \beta\right) \mid \phi \models_{\Delta \cup \Lambda} \phi_{\mathcal{L}}\right\}
\end{aligned}
$$

The idea of the proof is to build a component-based system with linear architecture that simulates the execution of a Post-Turing machine. We present the construction in $\$ 4.2$ and complete the proof in $\$ 4.3$.

### 4.2. Simulation of Post-Turing Machines by Linear Systems

A Post-Turing machine [10, 11] executes sequential deterministic programs $M$ of the form $1:{s t m t_{1}}^{2} 2:{s t m t_{2}}$. ...; m: stmt $t_{m}$, where each statement $s t m t_{i}$ is one of following: write 0 , write 1 , go right, go left, goto step $j$ if read 0 , goto step $j$ if read 1 , stop, for some $j \in[1, m]$. The machine operates on an infinite tape of zeroes and ones. Initially, the head is pointing at some position on the tape and the program control is at the first statement. At any step, the current statement is executed and the tape content, the head position and control are updated according to that statement.

We simulate a Post-Turing machine by a component-based system with linear architecture. Its signature $\Sigma_{P T}=\left(\mathfrak{C}_{P T}, \mathfrak{I}_{P T}, \mathfrak{P}_{P T}\right)$ is presented in Figure 4 (a) and consists of one component type $C_{P T}$ and ten interaction types $\mathfrak{I}_{P T}$, with associated ports as presented in the figure. In particular, the signature does not depend on the program executed by the machine.

The linear system that simulates a Post-Turing machine is depicted in Figure 4 (b). First of all, the machine program $M$ is encoded in the behavior $\beta_{M}$ of the component type $\mathrm{C}_{P T}$. In particular, this behavior includes three disjoint state machines, as described in Table 1, labeled with the same set of ports. Second, in the linear system, each component plays a different role depending on its position, i.e. it runs according to one of the three state machines below:

- the leftmost ctrl component behaves according to the control state machine derived from the program $M$ : it issues the commands to be executed by the tape components depending on the current statement, and proceeds further according to the control flow of $M$. In case of read commands, it goes to an intermediate state, waiting to receive either read_0 or read_1 answers. In case of left! commands, it goes to an error state signaling the overflow of the tape on the left side. The complete definition is given in Table 1 (top).
- the middle tape components behave according to the tape state machine and mimic the behavior of a single tape cell: the $k$-th component state records the $k$-th symbol of the tape $\gamma_{k}$ and the presence $(T)$ or absence


Figure 4: Undecidability of Deadlock and Reachability for Linear Component-based Systems
$(\perp)$ of the machine head at that cell. They react to commands (received through in-ports) by changing their state and/or further issuing com- mands to their neighbors (sent through out-ports). The complete definition is given in Table 1 (middle).

- the rightmost sink component behaves according to the sink state machine and detects the overflow of the tape on the right side and signals it as an error. This happens when this component receives the right! command. The complete definition is given in Table 1 (bottom).


### 4.3. Proof of Theorem 1

Let $M$ be the program of a Post-Turing machine and let $w=\gamma_{1} \gamma_{2} \ldots \gamma_{n}$ be a finite word in $\{0,1\}^{n}$, assuming moreover wlog ${ }^{3}$ that $n \geq 2$. Consider the signature $\Sigma_{P T}$ and the behavior $\beta_{M}$ as defined in the previous section. Furthermore, consider the following set $\Delta_{w}$ of rules, where $\mathfrak{I}_{1} \stackrel{\text { def }}{=} \mathfrak{I}_{P T}^{(1)}, \mathfrak{I}_{2} \stackrel{\text { def }}{=} \mathfrak{I}_{P T}^{(2)}$ :

$$
\begin{array}{rlc}
\operatorname{Zeroes}(x, x) & \leftarrow & \mathrm{C}_{P T}^{(0, \perp)}(z) * *_{\mathrm{I} \in \mathfrak{I}_{1}} \mathrm{I}(x) \\
\operatorname{Zeroes}(x, y) & \leftarrow & \exists z \cdot \mathrm{C}_{P T}^{(0, \perp)}(x) * *_{\mathrm{I} \in \mathfrak{I}_{1}} \mathrm{I}(x) * *_{\mathrm{I} \in \mathfrak{I}_{2}} \mathrm{I}(x, z) * \operatorname{Zeroes}(z, y)
\end{array}
$$

[^2]control state machine

| $Q_{c t r l}^{M}{ }^{\text {def }}=\{i,(i,-) \mid i \in[1, m+1]\}$ represents the program $M$ |  |
| :---: | :---: |
| $i$ : write 0 | $i \xrightarrow{\text { out_write_ } 0}$ O $i+1$ |
| $i$ : write 1 | $i \xrightarrow{\text { out_write_-1 }} i+1$ |
| $i$ : go right | $i \xrightarrow{\text { out_right }} i+1$ |
| $i$ : go left | $i \xrightarrow{\text { out_left }} i+1, i \xrightarrow{\text { out_left }!}(i, \ldots),(i,)_{-}^{\text {out_err }}(i,-)$ |
| $i$ : goto step $j$ if read 0 | $i \xrightarrow{\text { out_read }}(i,-),\left(i, \_\right) \xrightarrow{i n \_ \text {read_0 }} j,\left(i,,_{-}\right) \xrightarrow{\text { in_read_1 }} i+1$ |
| $i$ : goto step $j$ if read 1 | $i \xrightarrow{\text { out_read }}(i, \ldots),(i,)_{-} \xrightarrow{\text { in_read_1 }}$, $j,(i,)_{-} \xrightarrow{\text { in_read_0 }} i+1$ |
| $i$ : stop | / * nothing */ |


| $Q_{\text {tape }} \xlongequal{\text { def }}\{(\gamma, \pi),(\gamma, \pi, a) \mid$ | $\left.\{0,1\}, \pi \in\{\perp, \top\}, a \in \mathfrak{I}_{P T} \backslash\{e r r\}\right\}$ |
| :---: | :---: |
| $(\gamma, \perp) \xrightarrow{\text { in-a }}(\gamma, \perp, a)$ | $(\gamma, \perp, a) \xrightarrow{\text { out_a }}(\gamma, \perp) \quad \forall a \neq$ right!, left!, err |
| $(\gamma, \perp) \xrightarrow{\text { in_right } t}(\gamma, \top)$ | $(\gamma, \perp$, left $) \xrightarrow{\text { out_left }!}(\gamma, \top)$ |
| $(\gamma, \top) \xrightarrow{\text { in_write_} \gamma^{\prime}}\left(\gamma^{\prime}, \top\right)$ |  |
| $(\gamma, \top) \xrightarrow{\text { in_read }}(\gamma, \top$, read $)$ | $(\gamma, \top$, read $) \xrightarrow{\text { out_read_ } \gamma}(\gamma, \top)$ |
| $(\gamma, \top) \xrightarrow{\text { in_right }}(\gamma, \top$, right $)$ | $(\gamma, \top$, right $) \xrightarrow{\text { out_right }!}(\gamma, \perp)$ |
| $(\gamma, \top) \xrightarrow{\text { in_left! }}(\gamma, \perp)$ |  |

$\xrightarrow{\text { sink state machine with states } Q_{\text {sink }} \stackrel{\text { def }}{=}\{i d l e, \text { busy }\}}$ idle $\xrightarrow{\text { in_right } t}$ busy $\quad$ busy $\xrightarrow{\text { out_err }}$ busy

Table 1: The behavior $\beta_{M}$ of $\mathrm{C}_{P T}$ for a Post-Turing machine executing a program $M$

$$
\begin{aligned}
& \text { Word_w }(x, y) \leftarrow \exists z_{2} \ldots \exists z_{n-1} \cdot \mathrm{C}_{P T}^{\left(\gamma_{1}, \mathrm{~T}\right)}(x) * *_{\mathrm{I} \in \mathfrak{I}_{1}} \mathrm{I}(x) * \\
& *_{\mathrm{I} \in \mathfrak{I}_{2}} \mathrm{I}\left(x, z_{2}\right) * \mathrm{C}_{P T}^{\left(\gamma_{2}, \perp\right)}\left(z_{2}\right) * *_{\mathrm{I} \in \mathfrak{I}_{1}} \mathrm{I}\left(z_{2}\right) * \\
& *_{\mathrm{I} \in \mathfrak{I}_{2}} \mathrm{I}\left(z_{2}, z_{3}\right) * \mathrm{C}_{P T}^{\left(\gamma_{3}, \perp\right)}\left(z_{3}\right) * *_{\mathrm{I} \in \mathfrak{I}_{1}} \mathrm{I}\left(z_{3}\right) * \\
& \text {... } \\
& *_{\mathrm{I} \in \mathfrak{I}_{2}} \mathrm{I}\left(z_{n-1}, y\right) * \mathrm{C}_{P T}^{\left(\gamma_{n}, \perp\right)}(y) * *_{\mathrm{I} \in \mathfrak{I}_{1}} \mathrm{I}(y) \\
& \operatorname{Tape} \_w(x, y) \leftarrow \exists z_{1} \exists w_{1} \exists z_{2} \exists w_{2} . \operatorname{Zeroes}\left(x, z_{1}\right) * \\
& *_{l \in \mathfrak{I}_{2}} \mathbf{I}\left(z_{1}, w_{1}\right) * \text { Word_w }\left(w_{1}, w_{2}\right) \\
& *_{\mathrm{I} \in \mathfrak{I}_{2}} \mathrm{I}\left(w_{2}, z_{2}\right) * \operatorname{Zeroes}\left(z_{2}, y\right) \\
& \text { Init_w }(x, y) \leftarrow \exists z_{1} \exists z_{2} \cdot \mathrm{C}_{P T}^{1}(x) * * \mathfrak{I}_{\mathfrak{I}_{1}} \mathrm{I}(x) \\
& *_{\mathrm{I} \in \mathfrak{I}_{2}} \mathrm{I}\left(x, z_{1}\right) * \text { Tape_w }\left(z_{1}, z_{2}\right) * \\
& * \underset{\mathbf{I} \in \mathfrak{I}_{2}}{ } \mathrm{I}\left(z_{2}, y\right) * \operatorname{C}_{P T}^{\text {idle }}(y) * * \mathbb{I}_{\in \mathfrak{I}_{1}} \mathrm{I}(y)
\end{aligned}
$$

and let $\phi_{w} \stackrel{\text { def }}{=} \exists x \exists y$. Init_w $(x, y)$. Intuitively, $\phi_{w}$ is a linear sentence which defines the valid (initial) configurations of the linear system encoding the PostTuring machine, where moreover $w$ is written on the tape and the machine head is pointing at the beginning $w$. We prove the following two assertions equivalent:

1. the Post-Turing machine running $M$ terminates on input $w$
2. the answer to deadlock $\left(\phi_{w}, \Delta_{w}, \beta_{M}\right)$ is yes
" $1 \Rightarrow 2$ ": If the machine terminates then it terminates by visiting finite portions of the tape to the left and to the right, with respect to the initial placement of $w$ on the tape. Henceforth, if the linear component-based system is started with the amount of tape cells needed on both sides, it will run without errors (that is, neither left or right tape overflow) until termination as well.
" $2 \Rightarrow 1$ ": First of all, if the linear system reaches a deadlock from its initial configuration, it means that neither left or right overflow occur (otherwise, the unary err interactions are enabled and run in an endless loop). Henceforth, the computation of the linear system involved only the allocated tape components and followed the execution of the statements in $M$. Moreover, by construction of the linear system, the relationship between the control and the tape state machines ensures that no blocking can occur between them: tape cells are con465 tinuously ready to receive the commands from the control and to execute them, in order. Therefore, the system stops in a deadlock only if no more commands are sent, i.e. when the control machine reaches a stop statement and all tape components are "idle", that is, no other commands are pending for completion.

The termination problem of Post-Turing machines being undecidable [11, 70 it implies that the LinearDeadlock problem is also undecidable. Moreover, we can use a similar argument to show that the LinearReachability problem is undecidable as well. Actually, we can restrict without loss of generality to programs $M$ containing a unique stop statement moreover occurring as the last statement $m$ in $M$. Then, for such programs we can prove the equivalence of
3. the answer to $\operatorname{reach}\left(\phi_{w},\langle m\rangle, \Delta_{w}, \beta_{M}\right)$ is yes
that is, the linear system reaches some configuration where a component is in state $m$. In fact, by construction, the only component that could reach the $m$ state is the leftmost control component, and therefore we can map back 480 any execution of the linear component-based system reaching $m$ in the control component, to a terminating execution of the Post-Turing machine.

## 5. Translating CL Specifications into Rewriting Trees

The inductive interpretation of predicate symbols in the CL logic, by means of a finite set of definitions, supports the idea of designing systems hierarchically (top-down). For instance, in $\$ 1.1$, we specify a ring system first by a chain of components, with an interaction between the out port of the last to the in port of the first component (rule 11). Then a chain consists of one component and an interaction between the out port of that component and the in port of the first component of a separate chain (rule 3), or of two components (rule
490 2. Intuitively, one can view these stages of the definition as rewriting steps, in which a predicate atom is replaced by one of the rules defining it. In this section, we formalize this idea by introducing rewriting trees, i.e. trees labeled with rules, that define a partial order in which the rules are applied.

Moreover, the verification method described in $\sqrt[6]{6}$ uses rewriting trees as

### 5.1. Rewriting Trees

Trees play an important role in the subsequent developments, hence we introduce a few formal definitions, for self-containment reasons. Let $\kappa \geq 1$ be an integer constant and let $[1, \kappa]^{*}$ denote the set of finite sequences of integers between 1 and $\kappa$, called nodes in the following. We denote the concatenation of nodes $w, u \in[1, \kappa]^{*}$ as $w \cdot u$ or simply $w u$, when no confusion arises. A set of nodes $T \subseteq[1, \kappa]^{*}$ is said to be:

1. prefix-closed if $w i \in T$, for some $i \in[1, \kappa]$, only if $w \in T$, and
2. complete if $w i \in T$, for some $i \in[1, \kappa]$, only if $w j \in T$, for all $j \in[1, i-1]$.
${ }_{520}$ A $\kappa$-ary tree $\mathcal{T}$ is a function mapping a complete prefix-closed set of nodes $\operatorname{nodes}(\mathcal{T})$ into a finite set of labels. The root of $\mathcal{T}$ is the empty sequence $\epsilon$, the children of a node $w \in \operatorname{nodes}(\mathcal{T})$ are $\operatorname{childs}_{\mathcal{T}}(w) \stackrel{\text { def }}{=}\{w i \in \operatorname{nodes}(\mathcal{T}) \mid i \in[1, \kappa]\}$ and the parent of a node $w i \in \operatorname{nodes}(\mathcal{T})$, where $i \in[1, \kappa]$, is $w$. The leaves of $\mathcal{T}$ are leaves $(\mathcal{T}) \stackrel{\text { def }}{=}\{w \in \operatorname{nodes}(\mathcal{T}) \mid w \cdot 1 \notin \operatorname{nodes}(\mathcal{T})\}$. The subtree of $\mathcal{T}$ rooted at ${ }_{525} \quad w$ is defined by $\operatorname{nodes}\left(\mathcal{T} \downarrow_{w}\right) \stackrel{\text { def }}{=}\left\{w^{\prime} \mid w w^{\prime} \in \operatorname{nodes}(\mathcal{T})\right\}$ and $\mathcal{T} \downarrow_{w}\left(w^{\prime}\right) \stackrel{\text { def }}{=} \mathcal{T}\left(w w^{\prime}\right)$, for all $w^{\prime} \in \operatorname{nodes}\left(\mathcal{T} \downarrow_{w}\right)$. A prefix of a tree $\mathcal{T}$ is the restriction of $\mathcal{T}$ to a complete prefix-closed subset of $\operatorname{nodes}(\mathcal{T})$. We say that $\mathcal{T}$ is finite if $\operatorname{nodes}(\mathcal{T})$ is finite.

Let $\Delta$ be a fixed SID in the rest of this section. For an arbitrary CL formula $\phi$, we denote by $\#_{\text {pred }}(\phi)$ the number of occurrences of predicate atoms and 530 by $\operatorname{pred}_{i}(\phi)$ the $i$-th predicate atom from $\phi$, in some predefined total ordering of the symbols in the syntax tree of $\phi$. A formula $\phi$ is said to be predicatefree if $\#$ pred $(\phi)=0$. Without loss of generality, we assume from now on that $\#$ pred $(\phi) \leq \kappa$, for each CL formula $\phi$ considered in the following. We shall also simplify our technical life by considering the following assumption:

Assumption 1. For each sentence $\phi$, there exists a rule $\mathcal{A}_{\phi}() \leftarrow \phi$ in $\Delta$, where $\mathcal{A}_{\phi}$ is a predicate symbol of zero arity, not occurring in $\phi$ or elsewhere in $\Delta$.

This assumption loses no generality because each query in a decision problem (\$4) considers finitely many sentences, for which finitely many rules of the above form are added to $\Delta$.

Definition 6. Given a CL formula $\phi$, a rewriting tree for $\phi$ is a tree $\mathcal{T}$ with labels from $\Delta$, such that $\mathcal{T}(\epsilon)=\left(\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \phi\right)$, for some predicate symbol $\mathcal{A}$ and, for each $w \in \operatorname{nodes}(\mathcal{T})$ such that $\mathcal{T}(w)=\left(\mathcal{A}_{0}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{0}\right)}\right) \leftarrow \psi_{0}\right)$, the following hold:

1. for all $i \in\left[1, \#\right.$ pred $\left.\left(\psi_{0}\right)\right]$, if $\operatorname{pred}_{i}\left(\psi_{0}\right)=\mathcal{A}_{i}\left(y_{1}^{i}, \ldots, y_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)$, then wi $\in$ $\operatorname{nodes}(\mathcal{T})$ and $\mathcal{T}(w i)=\left(\mathcal{A}_{i}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)}\right) \leftarrow \psi_{i}\right)$ is a rule from $\Delta$,
2. $w i \notin \operatorname{nodes}(\mathcal{T})$, for all $i>\#$ pred $\left(\psi_{i}\right)$.

We denote by $\mathbb{T}(\phi)$ the set of rewriting trees for $\phi$.
By slight abuse of notation, we write $\phi\left[\psi_{1} / \varphi_{1} \ldots \psi_{n} / \varphi_{n}\right]$ for the result of replacing each subformula $\psi_{i}$ of $\phi$ by the formula $\varphi_{i}$, for all $i \in[1, n]$. We associate each rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$ a characteristic formula $\mathfrak{F}(\mathcal{T})$, defined inductively on the structure of $\mathcal{T}$, as follows:

$$
\mathfrak{F}(\mathcal{T}) \stackrel{\text { def }}{=}\left\{\begin{array}{l}
\phi, \text { if } \operatorname{nodes}(\mathcal{T})=\{\epsilon\} \\
\phi\left[\mathcal{A}_{1}\left(\mathbf{y}_{1}\right) / \mathfrak{F}\left(\mathcal{T} \downarrow_{1}\right)\left[\mathbf{x}_{1} / \mathbf{y}_{1}\right] \ldots \mathcal{A}_{n}\left(\mathbf{y}_{n}\right) / \mathfrak{F}\left(\mathcal{T} \downarrow_{n}\right)\left[\mathbf{x}_{n} / \mathbf{y}_{n}\right]\right] \\
\text { if } n=\#_{\text {pred }}(\phi) \geq 1 \text { and } \operatorname{pred}_{i}(\phi)=\mathcal{A}_{i}\left(\mathbf{y}_{i}\right), \text { for all } i \in[1, n]
\end{array}\right.
$$

Intuitively, the characteristic formula of a rewriting tree is the predicate-free formula obtained by replacing each predicate atom occurring in a node of the tree by the characteristic formula of its corresponding subtree, recursively. In the process of building a characteristic formula, the free variables of the characteristic formulæ of the subtrees are substituted with the parameters of their corresponding predicate atoms.

Let $\mathfrak{M}(\mathcal{T}) \stackrel{\text { def }}{=} \eta$ denote the matrix of the characteristic formula $\mathfrak{F}(\mathcal{T})$ i.e., the largest quantifier-free formula of $\mathfrak{F}(\mathcal{T})=\exists x_{1} \ldots \exists x_{n} \cdot \eta$ written in prenex form. Since the separating conjunction and the existential quantifier distribute, every characteristic formula has a prenex form that is unique, modulo the commutativity and associativity of the separating conjunction.

Example 6. Let $\Sigma=(\{\mathrm{N}, \mathrm{L}\},\{\mathrm{R}, \mathrm{I}\}, \mathfrak{P})$ be a signature, where $\mathfrak{P}(\mathrm{N})=\{$ req, reply $\}, \mathfrak{P}(\mathrm{L})=\{$ reply, in, out $\}, \mathfrak{P}(\mathrm{R})=\langle$ req, reply, reply $\rangle$ and $\mathfrak{P}(\mathrm{I})=\langle$ out, in $\rangle$. Consider a behavior map $\beta$, such that states $_{\beta}(\mathrm{N})=\left\{q_{0}, q_{1}\right\}$ and states $_{\beta}(\mathrm{L})=$ $\left\{s_{0}, s_{1}, s_{2}\right\}$. The SID below defines tree-shaped architectures, whose leaves are

connected in a token ring:

$$
\begin{array}{r}
\operatorname{Root}() \leftarrow \exists r \exists n_{1} \exists \ell_{1} \exists r_{1} \exists n_{2} \exists \ell_{2} \exists r_{2} \cdot \mathrm{~N}^{q_{0}}(r) * \mathrm{R}\left(r, n_{1}, n_{2}\right) * \mathrm{I}\left(r_{1}, \ell_{2}\right) \\
* \mathrm{I}\left(r_{2}, \ell_{1}\right) * \operatorname{Node}\left(n_{1}, \ell_{1}, r_{1}\right) * \operatorname{Node}\left(n_{2}, \ell_{2}, r_{2}\right) \tag{7}
\end{array}
$$

$$
\operatorname{Node}(n, \ell, r) \leftarrow \exists n_{1} \exists r_{1} \exists n_{2} \exists \ell_{2} . \mathrm{N}^{q_{0}}(n) * \mathrm{R}\left(n, n_{1}, n_{2}\right) * \mathrm{I}\left(r_{1}, \ell_{2}\right)
$$

$$
\begin{equation*}
* \operatorname{Node}\left(n_{1}, \ell, r_{1}\right) * \operatorname{Node}\left(n_{2}, \ell_{2}, r\right) \tag{8}
\end{equation*}
$$

$\operatorname{Node}(n, \ell, r) \leftarrow \mathrm{N}^{q_{0}}(n) * \mathrm{R}(n, \ell, r) * \operatorname{Leaf}(\ell) * \mathrm{I}(\ell, r) * \operatorname{Leaf}(r)$

$$
\begin{equation*}
\operatorname{Leaf}(x) \leftarrow \mathrm{L}^{s_{0}}(x) \tag{9}
\end{equation*}
$$

Figure 5 (a) shows an unfolding tree for $\phi=\exists r \exists n_{1} \exists \ell_{1} \exists r_{1} \exists n_{2} \exists \ell_{2} \exists r_{2} . \mathrm{N}^{q_{0}}(r) *$ $\mathrm{R}\left(r, n_{1}, n_{2}\right) * \mathrm{I}\left(r_{1}, \ell_{2}\right) * \mathrm{I}\left(r_{2}, \ell_{1}\right) * \operatorname{Node}\left(n_{1}, \ell_{1}, r_{1}\right) * \operatorname{Node}\left(n_{2}, \ell_{2}, r_{2}\right)$, with $\mathcal{A}_{\phi}=$ Root, and Figure 5 (b) shows a model of the characteristic formula of this unfolding tree. To avoid name clashes, we annotate existentially quantified variables in the characteristic formula with the node of the unfolding tree where they
are introduced, for instance $r^{\epsilon}, n_{1}^{\epsilon}$ and $n_{2}^{\epsilon}$ are introduced at the root, whereas

Lemma 1. For each CL sentence $\phi$, we have $\llbracket \phi \rrbracket=\bigcup_{\mathcal{T} \in \mathbb{T}(\phi)} \llbracket \mathfrak{F}(\mathcal{T}) \rrbracket$.
Proof. We prove a slightly more general statement: for each formula $\phi$, each rule $\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \phi$ from $\Delta$, each configuration $(\alpha, \mathrm{m})$ and each store $\mathfrak{s}$ :

$$
\begin{aligned}
(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \phi \Longleftrightarrow & (\alpha, \mathrm{m}) \models^{\mathfrak{s}} \mathfrak{F}(\mathcal{T}), \text { for some } \mathcal{T} \in \mathbb{T}(\phi), \text { such } \\
& \text { that } \mathcal{T}(\epsilon)=\left(\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \phi\right)
\end{aligned}
$$

$" \Rightarrow$ " By induction on the definition of the satisfaction relation $\models$, considering the following cases:

- $\phi \in\left\{\operatorname{emp}, \mathrm{C}^{q}(x), \mathrm{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)\right\}$ : we define $\mathcal{T}$ by $\operatorname{nodes}(\mathcal{T}) \stackrel{\text { def }}{=}\{\epsilon\}$ and $\mathcal{T}(\epsilon) \stackrel{\text { def }}{=}\left(\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \phi\right)$. By Def. 6, we have $\mathcal{T} \in \mathbb{T}(\phi)$ and $\mathfrak{F}(\mathcal{T})=\phi$, thus $(\alpha, \mathrm{m}) \not{ }^{\mathfrak{s}} \mathfrak{F}(\mathcal{T})$.
- $\phi=\exists y_{1} \ldots \exists y_{n} . \phi_{0} * *{ }_{i=1}^{k} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)$, where $\phi_{0}$ is quantifier- and predicate-free: since $(\alpha, \mathrm{m}) \not \models^{\mathfrak{s}} \exists y_{1} \ldots \exists y_{n} \cdot \phi_{0} * *_{i=1}^{k} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)$, there exists a store $\mathfrak{s}^{\prime}$ that agrees with $\mathfrak{s}$ over $\mathbb{X}^{1} \backslash\left\{y_{1}, \ldots, y_{n}\right\}$ and configurations $\left(\alpha_{i}, \mathrm{~m}_{i}\right)$, such that $(\alpha, \mathrm{m})=\left(\alpha_{0}, \mathrm{~m}_{0}\right) \bullet \ldots \bullet\left(\alpha_{k}, \mathrm{~m}_{k}\right),\left(\alpha_{0}, \mathrm{~m}_{0}\right) \models^{\mathfrak{s}^{\prime}}$ $\phi_{0}$ and $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s}^{\prime}} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)$, for all $i \in[1, k]$. For each $i \in[1, k]$, since $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s}^{\prime}} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)$, there exists $\mathcal{A}_{i}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)}\right) \leftarrow \psi_{i}$ in $\Delta$, such that $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s}^{\prime}} \psi_{i}\left[x_{1} / z_{1}^{i}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)} / z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right]$, by the definition of $\models$. Let $\mathfrak{s}_{i}^{\prime} \stackrel{\text { def }}{=} \mathfrak{s}^{\prime} \circ\left[x_{1} / z_{1}^{i}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)} / z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right]$ be a store. We have $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \not \models^{\mathfrak{s}_{i}^{\prime}} \psi_{i}$ and, by the induction hypothesis, there exists a rewriting tree $\mathcal{T}_{i} \in \mathbb{T}\left(\psi_{i}\right)$, such that $\mathcal{T}_{i}(\epsilon)=\left(\mathcal{A}_{i}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)}\right) \leftarrow \psi_{i}\right)$ and $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s}_{i}^{\prime}} \mathfrak{F}\left(\mathcal{T}_{i}\right)$, hence $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models \models^{\mathfrak{s}^{\prime}} \mathfrak{F}\left(\mathcal{T}_{i}\right)\left[x_{1} / z_{1}^{i}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)} / z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right]$. We define the rewriting tree $\mathcal{T}$ as:

$$
-\operatorname{nodes}(\mathcal{T})=\{\epsilon\} \cup \bigcup_{i=1}^{k} i \cdot \operatorname{nodes}\left(\mathcal{T}_{i}\right)
$$

We obtain $\mathfrak{F}(\mathcal{T})=\exists y_{1} \ldots \exists y_{n} \cdot \phi_{0} * *{ }_{i=1}^{k} \mathfrak{F}\left(\mathcal{T}_{i}\right)\left[x_{1} / z_{1}^{i}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)} / z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right]$, leading to $(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \mathfrak{F}(\mathcal{T})$.
$" \Leftarrow "$ By induction on the structure of $\mathcal{T} \in \mathbb{T}(\phi)$, such that moreover $\mathcal{T}(\epsilon)=$ $n_{1}^{1}, n_{2}^{1}, \ell_{2}^{1}, r_{1}^{1}\left(n_{1}^{2}, n_{2}^{2}, \ell_{2}^{2}, r_{1}^{2}\right)$ occur first at the left (right) child of the root.

The following result shows that each model of a sentence is the model of the characteristic formula of some rewriting tree for that sentence, that represents the way in which the model is "produced" by unfolding the rules in $\Delta$. In general, such rewriting trees are not unique, as different rewriting trees can have equivalent characteristic formulæ.

$$
-\mathcal{T}(\epsilon)=\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \phi
$$

$$
-\mathcal{T}(i w)=\mathcal{T}_{i}(w), \text { for each } i \in[1, k] \text { and each } w \in \operatorname{nodes}\left(\mathcal{T}_{i}\right)
$$

- $\operatorname{nodes}(\mathcal{T})=\{\epsilon\}:$ since $\mathcal{T}(\epsilon)=\left(\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \phi\right)$, we have $\mathfrak{F}(\mathcal{T})=$ $\phi$, hence $(\alpha, \mathrm{m}) \not{ }^{\mathfrak{s}} \phi$.
- $\operatorname{childs}_{\mathcal{T}}(\epsilon)=\{1, \ldots, k\}$, for some $k \geq 1$ : assume w.l.o.g. that $\phi=$ $\exists y_{1} \ldots \exists y_{n} . \phi_{0} * *{ }_{i=1}^{k} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)$, where $\phi_{0}$ is a quantifier- and predicate-free formula. By the definition of characteristic formulæ, we have $\mathfrak{F}(\mathcal{T})=\exists y_{1} \ldots \exists y_{n} . \phi_{0} * *{ }_{i=1}^{k} \mathfrak{F}\left(\mathcal{T} \downarrow_{i}\right)\left[x_{1} / z_{1}^{i}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)} / z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right]$, where $\mathcal{T}(i)=\left(\mathcal{A}_{i}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)}\right) \leftarrow \psi_{i}\right)$, hence $\mathcal{T} \downarrow_{i} \in \mathbb{T}\left(\psi_{i}\right)$. Since $(\alpha, \mathrm{m})$ $\mid={ }^{\mathfrak{s}} \mathfrak{F}(\mathcal{T})$, there exists a store $\mathfrak{s}^{\prime}$ that agrees with $\mathfrak{s}$ over $\mathbb{X}^{1} \backslash\left\{y_{1}, \ldots, y_{n}\right\}$ and configurations $\left(\alpha_{0}, \mathrm{~m}_{0}\right), \ldots,\left(\alpha_{k}, \mathrm{~m}_{k}\right)$, such that $\left(\alpha_{0}, \mathrm{~m}_{0}\right) \not \models^{\mathfrak{s}^{\prime}} \phi_{0}$ and $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s}^{\prime}} \mathfrak{F}\left(\mathcal{T} \downarrow_{i}\right)\left[x_{1} / z_{1}^{i}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)} / z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right]$, for all $i \in[1, m]$. Consider an arbitrary index $i \in[1, m]$ and let $\mathfrak{s}_{i}^{\prime} \stackrel{\text { def }}{=} \mathfrak{s}^{\prime} \circ\left[x_{1} / z_{1}^{i}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)} / z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right]$, hence $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s}_{i}^{\prime}} \mathfrak{F}\left(\mathcal{T} \downarrow_{i}\right)$. By the inductive hypothesis, we have $\left(\alpha_{i}, \mathrm{~m}_{i}\right)$ $\mid=^{\mathfrak{s}_{i}^{\prime}} \psi_{i}$, hence $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s}_{i}^{\prime}} \mathcal{A}_{i}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)}\right)$ and $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s}^{\prime}} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots\right.$, $\left.z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)$ follows. Because the above holds for all $i \in[1, k]$, we conclude that $(\alpha, \mathrm{m}) \neq^{\mathfrak{s}} \phi$.


### 5.2. Canonical Models

In the following, we shall equate the universe of indices $\mathbb{U}$ with the set $[1, \kappa]^{*}$ of tree nodes, i.e. finite sequences of integers between 1 and $\kappa$. This is without loss of generality, because both sets are infinitely countable and the nature of indices plays no role in the interpretation of CL formulæ. In this setting, we define the canonical model $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)$ of the characteristic formula $\mathfrak{F}(\mathcal{T})$ of a rewriting tree $\mathcal{T}$. In order to simplify its definition, we proceed under the following assumption:

Assumption 2. Each rule of $\Delta$ is of one of the forms:

$$
\begin{align*}
\mathcal{A}\left(x_{1}\right) & \leftarrow \phi, \text { such that } \phi \neq \mathrm{emp}  \tag{I}\\
\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) & \leftarrow \exists y_{1} \ldots \exists y_{n} \cdot \phi * \psi * *{ }_{i=1}^{p} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right) \tag{II}
\end{align*}
$$

for some $n \geq 0$ and $p \geq 1$, where:

- $\phi \in\left\{\mathrm{C}^{q}\left(x_{1}\right) \mid \mathrm{C} \in \mathfrak{C}, q \in\right.$ states $\left._{\beta}(\mathrm{C})\right\} \cup\{\mathrm{emp}\}$,
- $\psi$ is a separating conjunction of interaction atoms,
- $\bigcup_{i=1}^{p}\left\{z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right\}=\left(\left\{x_{1}, \ldots, x_{\#(\mathcal{A})}\right\} \backslash \operatorname{fv}(\phi)\right) \cup\left\{y_{1}, \ldots, y_{n}\right\}$, and
- $\left\{z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right\} \cap\left\{z_{1}^{j}, \ldots, z_{\#\left(\mathcal{A}_{j}\right)}^{j}\right\}=\emptyset$, for all $1 \leq i<j \leq p$.

The above assumption is not without loss of generality. On the positive side, all examples considered in this paper can be written using only rules of this form, for instance, the definition of the Chain $_{n, t}$ predicates, for constant integers $n, t \geq 0$ (Example 5) or the definition of the Root, Node and Leaf predicates (Example 6).

A direct consequence of the above assumption is that, each variable that occurs in a characteristic formula of a rewriting tree, occurs in exactly one component atom, in a rule that labels exactly one node from the rewriting tree. This fact is proved below:

Lemma 2. Let $\phi$ be a $C L$ formula and $\mathcal{T} \in \mathbb{T}(\phi)$ be a rewriting tree for $\phi$. For each variable $x$, that occurs free or existentially quantified in $\mathfrak{F}(\mathcal{T})$, there exists exactly one node $w \in \operatorname{nodes}(\mathcal{T})$ and one component atom $\mathrm{C}^{q}(x)$ in $\mathcal{T}(w)$.

Proof. It is sufficient to prove the statement for the case where $x$ occurs in the label of the root of $\mathcal{T}$. If it is introduced at some node $w \in \operatorname{nodes}(\mathcal{T}) \backslash\{\epsilon\}$ by an existential quantifier, the same proof can be done taking $\mathcal{T} \downarrow_{w}$ instead of $\mathcal{T}$. The proof goes by induction on the structure of $\mathcal{T}$. For the base case $\operatorname{nodes}(\mathcal{T})=\{\epsilon\}$, the only possibility is that $\mathcal{T}(\epsilon)$ is a rule of the form (II) and the proof is immediate. For the inductive step, let $\mathcal{T}(\epsilon)$ be the following rule:

$$
\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \exists y_{1} \ldots \exists y_{n} \cdot \phi * \psi * *_{i=1}^{p} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)
$$

If $\phi$ is of the form $\mathrm{C}^{q}\left(x_{1}\right)$ and $x=x_{1}$, then $\epsilon$ is the unique node such that $x$ occurs in a component atom. Else, $x \in\left\{x_{1}, \ldots, x_{\#(\mathcal{A})}\right\} \cup\left\{y_{1}, \ldots, y_{n}\right\} \backslash \mathrm{fv}(\phi)$, thus $x \in\left\{z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right\}$, for exactly one $i \in[1, p]$, by Assumption 2 . In this case the inductive hypothesis applies, thus $x$ occurs in exactly one component ${ }_{640}$ atom $\mathrm{C}^{q}(x)$ in $\mathcal{T}(w)$, for exactly one $w \in \operatorname{nodes}\left(\mathcal{T} \downarrow_{i}\right)$.

The canonical model of a characteristic formula $\mathfrak{F}(\mathcal{T})$ is obtained by instantiating each variable $x$ that occurs free or existentially quantified in $\mathfrak{F}(\mathcal{T})$ by the unique node $w \in \operatorname{nodes}(\mathcal{T})$, such that $x$ occurs in a single component atom $\mathrm{C}^{q}(x)$ in $\mathcal{T}(w)$ (Lemma 22. Formally, let $\phi=\exists y_{1} \ldots \exists y_{n} . \psi * *{ }_{i=1}^{p} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)$ be a formula, where $\psi$ is quantifier- and predicate-free, and define, for each rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, a store $\mathfrak{s}_{\mathcal{T}}^{\epsilon}: \operatorname{fv}(\psi) \cup \bigcup_{i=1}^{p}\left\{z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right\} \rightarrow \operatorname{nodes}(\mathcal{T})$, as follows:

$$
\mathfrak{s}_{\mathcal{T}}^{\epsilon}(x) \stackrel{\text { def }}{=} \begin{cases}\epsilon & \text { if } x \text { occurs in a component atom from } \psi \\ i \cdot \mathfrak{s}_{\mathcal{T}_{i}}^{\epsilon}\left(x_{j}\right) & \text { if } x=z_{j}^{i}, \text { for some } j \in\left[1, \#\left(\mathcal{A}_{i}\right)\right], \text { where } \\ & \mathcal{T}(i)=\left(\mathcal{A}_{i}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)}\right) \leftarrow \varphi_{i}\right), \text { for all } i \in[1, p]\end{cases}
$$

Note that, because the rules of the SID meet the conditions of Assumption 2, the two cases of the definition of $\mathfrak{s}_{\mathcal{T}}^{\epsilon}$ above are exclusive and, moreover, $\mathfrak{s}_{\mathcal{T}}^{\epsilon}(x)$ is defined for all free variables of the matrix $\psi * *{ }_{i=1}^{p} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)$ of $\phi$.

Recall that $\mathfrak{M}(\mathcal{T})$ denotes the quantifier-free matrix of the characteristic formula $\mathfrak{F}(\mathcal{T})$. We extend the store $\mathfrak{s}_{\mathcal{T}}^{\epsilon}$ to a store $\mathfrak{s}_{\mathcal{T}}: \operatorname{fv}(\mathfrak{M}(\mathcal{T})) \rightarrow \operatorname{nodes}(\mathcal{T})$ mapping all variables (free or existentially quantified) that occur in the characteristic formula $\mathfrak{F}(\mathcal{T})$ :

$$
\mathfrak{s}_{\mathcal{T}}(x) \stackrel{\operatorname{def}}{=} \begin{cases}\mathfrak{s}_{\mathcal{T}}^{\epsilon}(x) & \text { if } x \in \mathrm{fv}(\mathcal{F}(\mathcal{T}))  \tag{11}\\ w \cdot \mathfrak{s}_{\mathcal{T}_{w}}^{\epsilon}(x) & \text { if } x \text { is existentially quantified at } w \in \operatorname{nodes}(\mathcal{T})\end{cases}
$$

In the above definition, we have assumed that all existentially quantified variables have pairwise distinct names (this is w.l.o.g. as quantified variables can be $\alpha$-renamed, if necessary). Note that $\mathfrak{s}_{\mathcal{T}}$ is uniquely defined, for each given rewriting tree $\mathcal{T}$ and induces a unique canonical model, defined below:
Definition 7. The canonical model of a rewriting tree $\mathcal{T}$ is the configuration $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)$, defined as follows:

- $\alpha_{\mathcal{T}}(\mathrm{C}) \stackrel{\text { def }}{=}\left\{\mathfrak{s}_{\mathcal{T}}(x) \mid \mathrm{C}^{q}(x)\right.$ occurs in $\left.\mathfrak{M}(\mathcal{T})\right\}$,
- $\alpha_{\mathcal{T}}(\mathbf{I}) \stackrel{\text { def }}{=}\left\{\left\langle\mathfrak{s} \mathcal{T}\left(x_{1}\right), \ldots, \mathfrak{s}_{\mathcal{T}}\left(x_{\#(\mathrm{I})}\right)\right\rangle \mid \mathbf{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)\right.$ occurs in $\left.\mathfrak{M}(\mathcal{T})\right\}$, and
- $\mathrm{m}_{\mathcal{T}} \stackrel{\text { def }}{=}\left\{q\left[\mathfrak{s}_{\mathcal{T}}(x)\right] \mid \mathrm{C}^{q}(x)\right.$ occurs in $\left.\mathfrak{M}(\mathcal{T})\right\}$.

Lemma 3. Given a sentence $\phi$, for each rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, we have $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right) \vDash \mathfrak{F}(\mathcal{T})$.

Proof. We prove a more general statement, namely if $\phi$ is a formula, not necessarily a sentence, then $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right) \models{ }^{\mathfrak{s} \mathcal{T}} \mathfrak{M}(\mathcal{T})$. The proof goes by induction on the structure of $\mathcal{T}$, distinguishing the cases below:

- $\operatorname{nodes}(\mathcal{T})=\{\epsilon\}$ and $\mathcal{T}(\epsilon)=\left(\mathcal{A}(x) \leftarrow \mathcal{C}^{q}(x)\right)$ is a rule of type ( $\left.\sqrt{\mathbb{I}}\right)$ : then $\alpha_{\mathcal{T}}(\mathrm{C})=\left\{\mathfrak{s}_{\mathcal{T}}(x)\right\}=\{\epsilon\}, \mathrm{m}_{\mathcal{T}}=\left\{q\left[\mathfrak{s}_{\mathcal{T}}(x)\right]\right\}=\{q[\epsilon]\}, \alpha_{\mathcal{T}}\left(\mathrm{C}^{\prime}\right)=\emptyset$, for all $\mathrm{C}^{\prime} \in \mathfrak{C} \backslash\{\mathrm{C}\}$ and $\alpha_{\mathcal{T}}(\mathrm{I})=\emptyset$, for all $\mathrm{I} \in \mathfrak{I}$, thus $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right) \models^{\mathfrak{s} \mathcal{T}} \mathrm{C}^{q}(x)$.
- $\operatorname{nodes}(\mathcal{T}) \neq\{\epsilon\}$ and $\mathcal{T}(\epsilon)=\left(\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \exists y_{1} \ldots \exists y_{n} \cdot \phi * \psi *\right.$ $\left.*_{i=1}^{p} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)\right)$ is a rule of type (II). By the definition of $\mathfrak{s} \mathcal{T}$, we have $\mathfrak{s}_{\mathcal{T}}(x)=i \cdot \mathfrak{s}_{\mathcal{Z}_{i}}(x)$, for all $x \in \operatorname{fv}\left(\mathfrak{M}\left(\mathcal{T}_{\downarrow_{i}}\right)\right)$ and $i \in[1, p]$. We define the structures $\left(\alpha_{0}, \mathrm{~m}_{0}\right),\left(\alpha_{1}, \mathrm{~m}_{1}\right), \ldots,\left(\alpha_{p}, \mathrm{~m}_{p}\right)$ as follows:
$-\alpha_{0}(\mathrm{C}) \stackrel{\text { def }}{=}\left\{\mathfrak{s}_{\mathcal{T}}(x)\right\}=\{\epsilon\}$ and $\mathrm{m}_{0} \stackrel{\text { def }}{=}\{q[\mathfrak{s} \mathcal{T}(x)]\}=\{q[\epsilon]\}$ if $\phi=\mathrm{C}^{q}\left(x_{1}\right)$, else $\alpha_{0}(\mathrm{C})=\emptyset$ and $\mathrm{m}_{0}=\emptyset$ and, moreover, $\alpha_{0}(\mathrm{I}) \stackrel{\text { def }}{=}\left\{\left\langle\mathfrak{s}_{\mathcal{T}}\left(x_{1}\right), \ldots\right.\right.$, $\left.\mathfrak{s}^{\mathcal{T}}\left(x_{\#(\mathrm{I})}\right)\right\rangle \mid \mathrm{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)$ occurs in $\left.\psi\right\}$, thus we have $\left(\alpha_{0}, \mathrm{~m}_{0}\right) \nmid{ }^{\mathfrak{s} \mathcal{T}}$ $\phi * \psi$.
$-\alpha_{i}(\mathrm{C}) \stackrel{\text { def }}{=}\left\{\mathfrak{s}_{\mathcal{T}}(x) \mid \mathrm{C}^{q}(x)\right.$ occurs in $\left.\mathfrak{M}\left(\mathcal{T} \downarrow_{i}\right)\right\}, \alpha_{i}(\mathrm{I}) \stackrel{\text { def }}{=}\left\{\left\langle\mathfrak{s}_{\mathcal{T}}\left(x_{1}\right), \ldots\right.\right.$, $\left.\mathfrak{s}^{\mathcal{T}}\left(x_{\#(\mathrm{I})}\right)\right\rangle \mid \mathrm{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)$ occurs in $\left.\mathfrak{M}\left(\mathcal{T} \downarrow_{i}\right)\right\}$ and $\mathrm{m}_{i} \stackrel{\text { def }}{=}\{q[\mathfrak{s} \mathcal{T}(x)] \mid$ $\mathrm{C}^{q}(x)$ occurs in $\left.\mathfrak{M}\left(\mathcal{T} \downarrow_{i}\right)\right\}$, for each $i \in[1, p]$. Since $\mathfrak{s}_{\mathcal{T}}(x)=i \cdot \mathfrak{s}_{\mathcal{J}_{i}}(x)$, for all $x \in \eta_{i}$ and $i \in[1, p]$, we have $\alpha_{i}(\mathrm{C})=\left\{i \cdot u \mid u \in \alpha_{\mathcal{I}_{i}}(\mathrm{C})\right\}$, for all $\mathrm{C} \in \mathfrak{C}, \mathrm{m}_{i}=\left\{q[i \cdot u] \mid q[u] \in \mathrm{m}_{\mathcal{T}_{i}}\right\}$ and $\alpha_{i}(\mathrm{I})=\left\{\left\langle i \cdot u_{1}, \ldots, i \cdot u_{\#(\mathrm{I})}\right\rangle\right.$ $\left.\mid\left\langle u_{1}, \ldots, u_{\#(\mathrm{I})}\right\rangle \in \alpha_{\mathcal{T}_{i}}(\mathrm{I})\right\}$, for all $\mathrm{I} \in \mathfrak{I}$. By the inductive hypothesis, we have $\left(\alpha_{\mathcal{J}_{i}}, \mathrm{~m}_{\mathcal{J}_{i}}\right) \models^{\mathfrak{s} \mathcal{J}_{i}} \eta_{i}$, thus $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s} \mathcal{T}} \eta_{i}$, for all $i \in[1, p]$.
Since the sets of indices of $\alpha_{0}, \ldots, \alpha_{p}$ and $\mathrm{m}_{0}, \ldots, \mathrm{~m}_{p}$ are pairwise disjoint, their composition is defined and, moreover, we have $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)=$ $\left(\alpha_{0}, \mathrm{~m}_{0}\right) \bullet \ldots \bullet\left(\alpha_{p}, \mathrm{~m}_{p}\right)$. Consequently, we obtain $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right) \models^{\mathfrak{s} \mathcal{T}} \phi * \psi *$ $\boldsymbol{*}_{i=1}^{p} \mathcal{A}_{i}\left(z_{1}^{i}, \ldots, z_{\#\left(\mathcal{A}_{i}\right)}^{i}\right)$, as required.
Example 7. Since $\mathcal{T}$ is a rewriting tree for a sentence, each variable occurs bound in $\mathfrak{F}(\mathcal{T})$ and is mapped by the store $\mathfrak{s} \mathcal{T}$ into the unique node of $\mathcal{T}$ that contains a unique component atom in which that variable occurs:

$$
\begin{aligned}
& \mathfrak{s}_{\mathcal{T}}\left(r^{\epsilon}\right)=\epsilon \quad \mathfrak{s}_{\mathcal{T}}\left(n_{1}^{\epsilon}\right)=1 \quad \mathfrak{s}_{\mathcal{T}}\left(n_{2}^{\epsilon}\right)=2 \\
& \mathfrak{s}_{\mathcal{T}}\left(n_{1}^{1}\right)=11 \quad \mathfrak{s}_{\mathcal{T}}\left(n_{2}^{1}\right)=12 \quad \mathfrak{s} \mathcal{T}\left(n_{1}^{2}\right)=21 \quad \mathfrak{s}_{\mathcal{T}}\left(n_{2}^{2}\right)=22 \\
& \mathfrak{s} \mathcal{T}\left(\ell_{1}^{\epsilon}\right)=111 \quad \mathfrak{s} \mathcal{T}\left(r_{1}^{1}\right)=112 \quad \mathfrak{s} \mathcal{T}\left(\ell_{2}^{1}\right)=121 \quad \mathfrak{s}_{\mathcal{T}}\left(r_{1}^{\epsilon}\right)=122 \\
& \mathfrak{s}_{\mathcal{T}}\left(\ell_{2}^{\epsilon}\right)=211 \quad \mathfrak{s} \mathcal{T}\left(r_{1}^{2}\right)=212 \quad \mathfrak{s}_{\mathcal{T}}\left(\ell_{2}^{2}\right)=221 \quad \mathfrak{s}_{\mathcal{T}}\left(r_{2}^{\epsilon}\right)=222
\end{aligned}
$$

The configuration $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)$ corresponding to the rewriting tree $\mathcal{T}$ from Fig-
ure 5 (a) is given below:

$$
\begin{aligned}
\alpha_{\mathcal{T}}(\mathrm{N})= & \{\epsilon, 1,2,11,12,21,22\} \\
\alpha_{\mathcal{T}}(\mathrm{L})= & \{111,112,121,122,211,212,221,222\} \\
\alpha_{\mathcal{T}}(\mathrm{R})= & \{\langle\epsilon, 1,2\rangle,\langle 1,11,12\rangle,\langle 11,111,112\rangle,\langle 12,121,122\rangle,\langle 21,211,212\rangle, \\
& \langle 22,221,222\rangle\} \\
\alpha_{\mathcal{T}}(\mathrm{I})= & \{\langle 111,112\rangle,\langle 112,121\rangle,\langle 121,122\rangle,\langle 211,212\rangle,\langle 212,221\rangle,\langle 221,222\rangle, \\
& \langle 222,111\rangle\} \\
\mathrm{m}_{\mathcal{T}}= & \left\{q_{0}[\epsilon], q_{0}[1], q_{0}[2], q_{0}[11], q_{0}[12], q_{0}[21], q_{0}[22], s_{0}[111], s_{0}[112],\right. \\
& \left.s_{0}[121], s_{0}[122], s_{0}[211], s_{0}[212], s_{0}[221], s_{0}[222]\right\}
\end{aligned}
$$ the architectures $\alpha_{1}$ and $\alpha_{2}$, where $\alpha_{1}\left(\mathrm{C}_{1}\right)=\alpha_{1}\left(\mathrm{C}_{2}\right)=\{1\}, \alpha_{1}\left(\mathrm{I}_{1}\right)=\{\langle 1,1\rangle\}$ and $\alpha_{2}\left(\mathrm{C}_{1}\right)=\{1\}, \alpha_{2}\left(\mathrm{C}_{2}\right)=\{2\}, \alpha_{2}\left(\mathrm{I}_{1}\right)=\{\langle 1,2\rangle\}$ have the same shape (assuming $N=2$ and $M=1$ ). However, this isomorphism cannot be captured by a global permutation of indices, as it is usually the cass ${ }^{4}$ in the literature 13, because the sets $\alpha_{1}\left(\mathrm{C}_{1}\right) \cup \alpha_{1}\left(\mathrm{C}_{2}\right)$ and $\alpha_{2}\left(\mathrm{C}_{1}\right) \cup \alpha_{2}\left(\mathrm{C}_{2}\right)$ have different cardinalities. For this reason, our definition of symmetry considers one permutation per component type.

Formally, given an architecture $\alpha$ over $\Sigma$ and a tuple of bijections $\mathbf{f}=$ $\left\langle f_{1}, \ldots, f_{N}\right\rangle$, where each $f_{i}: \mathbb{U} \rightarrow \mathbb{U}$ renames the indices of the component type $\mathrm{C}_{i}$, for all $i \in[1, N]$, we define:

$$
\begin{aligned}
&(\mathbf{f}(\alpha))\left(\mathrm{C}_{i}\right) \stackrel{\text { def }}{=} f_{i}\left(\alpha\left(\mathrm{C}_{i}\right)\right), \text { for all } i \in[1, N] \\
&(\mathbf{f}(\alpha))(\mathrm{I}) \stackrel{\text { def }}{=}\left\{\left\langle f_{i_{1}}\left(u_{1}\right), \ldots, f_{i_{\#(\mathrm{I})}}\left(u_{\#(\mathrm{I})}\right)\right\rangle \mid\left\langle u_{1}, \ldots, u_{\#(\mathrm{I})}\right\rangle \in \alpha(\mathrm{I}),\right. \\
&\left.\forall k \in[1, \#(\mathrm{I})] . \operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{k}\right)=\mathrm{C}_{i_{k}}\right\}
\end{aligned}
$$

For a marking m of the Petri net $\mathrm{N}(\Sigma, \alpha, \beta)$, we define, moreover:

$$
\mathbf{f}(\mathrm{m}) \stackrel{\text { def }}{=}\left\{q\left[f_{i}(u)\right] \mid q[u] \in \mathrm{m}, q \in \text { states }_{\beta}\left(\mathrm{C}_{i}\right), i \in[1, N]\right\}
$$

Since the sets states ${ }_{\beta}\left(\mathrm{C}_{i}\right)$, for $i \in[1, N]$, are assumed to be pairwise disjoint, for each state $q$ there is exactly one component type $\mathrm{C}_{i}$, such that $q \in \operatorname{states}_{\beta}\left(\mathrm{C}_{i}\right)$.

[^3]Definition 8. Two configurations are symmetric, denoted $\left(\alpha_{1}, \mathrm{~m}_{1}\right) \sim\left(\alpha_{2}, \mathrm{~m}_{2}\right)$, by taking the profile $\lambda_{\Delta}\left(\right.$ Chain $\left._{n, t}\right)=\langle\mathrm{S}, \mathrm{S}\rangle$, for all constants $n, t \geq 0$ (Example 5). Analogously, the predicate atoms $\operatorname{Node}(n, \ell, r)$ are tight, by taking the profile $\lambda_{\Delta}($ Node $)=\langle\mathrm{N}, \mathrm{L}, \mathrm{L}\rangle$ and $\lambda_{\Delta}($ Leaf $)=\langle\mathrm{L}\rangle$ (Example 6). The above conditions guarantee the tightness of architectures described by tight formulæ: if and only if there exists a tuple of bijections $\mathbf{f}=\left\langle f_{1}, \ldots, f_{N}\right\rangle$, where $f_{i}: \mathbb{U} \rightarrow \mathbb{U}$, for all $i \in[1, N]$, such that $\mathbf{f}\left(\alpha_{1}\right)=\alpha_{2}$ and $\mathbf{f}\left(\mathrm{m}_{1}\right)=\mathrm{m}_{2}$.

The main idea of using symmetries is to prove that models of the same characteristic formula $\mathfrak{F}(\mathcal{T})$ of some rewriting tree $\mathcal{T}$ are symmetric and, in particular, symmetric with the canonical model $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)$. This proof is greatly simplified by considering only those architectures in which all interactions are tightly connected to components, in the following sense:
Definition 9. An architecture $\alpha$ is tight if and only if, for each interaction $\mathrm{I}\left[u_{1}, \ldots, u_{\#(\mathrm{I})}\right]$ from $\alpha$ and each $k \in[1, \#(\mathrm{I})]$, we have $u_{k} \in \alpha(\mathrm{C})$ where $\mathrm{C}=$ $\left.\operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{k}\right)\right)$, that is, the unique component type such that $\langle\mathfrak{P}(\mathrm{I})\rangle_{k} \in \mathfrak{P}(\mathrm{C})$.

Note that the interactions from an architecture $\alpha$ having unconnected ports cannot fire in the Petri net $\mathrm{N}(\Sigma, \alpha, \beta)$ (Def. 1 ), thus having no impact on the answer of a decision problem (deadlock, reachability). However, such interactions are important for the definition of the composition operation (Def. 4). For instance, the tight architecture from Example 1 can only be decomposed into loose non-empty architectures. Consequently, the syntax of CL formulæ does not impose any restriction that guarantee tightness of the architectures defined. This is achieved by an easy-to-check condition below (Def. 10. In general, the tightness problem is decidable and its complexity has been studied for several fragments of CL [14].
Definition 10. A profile is a function $\lambda: \mathbb{A} \rightarrow \bigcup_{i \geq 1} \mathfrak{C}^{i}$, that associates each predicate symbol $\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right)$ of non-zero arity with a tuple of component types of length $\#(\mathcal{A}) \geq 1$. A formula $\phi$ is tight for a profile $\lambda$ if and only if it contains, for each interaction atom $\mathrm{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)$ and each $k \in[1, \#(\mathrm{I})]$ :

- a component atom $\mathrm{C}^{q}\left(x_{k}\right)$, such that $\langle\mathfrak{P}(\mathrm{I})\rangle_{k} \in \mathfrak{P}(\mathrm{C})$, or
- a predicate atom $\mathcal{A}\left(y_{1}, \ldots, y_{\#(\mathcal{A})}\right)$, such that $x_{k}=y_{\ell}$ and $\langle\mathfrak{P}(\mathrm{I})\rangle_{k} \in$ $\mathfrak{P}\left(\langle\lambda(\mathcal{A})\rangle_{\ell}\right)$, for some $\ell \in[1, \#(\mathcal{A})]$.
A SID $\Delta$ is tight if and only if there exists a profile $\lambda_{\Delta}$, such that, for each rule $\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \phi$ from $\Delta$, the formula $\phi$ is tight for $\lambda_{\Delta}$ and contains, for each $k \in[1, \#(\mathcal{A})]$ :
- a component atom $\mathrm{C}^{q}\left(x_{k}\right)$, such that $\left\langle\lambda_{\Delta}(\mathcal{A})\right\rangle_{k}=\mathrm{C}$, or
- a predicate atom $\mathcal{B}\left(y_{1}, \ldots, y_{\#(\mathcal{A})}\right)$, such that $x_{k}=y_{\ell}$ and $\left\langle\lambda_{\Delta}(\mathcal{A})\right\rangle_{k}=$ $\left\langle\lambda_{\Delta}(\mathcal{B})\right\rangle_{\ell}$, for some $\ell \in[1, \#(\mathcal{B})]$.
A formula $\phi$, interpreted over a tight SID $\Delta$, is tight if and only if it is tight for the profile $\lambda_{\Delta}$.

For instance, one can check that the predicate atoms $\operatorname{Chain}_{n, t}(x, y)$ are tight, Lemma 4. For each model ( $\alpha, \mathrm{m}$ ) of a tight sentence, the architecture $\alpha$ is tight.

Proof. Let $\phi$ be a tight sentence, such that $(\alpha, \mathrm{m}) \vDash \phi$. By Lemma 1 there exists a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, such that $(\alpha, \mathrm{m}) \models \mathfrak{F}(\mathcal{T})$. Let $\mathrm{I}\left(z_{1}, \ldots, z_{\#(\mathrm{I})}\right)$ be an interaction atom from $\mathfrak{F}(\mathcal{T})$ and $k \in[1, \#(\mathbf{I})]$. Also, let $w \in \operatorname{nodes}(\mathcal{T})$ be the node where this interaction atom was introduced, such that $\mathcal{T}(w)=$ $\left(\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \varphi\right)$ and $\mathrm{I}\left(y_{1}, \ldots, y_{\#(\mathrm{I})}\right)$ occurs in $\varphi$, where $y_{1}, \ldots, y_{\#(\mathrm{I})}$ are substituted by $z_{1}, \ldots, z_{\#(\mathcal{A})}$, respectively, during the construction of $\mathfrak{F}(\mathcal{T})$. Since the SID $\Delta$, that interprets the formula $\phi$, is tight, the rule body $\varphi$ is tight, hence one of the following holds:

- there exists a component atom $\mathrm{C}^{q}\left(y_{k}\right)$ in $\varphi$, such that $\langle\mathfrak{P}(\mathrm{I})\rangle_{k}=\mathrm{C}$, or
- there exists a predicate atom $\mathcal{B}\left(\xi_{1}, \ldots, \xi_{\#(\mathcal{B})}\right)$ in $\varphi$, such that $y_{k}=\xi_{\ell}$ and $\langle\mathfrak{P}(\mathrm{I})\rangle_{k}=\left\langle\lambda_{\Delta}(\mathcal{B})\right\rangle_{\ell}$, where $\lambda_{\Delta}$ is the profile from Def. 10. In this case, we apply induction to prove the existence of a component atom $\mathrm{C}^{q}\left(\xi_{k}\right)$ in $\mathfrak{F}\left(\mathcal{T} \downarrow_{i}\right)$, such that $\langle\mathfrak{P}(\mathrm{I})\rangle_{k}=\mathrm{C}$, where $i \in\left[1, \#_{\text {pred }}(\varphi)\right]$ is the child of $w$ corresponding to $\mathcal{B}\left(\xi_{1}, \ldots, \xi_{\#(\mathcal{B})}\right)$ in $\mathcal{T}$.
In both cases, $\mathfrak{F}(\mathcal{T})$ contains a component atom $\mathrm{C}^{q}\left(z_{k}\right)$, such that $\langle\mathfrak{P}(\mathrm{I})\rangle_{k}=\mathrm{C}$, hence $(\alpha, \mathrm{m})$ is tight, by Def. 9 .

The next result proves that each model of a tight sentence is necessarily symmetric to a canonical model of that sentence:
Proposition 3. Given a tight sentence $\phi$, for each $\mathcal{T} \in \mathbb{T}(\phi)$, if $(\alpha, \mathrm{m}) \models \mathfrak{F}(\mathcal{T})$, then $(\alpha, \mathrm{m}) \sim\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)$.

Proof. Since $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right) \models \mathfrak{F}(\mathcal{T})$, by Lemma 3] it is sufficient to prove that $\left(\alpha_{1}, \mathrm{~m}_{1}\right) \sim\left(\alpha_{2}, \mathrm{~m}_{2}\right)$, for any two configurations $\left(\alpha_{i}, \mathrm{~m}_{i}\right)$, such that $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models$ $\mathfrak{F}(\mathcal{T})$. Let $\mathfrak{F}(\mathcal{T}) \stackrel{\text { def }}{=} \exists x_{1} \ldots \exists x_{n} . \eta$, where $\eta$ is a quantifier- and predicate-free formula. Then $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models \mathfrak{F}(\mathcal{T})$ if and only if there exist stores $\mathfrak{s}_{i}$, such that $\left(\alpha_{i}, \mathrm{~m}_{i}\right) \models^{\mathfrak{s}_{i}} \eta$, for $i=1,2$. We define the sets $U_{j}^{i} \stackrel{\text { def }}{=}\left\{\mathfrak{s}_{i}(x) \mid C_{j}^{q}(x)\right.$ occurs in $\left.\eta\right\}$ and the bijections $f_{j}: U_{j}^{1} \rightarrow U_{j}^{2}$ as $f_{j}\left(\mathfrak{s}_{1}(x)\right)=\mathfrak{s}_{2}(x)$, for all variables $x$, such that $\mathrm{C}_{j}^{q}(x)$ occurs in $\eta$. These bijections are extended to bijections $\bar{f}_{j}: \mathbb{U} \rightarrow \mathbb{U}$, by the following fact:

Fact 1. Given finite sets $U_{1}, U_{2} \subseteq \mathbb{U}$, such that $f: U_{1} \rightarrow U_{2}$ is a bijection, there exists a bijection $\bar{f}: \mathbb{U} \rightarrow \mathbb{U}$, such that $\bar{f}(u)=f(u)$, for all $u \in U_{1}$.

Proof. Since $f$ is a bijection, we have $\left\|U_{1}\right\|=\left\|U_{2}\right\|$ and, since $\left\|U_{i}\right\|=\left\|U_{i} \backslash U_{3-i}\right\|+$ $\left\|U_{1} \cap U_{2}\right\|$, for $i=1,2$, we obtain $\left\|U_{1} \backslash U_{2}\right\|=\left\|U_{2} \backslash U_{1}\right\|$. Hence there exists a bijection $\bar{f}: \mathbb{U} \rightarrow \mathbb{U}$, such that $\bar{f}(u)=f(u)$, for all $u \in U_{1}, \bar{f}\left(U_{2} \backslash U_{1}\right)=U_{1} \backslash U_{2}$ and $\bar{f}\left(\mathbb{U} \backslash\left(U_{1} \cup U_{2}\right)\right)=f\left(\mathbb{U} \backslash\left(U_{1} \cup U_{2}\right)\right)$.
For $\mathbf{f} \stackrel{\text { def }}{=}\left\langle\bar{f}_{1}, \ldots, \bar{f}_{N}\right\rangle$, we prove the following points:

- $\mathbf{f}\left(\alpha_{1}\right)=\alpha_{2}$ : by the definition of $\bar{f}_{j}$, we have $\bar{f}_{j}\left(\alpha_{1}\left(\mathrm{C}_{j}\right)\right)=\alpha_{2}\left(\mathrm{C}_{j}\right)$, for all $j \in[1, N]$. Let $\mathrm{I} \in \mathfrak{I}$ and we prove $\alpha_{2}(\mathrm{I})=\left\{\left\langle\bar{f}_{i_{1}}\left(u_{1}\right), \ldots, \bar{f}_{i_{\#(I)}}\left(u_{\#(\mathrm{I})}\right)\right\rangle \mid\right.$ $\left.\left\langle u_{1}, \ldots, u_{\#(\mathrm{I})}\right\rangle \in \alpha_{1}(\mathrm{I}), \forall k \in[1, \#(\mathrm{I})] \cdot \operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{k}\right)=\mathrm{C}_{i_{k}}\right\}$. " $\subseteq$ " Let $\left\langle u_{1}, \ldots, u_{\#(I)}\right\rangle \in \alpha_{2}(\mathrm{I})$. Then there exists an interaction atom $\mathrm{I}\left(x_{1}, \ldots\right.$, $\left.x_{\#(\mathrm{I})}\right)$ in $\eta$, such that $u_{i}=\mathfrak{s}_{2}\left(x_{i}\right)$, for $i \in[1, \#(\mathrm{I})]$. Since $\phi$ is a tight sentence and $\left(\alpha_{2}, \mathrm{~m}_{2}\right) \models \phi$, by Lemma 4 , the architecture $\alpha_{2}$ is tight (Def. 9). Then, for each $k \in[1, \#(\mathrm{I})]$, we have $u_{k} \in \alpha_{2}\left(\mathrm{C}_{i_{k}}\right)$, where $\mathrm{C}_{i_{k}} \in \mathfrak{C}$ is
the unique component type such that $\langle\mathfrak{P}(\mathrm{I})\rangle_{k} \in \mathfrak{P}\left(\mathrm{C}_{i_{k}}\right)$. By the previous point, there exists $u_{k}^{\prime} \in \alpha_{1}\left(\mathrm{C}_{i_{k}}\right)$, such that $\bar{f}_{i_{k}}\left(u_{k}^{\prime}\right)=u_{k}=\mathfrak{s}_{2}\left(x_{k}\right)$. By the definition of $\bar{f}_{i_{k}}$, we have $u_{k}^{\prime}=\mathfrak{s}_{1}\left(x_{k}\right)$, hence $\left\langle u_{1}^{\prime}, \ldots, u_{\#(\mathrm{II})}^{\prime}\right\rangle \in \alpha_{1}(\mathrm{I})$. " $\supseteq$ " Let $\left\langle u_{1}, \ldots, u_{\#(\mathrm{I})}\right\rangle \in \alpha_{1}(\mathrm{I})$, such that $\operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{k}\right)=\mathrm{C}_{i_{k}}$, for all $k \in[1, \#(\mathrm{I})]$. Then there exists an interaction atom $\mathrm{I}\left(x_{1}, \ldots, x_{\#}(\mathrm{I})\right.$ in $\eta$, such that $u_{k}=\mathfrak{s}_{1}\left(x_{k}\right)$, for all $k \in[1, \#(\mathrm{I})]$. Since $\phi$ is a tight sentence and $\left(\alpha_{1}, \mathrm{~m}_{1}\right) \models \phi$, by Lemma 4 , the architecture $\alpha_{1}$ is tight, thus $\mathfrak{s}_{1}\left(x_{k}\right)=$ $u_{k} \in \alpha_{1}\left(\mathrm{C}_{i_{k}}\right)$. By the definition of $\bar{f}_{i_{k}}$, we have $\mathfrak{s}_{2}\left(x_{k}\right)=\bar{f}_{i_{k}}\left(u_{k}\right)$, for all $k \in[1, \#(\mathrm{I})]$, hence $\left\langle\bar{f}_{i_{1}}\left(u_{1}\right), \ldots, \bar{f}_{i_{\#(I)}}\left(u_{\#(\mathrm{I})}\right)\right\rangle \in \alpha_{2}(\mathrm{I})$.
- $\mathbf{f}\left(\mathrm{m}_{1}\right)=\mathrm{m}_{2}$ : we prove $\mathrm{m}_{2}=\left\{q\left[\bar{f}_{i}(u)\right] \mid q[u] \in \mathrm{m}_{1}, q \in \operatorname{states}_{\beta}\left(\mathrm{C}_{i}\right), i \in[1, N]\right\}$. " $\subseteq$ " Let $q[u] \in \mathrm{m}_{2}$. Since $\left(\alpha_{2}, \mathrm{~m}_{2}\right) \models^{\mathfrak{s}_{2}} \eta$, there exists a component atom $\mathrm{C}_{i}^{q}(x)$ in $\eta$, such that $u=\mathfrak{s}_{2}(x) \in \alpha_{2}\left(\mathrm{C}_{i}\right)$ and $\mathrm{C}_{i}$ is the unique component type, such that $q \in$ states $_{\beta}\left(\mathrm{C}_{i}\right)$. By the definition of $\bar{f}_{i}$, we have $\mathfrak{s}_{1}(x) \in \alpha_{1}\left(\mathrm{C}_{i}\right)$ and $\mathfrak{s}_{2}(x)=\bar{f}_{i}\left(\mathfrak{s}_{1}(x)\right)$. Moreover, since $\left(\alpha_{1}, \mathrm{~m}_{1}\right) \models^{\mathfrak{s}_{1}} \eta$, we have $q\left[\mathfrak{s}_{1}(x)\right] \in \mathrm{m}_{1}$. "Э" Let $q[u] \in \mathrm{m}_{1}$ and $q \in$ states $_{\beta}\left(\mathrm{C}_{i}\right)$, for some $i \in[1, N]$. Since $\left(\alpha_{1}, \mathrm{~m}_{1}\right) \models^{\mathfrak{s}_{1}} \eta$, there exists a component atom $\mathrm{C}^{q}(x)$ in $\eta$, such that $u=\mathfrak{s}_{1}(x)$. By the definition of $\bar{f}_{i}$, we have $\bar{f}_{i}\left(\mathfrak{s}_{1}(x)\right)=\mathfrak{s}_{2}(x)$, hence $q\left[\bar{f}_{i}(u)\right]=q\left[\mathfrak{s}_{2}(x)\right] \in \mathrm{m}_{2}$, since $\left(\alpha_{2}, \mathrm{~m}_{2}\right) \models^{\mathfrak{s}_{2}} \eta$.
Moreover, the symmetry relation is preserved by the Petri net describing the operational semantics of the system (Def. 1]:
Lemma 5. Given a tight architecture $\alpha$ and a behavior $\beta$, over a signature $\Sigma=\left(\left\{\mathrm{C}_{1}, \ldots, \mathrm{C}_{N}\right\}, \mathfrak{I}, \mathfrak{P}\right)$, for any tuple of bijections $\mathbf{f}=\left\langle f_{1}, \ldots, f_{N}\right\rangle$, where $f_{i}: \mathbb{U} \rightarrow \mathbb{U}$, for all $i \in[1, N]$, the following hold:

1. $\mathrm{m}^{\prime} \in \operatorname{Reach}(\mathrm{N}(\Sigma, \alpha, \beta), \mathrm{m}) \Longleftrightarrow \mathbf{f}\left(\mathrm{m}^{\prime}\right) \in \operatorname{Reach}(\mathrm{N}(\Sigma, \mathbf{f}(\alpha), \beta), \mathbf{f}(\mathrm{m}))$,
2. $\mathrm{m} \in \operatorname{Dead}(\mathrm{N}(\Sigma, \alpha, \beta)) \Longleftrightarrow \mathbf{f}(\mathrm{m}) \in \operatorname{Dead}(\mathrm{N}(\Sigma, \mathbf{f}(\alpha), \beta))$,
for any two markings $\mathrm{m}, \mathrm{m}^{\prime}$ of the Petri net $\mathrm{N}(\Sigma, \alpha, \beta)$.
Proof. By Def. 1. let $\mathrm{N}(\Sigma, \mathbf{f}(\alpha), \beta)$ be the Petri net $\left(S^{\mathbf{f}}, T^{\mathbf{f}}, E^{\mathbf{f}}\right)$, where:

$$
\begin{aligned}
S^{\mathbf{f}}= & \bigcup_{i=1}^{N}\left\{q[u] \mid q \in \operatorname{states}\left(\mathrm{C}_{i}\right), u \in(\mathbf{f}(\alpha))\left(\mathrm{C}_{i}\right)\right\} \\
= & \bigcup_{i=1}^{N}\left\{q\left[f_{i}(u)\right] \mid q \in \operatorname{states}\left(\mathrm{C}_{i}\right), u \in \alpha\left(\mathrm{C}_{i}\right)\right\} \\
T^{\mathbf{f}}= & \bigcup_{\mathbf{1} \in \mathfrak{J}}\left\{\left(I\left[u_{1}, \ldots, u_{\#(\mathrm{I}}\right),\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right) \mid\left\langle p_{1}, \ldots, p_{\#(\mathrm{I}}\right\rangle=\mathfrak{P}(\mathrm{I}),\right. \\
& \quad\left\langle u_{1}, \ldots, u_{\#(\mathrm{I})}\right\rangle \in(\mathbf{f}(\alpha))(\mathrm{I}),\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle \in T_{\beta}(\mathrm{I}), \\
& \left.\forall i, j \in[1, \#(\mathrm{I})] \cdot i \neq j \Rightarrow u_{i} \neq u_{j} \text { or } \operatorname{comp}\left(p_{i}\right) \neq \operatorname{comp}\left(p_{j}\right)\right\} \\
= & \bigcup_{\mathbf{I} \in \mathcal{I}}\left\{\left(\mathrm{I}\left[f_{k_{1}}\left(u_{1}\right), \ldots, f_{k_{\#(I)}}\left(u_{\#(\mathrm{I})}\right)\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right) \mid\right. \\
& \left\langle p_{1}, \ldots, p_{\#(\mathrm{I}}\right\rangle=\mathfrak{P}(\mathrm{I}), \forall i \in[1, \#(\mathrm{I})] . \mathrm{C}_{k_{i}}=\operatorname{comp}\left(p_{i}\right), \\
& \left\langle u_{1}, \ldots, u_{\#(\mathrm{I})}\right\rangle \in \alpha(\mathrm{I}),\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle \in T_{\beta}(\mathrm{I}), \\
& \left.\forall i, j \in[1, \#(\mathrm{I})] \cdot i \neq j \Rightarrow f_{k_{i}}\left(u_{i}\right) \neq f_{k_{j}}\left(u_{j}\right) \text { or } \operatorname{comp}\left(p_{i}\right) \neq \operatorname{comp}\left(p_{j}\right)\right\}
\end{aligned}
$$

$$
\begin{aligned}
E^{\mathbf{f}}=\bigcup_{\mathrm{I} \in \mathfrak{I}}\{ & \left(q\left[f_{k_{i}}\left(u_{i}\right)\right],\left(\mathrm{I}\left[f_{k_{1}}\left(u_{1}\right), \ldots, f_{k_{\#(\mathrm{I})}}\left(u_{\#(\mathrm{I})}\right)\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right)\right), \\
& \left(\left(\mathrm{I}\left[f_{k_{1}}\left(u_{1}\right), \ldots, f_{k_{\#(\mathrm{I})}}\left(u_{\#(\mathrm{I})}\right)\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right), q^{\prime}\left[f_{k_{i}}\left(u_{i}\right)\right]\right) \mid \\
t_{i}= & \left.\left(q \xrightarrow{p_{i}} \operatorname{comp}\left(p_{i}\right) q^{\prime}\right), i \in[1, \#(\mathrm{I})]\right\}
\end{aligned}
$$

We obtain the following equivalence:

$$
\begin{gathered}
\mathrm{m} \xrightarrow{\left(\mathrm{I}\left[u_{1}, \ldots, u_{\#(\mathrm{I})}\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right)} \mathrm{m}^{\prime} \text { in } \mathrm{N}(\Sigma, \alpha, \beta) \\
\mathbf{f}(\mathrm{m}) \xrightarrow{\left(\mathrm{I}\left[f_{k_{1}}\left(u_{1}\right), \ldots, f_{k_{\#(\mathrm{I}}}\left(u_{\#(\mathrm{I})}\right)\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right)} \Longrightarrow \mathbf{f}\left(\mathrm{m}^{\prime}\right) \text { in } \mathrm{N}(\Sigma, \mathbf{f}(\alpha), \beta)
\end{gathered}
$$

where $k_{1}, \ldots, k_{\#(\mathrm{I})} \in[1, N]$ are such that $\mathrm{C}_{k_{i}}=\operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{i}\right)$, for all $i \in$ [1, \#(I)]. Point (1) uses this fact inductively, on the length of the firing sequence leading from m to $\mathrm{m}^{\prime}$. For point $(2)$ we use, moreover, that $q[u] \in \mathrm{m} \Longleftrightarrow$ $q\left[f_{i}(u)\right] \in \mathbf{f}(\mathrm{m})$ and $q[u] \in{ }^{\bullet}\left(\mathrm{I}\left[u_{1}, \ldots, u_{\#(\mathrm{I})}\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right) \Longleftrightarrow q\left[f_{i}(u)\right] \in$ $\bullet\left(\mathrm{I}\left[f_{k_{1}}\left(u_{1}\right), \ldots, f_{k_{\#(\mathrm{I})}}\left(u_{\#(\mathrm{I})}\right)\right],\left\langle t_{1}, \ldots, t_{\#(\mathrm{I})}\right\rangle\right)$, where $\mathrm{C}_{i}$ is the unique component type such that $q \in$ states $_{\beta}\left(\mathrm{C}_{i}\right)$.
Theorem 2. Given a tight sentence $\phi$, interpreted over a SID $\Delta$, a behavior map $\beta$ and a tuple of states $\left\langle q_{1}, \ldots, q_{k}\right\rangle$, the following equivalences hold:

$$
\begin{aligned}
\operatorname{deadlock}_{t}(\phi, \Delta, \beta) & \Longleftrightarrow \operatorname{deadlock}(\phi, \Delta, \beta) \\
\operatorname{reach}_{t}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right) & \Longleftrightarrow \operatorname{reach}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right)
\end{aligned}
$$

where deadlock ${ }_{t}(\phi, \Delta, \beta)$ and reach $_{t}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right)$ are defined below:

- deadlock ${ }_{t}(\phi, \Delta, \beta)$ : does there exist rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, such that $\operatorname{Reach}\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right) \cap \operatorname{Dead}\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)\right) \neq \emptyset$ ?
- reach $_{t}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right)$ : does there exist a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$ and a configuration $\left(\alpha_{\mathcal{T}}, \mathrm{m}\right) \in \operatorname{Reach}\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right)$, such that $\left\{q_{i}\left[u_{i}\right]\right.$ $\mid i \in[1, k]\} \subseteq \mathrm{m}$, with $q_{1}\left[u_{1}\right], \ldots, q_{k}\left[u_{k}\right]$ pairwise distinct?
Proof. We prove the following points:
- $\operatorname{deadlock}_{t}(\phi, \Delta, \beta) \Rightarrow \operatorname{deadlock}(\phi, \Delta, \beta)$ : assume that $\operatorname{deadlock}_{t}(\phi, \Delta, \beta)$ has a positive answer and let $\mathcal{T} \in \mathbb{T}(\phi)$ be a rewriting tree, such that $\operatorname{Reach}\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right) \cap \operatorname{Dead}\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)\right) \neq \emptyset$, the existence of which is stated by deadlock ${ }_{t}(\phi, \Delta, \beta)$. By Lemma 3, $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right) \models \mathfrak{F}(\mathcal{T})$ and, by Lemma 1, we obtain $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right) \models \phi$, thus deadlock $(\phi, \Delta, \beta)$ has a positive answer.
- deadlock $(\phi, \Delta, \beta) \Rightarrow \operatorname{deadlock}_{t}(\phi, \Delta, \beta)$ : assume that $\operatorname{deadlock}(\phi, \Delta, \beta)$ has a positive answer and let ( $\alpha, \mathrm{m}$ ) be a model of $\phi$ and $\mathrm{m}^{\prime}$ be a marking such that $\mathrm{m}^{\prime} \in \operatorname{Reach}(\mathrm{N}(\Sigma, \alpha, \beta), \mathrm{m}) \cap \operatorname{Dead}(\mathrm{N}(\Sigma, \alpha, \beta))$. By Lemma 1, there exists a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, such that $(\alpha, \mathrm{m}) \models \mathfrak{F}(\mathcal{T})$. By Lemma 3. since $\phi$ is a tight sentence, we obtain that $(\alpha, \mathrm{m}) \sim\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)$ and let $\mathbf{f}=\left\langle f_{1}, \ldots, f_{N}\right\rangle$ be the bijections from Def. 8 , such that $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)=$ $(\mathbf{f}(\alpha), \mathbf{f}(\mathrm{m}))$. By Lemma 5 , we obtain that $\mathbf{f}\left(\mathrm{m}^{\prime}\right) \in \operatorname{Reach}(\mathrm{N}(\Sigma, \mathbf{f}(\alpha), \beta), \mathbf{f}(\mathrm{m}))$ $\cap \operatorname{Dead}(\mathrm{N}(\Sigma, \mathbf{f}(\alpha), \beta))$, thus $\operatorname{deadlock}_{t}(\phi, \Delta, \beta)$ has a positive answer.
- $\operatorname{reach}_{t}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right) \Rightarrow \operatorname{reach}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right)$ : assume that $\operatorname{reach}_{t}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right)$ has a positive answer and let $\mathcal{T} \in \mathbb{T}(\phi)$ be a rewriting tree and m be a marking of $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$, such that $\left(\alpha_{\mathcal{T}}, \mathrm{m}\right) \in$ $\operatorname{Reach}\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right)$ and $\left\{q_{i}\left[u_{i}\right] \mid i \in[1, k]\right\} \subseteq \mathrm{m}$, where $q_{1}\left[u_{1}\right], \ldots$, $q_{k}\left[u_{k}\right]$ are pairwise distinct places. By Lemma 3, $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right) \models \mathfrak{F}(\mathcal{T})$ and, by Lemma 1. we obtain $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right) \models \phi$, thus $\operatorname{reach}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right)$ has a positive answer.
- $\operatorname{reach}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right) \Rightarrow \operatorname{reach}_{t}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right)$ : assume that $\operatorname{reach}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right)$ has a positive answer and let $(\alpha, \mathrm{m})$ be a model of $\phi$ and $\mathrm{m}^{\prime} \in \operatorname{Reach}(\mathrm{N}(\Sigma, \alpha, \beta), \mathrm{m})$ be a marking, such that $\left\{q_{i}\left[u_{i}\right] \mid i \in[1, k]\right\} \subseteq$ $\mathrm{m}^{\prime}$, where $q_{1}\left[u_{1}\right], \ldots, q_{k}\left[u_{k}\right]$ are pairwise distinct places of $\mathrm{N}(\Sigma, \alpha, \beta)$. By Lemma 1, there exists a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, such that $(\alpha, \mathrm{m}) \models$ $\mathfrak{F}(\mathcal{T})$. By Lemma 3 , since $\phi$ is a tight sentence, we obtain that $(\alpha, \mathrm{m}) \sim$ $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)$ and let $\mathbf{f}=\left\langle f_{1}, \ldots, f_{N}\right\rangle$ be the bijections from Def. 8, such that $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)=(\mathbf{f}(\alpha), \mathbf{f}(\mathrm{m}))$. By Lemma 5, we obtain that $\mathbf{f}\left(\mathrm{m}^{\prime}\right) \in$ $\operatorname{Reach}(\mathrm{N}(\Sigma, \mathbf{f}(\alpha), \beta), \mathbf{f}(\mathrm{m}))$. Moreover, if $\mathrm{C}_{\ell_{i}}$ is the unique component type, such that $q_{i} \in$ states $_{\beta}\left(\mathrm{C}_{\ell_{i}}\right)$, for all $i \in[1, k]$, since $\left\{q_{i}\left[u_{i}\right] \mid i \in[1, k]\right\} \subseteq$ $\mathrm{m}^{\prime}$, we obtain $\left\{q_{i}\left[f_{\ell_{i}}\left(u_{i}\right)\right] \mid i \in[1, k]\right\} \subseteq \mathbf{f}\left(\mathrm{m}^{\prime}\right)$ and $\operatorname{reach}_{t}\left(\phi,\left\langle q_{1}, \ldots, q_{k}\right\rangle, \Delta, \beta\right)$ has a positive answer.


## 6. Parametric Verification using Structural Invariants

In this section we present a sound (but necessarily incomplete) method to address practical instances of the (undecidable) decision problems of deadlock and reachability (Theorem 1). The method proceeds by synthesizing a sufficient verification condition as a formula in a decidable fragment of MSO, interpreted over trees of bounded arity $\kappa$, with second-order set variables ranging over finite sets. This logic is known as the Weak Second-order calculus of $\kappa$ Successors $(\mathrm{WS} \kappa S)$. Well-known automata-theoretic decision procedures exist for $\mathrm{WS} \kappa S$ [12] and we rely on optimized provers [15] to check the verification conditions.

During the unfolding of the predicate symbols, according to the structure of a rewriting tree $\mathcal{T}$, an existentially quantified variable introduced by a rule might be renamed several times (the renaming occurs when replacing a predicate atom $\mathcal{A}\left(y_{1}, \ldots, y_{\#(\mathcal{A})}\right)$ with one of its definitions $\varphi\left[x_{1} / y_{1} \ldots x_{\#(\mathcal{A})} / y_{\#(\mathcal{A})}\right]$, for some rule $\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \varphi$ from the SID) before it is instantiated by a component atom $\mathrm{C}^{q}(x)$, that occurs in some node $w \in \operatorname{nodes}(\mathcal{T})$. Due to Assumption 2 on the syntax of the rules in the SID, this node is unique and the canonical store $\mathfrak{s} \mathcal{T}$ maps the variable to that node, i.e. $\mathfrak{s} \mathcal{T}(x)=w$. In order to determine this value, we must track $x$ along the path in the rewriting tree that leads from the node where it was introduced (by an existential quantifier, or the root of the tree if the variable is a top-level parameter) to the node where the component atom $\mathrm{C}^{q}(x)$ occurs.

Example 8. Let us consider the rewriting tree $\mathcal{T}$ depicted in Fig. 5(a). The variable $\ell_{1}^{\epsilon}$ introduced by an existential quantifier at the root of the tree replaces the $\ell$ parameter of the rule that labels the left successor of the root, i.e. the
node 1, that replaces the $\ell$ parameter of the label of the node 11, which finally replaces the $x$ parameter of the label of the node 111, that contains a component atom Leaf ${ }^{s_{0}}(x)$. Hence, the value of $\ell_{1}^{\epsilon}$ in the canonical store is $\mathfrak{s}_{\mathcal{T}}\left(\ell_{1}^{\epsilon}\right)=111$. The path from the node where the variable has been introduced by existential quantification and the node where its value is assigned by an instance atom is shown in dashed lines in Fig. 5.

Tracking variables is done by path automata that traverse the tree downwards, following the substitutions of a particular variable. A path automaton is then transformed into an equivalent $\mathrm{W} S \kappa S$ path formula, that defines the set of trees over which the automaton has an accepting run. The path formula is used to define a flow formula that describes the pre- and post-sets for each transition of the Petri net $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$ corresponding to the canonical architecture $\alpha_{\mathcal{T}}$, defined by the rewriting tree $\mathcal{T}$, and the given behavior map $\beta$. Moreover, path formulæ allow to define $\mathrm{WS} \kappa S$ formulæ describing the initial and final configurations of the system.

In particular, the flow formula is used to derive invariants (i.e. over-approximations of the set of reachable configurations) directly from the structure of the Petri net $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$. These invariants are defined by the sets of places of $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$ that do not lose (trap invariants) or generate extra tokens (mutex invariants), respectively. All verification conditions for the problems considered in this paper boil down to checking the satisfiability of a $\mathrm{WS} \kappa S$ formula.

### 6.1. Weak Second Order Calculus of $\kappa$ Successors

We define the logic $\mathrm{WS} \kappa S$, which is a fragment of MSO interpreted over a finite prefix of an infinite $\kappa$-ary tree. As a remainder, the prefix of a tree $\mathcal{T}$ is the restriction of $\mathcal{T}$ to a prefix-closed and complete subset of nodes $(\mathcal{T})$ ( $\$ 5.1$ ).

Let $\mathbb{X}^{2}=\{X, Y, Z, \ldots\}$ be a countably infinite set of second-order variables, ranging over subsets of $[1, \kappa]^{*}$. The formulæ of $\mathrm{WS} \kappa S$ are defined inductively by the following syntax:

$$
\begin{array}{rlrr}
\tau & ::=\epsilon|x| \tau . i & \text { terms } \\
\xi & ::=\tau=\tau|X(\tau)| \xi \wedge \xi|\neg \xi| \exists x \cdot \xi \mid \exists X . \xi & \text { formulæ }
\end{array}
$$

where $x \in \mathbb{X}^{1}, X \in \mathbb{X}^{2}$ and $i \in[1, \kappa]$. As usual, we write $x \neq y \stackrel{\text { def }}{=} \neg x=y$, $\xi_{1} \vee \xi_{2} \stackrel{\text { def }}{=} \neg\left(\neg \xi_{1} \wedge \neg \xi_{2}\right), \xi_{1} \rightarrow \xi_{2} \stackrel{\text { def }}{=} \neg \xi_{1} \vee \xi_{2}, \xi_{1} \leftrightarrow \xi_{2} \stackrel{\text { def }}{=}\left(\xi_{1} \rightarrow \xi_{2}\right) \wedge\left(\xi_{2} \rightarrow \xi_{1}\right)$, $\forall x . \xi \stackrel{\text { def }}{=} \neg \exists x . \neg \xi$ and $\forall X . \xi \stackrel{\text { def }}{=} \neg \exists X . \neg \xi$. For a constant $n \in \mathbb{N}$, we define:

$$
\begin{align*}
X=\left\{y_{1}, \ldots, y_{n}\right\} & \Longleftrightarrow \forall x \cdot X(x) \leftrightarrow \bigvee_{i=1}^{n} x=y_{i}  \tag{12}\\
\|X\| \geq n & \Longleftrightarrow \exists y_{1} \ldots \exists y_{n} \cdot \bigwedge_{1 \leq i<j \leq n} y_{i} \neq y_{j} \wedge \bigwedge_{i=1}^{n} X\left(y_{i}\right)  \tag{13}\\
\|X\|=n & \Longleftrightarrow\|X\| \geq n \wedge \neg\|X\| \geq n+1 \tag{14}
\end{align*}
$$

The variables of a $W S \kappa S$ formula are interpreted by a store $\nu: \mathbb{X}^{1} \cup \mathbb{X}^{2} \rightarrow$ $[1, \kappa]^{*} \cup 2^{[1, \kappa]^{*}}$, such that $\nu(x) \in[1, \kappa]^{*}$, for each $x \in \mathbb{X}^{1}$ and $\nu(X) \subseteq[1, \kappa]^{*}$ is
a finite set, for each $X \in \mathbb{X}^{2}$. The terms of $\mathrm{WS} \kappa S$ are interpreted inductively, as $\nu(\epsilon) \stackrel{\text { def }}{=} \epsilon$ and $\nu(\tau . i) \stackrel{\text { def }}{=} \nu(\tau) \cdot i$. We denote by $\models_{\text {wsks }}^{\nu} \xi$ the fact that the WS $\kappa S$ formula $\xi$ is valid for the valuation $\nu$. This relation defined below, by induction on the structure of the formulæ:

$$
\begin{array}{ll}
\models_{\text {wsks }}^{\nu} \tau_{1}=\tau_{2} & \Longleftrightarrow \nu\left(\tau_{1}\right)=\nu\left(\tau_{2}\right) \\
\models_{\text {wsks }}^{\nu} X(\tau) & \Longleftrightarrow \nu(\tau) \in \nu(X) \\
\models_{\text {wsks }} \xi_{1} \wedge \xi_{2} & \Longleftrightarrow \\
\models_{\text {wsks }}^{\nu} \neg \xi & \Longleftrightarrow \xi_{1} \text { and } \models_{\text {wsks }}^{\nu} \xi_{2} \\
\models_{\text {wsks }}^{\nu} \exists x \cdot \xi & \Longleftrightarrow \models_{\text {wsts }}^{\nu} \models_{\text {wsks }}^{\nu} \xi \\
\models_{\text {wsks }}^{\nu} \exists X \cdot \xi, \text { for some node } w \in[1, \kappa]^{*} \\
& \Longleftrightarrow \\
\models_{\text {wsks }}^{\nu[X \leftarrow S]} \xi, \text { for some finite set } S \subseteq[1, \kappa]^{*}
\end{array}
$$

A $\mathrm{WS} \kappa S$ formula $\xi$ is satisfiable if and only if there exists a store $\nu$, such that $\models_{\text {wsks }}^{\nu} \xi$. Such a store is said to be a model of $\xi$. Note that, because we have assumed the successor functions.$i$ to be total, for all $i \in[1, \kappa]$, formulæ defining infinite sets, such as e.g. $\forall x . X(x) \rightarrow X(x .1)$, are unsatisfiable, under the interpretation of second-order variables as finite sets.

### 6.2. Rewriting Trees and Configurations

We begin by building a $\mathrm{WS} \kappa S$ formula that describes an infinite $\kappa$-ary tree, whose finite prefix is a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, for a $C L$ formula $\phi$. Let $\Delta=\left\{\rho_{1}, \ldots, \rho_{P}\right\}$ be a fixed SID in the following, such that $\rho_{1}$ is the rule that labels the root of $\mathcal{T}$, by convention. We use a designated tuple of second order variables $\mathbf{R}=\left\langle R_{1}, \ldots, R_{P}\right\rangle$, where each variable $R_{i}$ is interpreted as the set of tree nodes labeled with the rule $\rho_{i}$ in some rewriting tree, for all $i \in[1, P]$. With this convention, the RewrTree ${ }_{\Delta}(\mathbf{R})$ formula (Figure 6) defines rewriting trees:

- line (15) states that the sets that interpret the second-order variables $\mathbf{R}$ are pairwise disjoint and that $R_{1}$ is a singleton containing the root of the rewriting tree,
- line (16) states that the union of the sets $\mathbf{R}$ is prefix-closed, i.e. the parent of each node from the interpretation of a variable $R_{i}$ belongs to the interpretation of some variable $R_{j}$, for $i, j \in[1, P]$,
- lines (17) and (18) encode the conditions 1 and 2 of Def. 6 respectively, i.e. for that every predicate atom $\mathcal{A}^{\prime}\left(y_{1}, \ldots, y_{\#\left(\mathcal{A}^{\prime}\right)}\right)$ from a rule that labels a node $w$ in the rewriting tree there is exactly one child of that node, labeled with a rule $\mathcal{A}^{\prime}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}^{\prime}\right)}\right) \leftarrow \psi$ and $w$ has no other children.

It is not very hard to show that, for each model $\nu$ of $\operatorname{Rewr}_{\text {Tree }}^{\Delta} \boldsymbol{( R )}$ there exists a unique rewriting tree, denoted by $\mathcal{T}_{\nu}^{\mathbf{R}}$, such that $\operatorname{nodes}\left(\mathcal{T}_{\nu}^{\mathbf{R}}\right)=\bigcup_{i=1}^{P} \nu\left(R_{i}\right)$ and $\mathcal{T}_{\nu}^{\mathbf{R}}(w)=\rho_{i} \Longleftrightarrow w \in \nu\left(R_{i}\right)$, for all $i \in[1, P]$ and $w \in \operatorname{nodes}\left(\mathcal{T}_{\nu}^{\mathbf{R}}\right)$.

As discussed in $\$ 5$, a rewriting tree $\mathcal{T}$ defines a canonical configuration $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)$, where $\alpha_{\mathcal{T}}$ is the architecture consisting of the components and interactions corresponding to the atoms of $\mathfrak{F}(\mathcal{T})$ and $m_{\mathcal{T}}$ is a marking of the Petri

$$
\begin{align*}
& \operatorname{RewrTree}_{\Delta}(\mathbf{R}) \stackrel{\text { def }}{=} \forall x \underset{1 \leq i<j \leq P}{ }\left(\neg R_{i}(x) \vee \neg R_{j}(x)\right) \wedge\left(R_{1}(x) \leftrightarrow x=\epsilon\right) \wedge  \tag{15}\\
& \forall x \cdot \bigwedge_{i=1}^{P} \bigwedge_{\ell=1}^{\kappa} R_{i}(x . \ell) \rightarrow \bigvee_{j=1}^{P} R_{j}(x) \wedge  \tag{16}\\
& \forall x . \bigwedge_{i: \rho_{i}=\left(\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \varphi\right)} \bigwedge_{j=1}^{\# \text { pred }(\varphi)} R_{i}(x) \rightarrow \bigvee_{\substack{j: \operatorname{pred}_{j}(\varphi)=\mathcal{A}^{\prime}\left(y_{1}, \ldots, y_{\#\left(\mathcal{A}^{\prime}\right)}\right) \\
\ell: \rho_{\ell}=\left(\mathcal{A}^{\prime}\left(x_{1}, \ldots, x_{\#}\left(\mathcal{A}^{\prime}\right) \leftarrow \psi\right)\right.}} R_{\ell}(x . j) \wedge  \tag{17}\\
& \forall x . \bigwedge_{i: \rho_{i}=\left(\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \varphi\right)} \bigwedge_{j=\# \text { pred }(\varphi)+1}^{\kappa} R_{i}(x) \rightarrow \bigwedge_{\ell=1}^{P} \neg R_{\ell}(x . j) \tag{18}
\end{align*}
$$

Figure 6: Rewriting Trees
net $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$, for a given signature $\Sigma=\left(\left\{\mathrm{C}_{1}, \ldots, \mathrm{C}_{N}\right\},\left\{\mathrm{I}_{1}, \ldots, \mathrm{I}_{M}\right\}, \mathfrak{P}\right)$ and and recall that $\mathcal{Q}_{\beta}$ denotes the finite set of states used to describe the behaviors $\beta\left(\mathrm{C}_{1}\right), \ldots, \beta\left(\mathrm{C}_{N}\right)$.

Our next concern is defining the precise markings of $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$ (Def. 2) by a $W S \kappa S$ formula. We define sets of places of $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$ by means of a tuple of second-order variables $\mathbf{X} \xlongequal{\text { def }}\left\langle X_{q} \mid q \in \mathcal{Q}_{\beta}\right\rangle$. Any valuation $\nu$ of the variables $\mathbf{X}$ corresponds to a set of places $\sigma_{\nu}^{\text {x }} \stackrel{\text { def }}{=}\left\{q[u] \mid u \in \nu\left(X_{q}\right)\right\}$. When needed, we shall use distinct copies of $\mathbf{X}$, such as $\mathbf{X}^{\prime} \stackrel{\text { def }}{=}\left\langle X_{q}^{\prime} \mid q \in \mathcal{Q}_{\beta}\right\rangle, \mathbf{Y} \stackrel{\text { def }}{=}\left\langle Y_{q} \mid q \in \mathcal{Q}_{\beta}\right\rangle$ and $\mathbf{Z} \stackrel{\text { def }}{=}\left\langle Z_{q} \mid q \in \mathcal{Q}_{\beta}\right\rangle$. The $\mathrm{W} S_{\kappa} S$ formula below defines the precise markings of a Petri net $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$, assuming that $\mathcal{T}$ is the rewriting tree corresponding to some model of RewrTree $_{\Delta}(\mathbf{R})$ :

$$
\begin{align*}
\mathrm{m}(\mathbf{X}, \mathbf{R}) \stackrel{\text { def }}{=} \forall x \cdot & \bigwedge_{i=1}^{N}(  \tag{19}\\
& \left(\bigwedge_{q \neq q^{\prime} \in \text { states }_{\beta}\left(\mathrm{C}_{i}\right)}\left(\neg X_{q}(x) \vee \neg X_{q^{\prime}}(x)\right) \wedge\right.  \tag{20}\\
& \left.\left(\bigvee_{q \in \text { states }_{\beta}\left(\mathrm{C}_{i}\right)} X_{q}(x) \leftrightarrow \bigvee_{\rho_{j} \in \operatorname{inst}\left(\mathrm{C}_{i}\right)} R_{j}(x)\right)\right)
\end{align*}
$$

Intuitively, line above states that a component of type $C_{i}$ may not be in two different states from states ${ }_{\beta}\left(\mathrm{C}_{i}\right)$, and line 20 states that the index of a component of type $C_{i}$ must be a node of the rewriting tree labeled with a rule in which a component atom of the form $\mathrm{C}_{i}^{q}(x)$ occurs (see Assumption 2 on the syntax of the rules labeling a rewriting tree). We write inst(C) for the subset of $\Delta$ consisting of those rules that contain a component atom of the form $\mathrm{C}^{q}(x)$. The correctness of the encoding is stated and proved below:

Lemma 6. For any model $\nu$ of $\operatorname{RewrTree}_{\Delta}(\mathbf{R}) \wedge \mathrm{m}(\mathbf{X}, \mathbf{R})$, the set $\sigma_{\nu}^{\mathbf{x}}$ is a precise marking of the Petri net $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}^{\mathrm{R}}}, \beta\right)$.

Proof. Let $\mathrm{C} \in \mathfrak{C}$ be a component type and $u \in \alpha_{\mathcal{T}_{\nu}^{\mathbf{R}}}(\mathrm{C})$ be an index. We prove that $\| \sigma_{\nu}^{\mathrm{X}} \cap\left\{q[u] \mid q \in\right.$ states $\left._{\beta}(\mathrm{C})\right\} \|=1$, by proving the following points:

- $\sigma_{\nu}^{\mathbf{x}} \cap\left\{q[u] \mid q \in \operatorname{states}_{\beta}(\mathrm{C})\right\} \neq \emptyset:$ since $\models_{\text {wsks }}^{\nu} \operatorname{RewrTree}_{\Delta}(\mathbf{R})$, we have that $\mathcal{T}_{\nu}^{\mathbf{R}}$ is a rewriting tree, such that $\operatorname{nodes}\left(\mathcal{T}_{\nu}^{\mathbf{R}}\right)=\bigcup_{i=1}^{P} \nu\left(R_{i}\right)$ and $\nu\left(R_{i}\right)$ is the set of nodes of $\mathcal{T}_{\nu}^{\mathbf{R}}$ that are labeled with the rule $\rho_{i}$, for all $i \in[1, P]$. Since $u \in \alpha_{\mathcal{T}_{\nu}^{\mathbf{R}}}(\mathbf{C}) \subseteq \operatorname{nodes}\left(\mathcal{T}_{\nu}^{\mathbf{R}}\right)$, there exists a rule $\rho_{i}$, such that $\mathrm{C} \in \operatorname{inst}\left(\rho_{i}\right)$ and $u \in \nu\left(R_{i}\right)$, hence by line 20 , we have $u \in \nu\left(X_{q}\right)$, i.e. $q[u] \in \sigma_{\nu}^{\mathbf{x}}$, for some $q \in \operatorname{states}_{\beta}(\mathrm{C})$. Consequently, $q[u] \in \sigma_{\nu}^{\mathrm{x}} \cap\left\{q[u] \mid q \in \operatorname{states}_{\beta}(\mathrm{C})\right\}$.
- $\| \sigma_{\nu}^{\mathbf{X}} \cap\left\{q[u] \mid q \in\right.$ states $\left._{\beta}(\mathrm{C})\right\} \|<2$ : suppose that $q[u], q^{\prime}[u] \in \sigma_{\nu}^{\mathbf{x}}$, for $q \neq$ $q^{\prime} \in$ states $_{\beta}(\mathrm{C})$. Hence $u \in \nu\left(X_{q}\right) \cap \nu\left(X_{q^{\prime}}\right)$, which contradicts line 19 and the fact that $=_{\text {wsks }}^{\nu} m(\mathbf{X}, \mathbf{R})$.


### 6.3. Path Automata

The next step in building $\mathrm{W} S \kappa S$ verification conditions that are sufficient for proving deadlock freedom and unreachability of several control states at once (\$4) is tracking the variables that occur in a rule labeling a node of a rewriting tree $\mathcal{T}$, down to the component atoms which set their values, in the canonical architecture $\alpha_{\mathcal{T}}$. These values are used to define the interactions of $\alpha_{\mathcal{T}}$ and, in particular, to encode the pre-set $\left({ }^{\bullet} t\right)$ and post-set $\left(t^{\bullet}\right)$ of a transition $(t)$ from the Petri net $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$, by a flow formula, described next (6.4).

As briefly mentioned (see Example 8) a variable "traverses" the rewriting tree of a given formula downwards, while replacing the formal parameters of the rule labeling the child. This traversal is described by a special type of tree automaton, that walks down the tree, following the identity of the tracked variable. These automata are a restricted version of tree walking automata [16].

A path in a tree $\mathcal{T}$ is a sequence $n_{1}, \ldots, n_{\ell} \in \operatorname{nodes}(\mathcal{T})$, such that $n_{i+1}$ is a child of $n_{i}$, for all $i \in[1, \ell-1]$. If $m$ is a descendant of $n$, we denote by $\pi(n, m)$ the unique path starting in $n$ and ending in $m$. Note that $\pi(n, m)$ is determined by $n$ and the sequence of directions $d_{1}, \ldots, d_{p} \in[1, \kappa]$, such that $m=n \cdot d_{1} \cdot \ldots \cdot d_{p}$. A path automaton is a tuple $A=(S, I, F, \delta)$, where $S$ is a set of states, $I, F \subseteq S$ are the sets of initial and final states, respectively, and $\delta \subseteq S \times[1, \kappa] \times S$ is a set of transitions $s \xrightarrow{d} s^{\prime}$, where $s, s^{\prime} \in S$ are states and $d \in[1, \kappa]$ is a direction. A run of $A$ over the path $\pi(n, m)$, where $m=n \cdot d_{1} \cdot \ldots \cdot d_{p}$, is a sequence of states $s_{1}, \ldots, s_{p+1} \in S$, such that $s_{1} \in I$ and $s_{i} \xrightarrow{d_{i}} s_{i+1}$, for all $i \in[1, p]$. The run is accepting if and ony if $s_{p+1} \in F$. The language $\mathcal{L}(A)$ of $A$ is the set of paths in $\mathcal{T}$ over which $A$ has an accepting run.

A path automaton $A=\left(\left\{s_{1}, \ldots, s_{L}\right\}, I, F, \delta\right)$ corresponds, in the sense of Lemma 7 below, to the formula in Figure 7 which is, moreover, built directly from the syntactic description of $A$. Here $\mathbf{Y}=\left\langle Y_{1}, \ldots, Y_{L}\right\rangle$ are second order variables interpreted as the sets of tree nodes labeled by the automaton with the states $s_{1}, \ldots, s_{L}$, respectively. Intuitively, the first three conjuncts of the above formula (line 21) encode the facts that $\mathbf{Y}$ are disjoint (no tree node is labeled by

$$
\begin{align*}
& \Delta_{A}(x, y, \mathbf{Y}) \stackrel{\text { def }}{=} \bigwedge_{1 \leq i \neq j \leq L} \forall z \cdot\left(\neg Y_{i}(z) \vee \neg Y_{j}(z)\right) \wedge \bigvee_{s_{i} \in I} Y_{i}(x) \wedge \bigvee_{s_{j} \in F} Y_{j}(y)  \tag{21}\\
& \wedge \\
& \bigwedge_{i=1}^{L}\left(\forall z \cdot z \neq y \wedge Y_{i}(z) \rightarrow\right.  \tag{22}\\
&\left.\bigvee_{j}(z \cdot d) \vee \bigvee_{j: s_{i} \xrightarrow{d} s_{j}} \exists z^{\prime} \cdot z^{\prime} \cdot d=z \wedge Y_{j}\left(z^{\prime}\right)\right) \\
& \wedge \bigwedge_{j=1}^{L}\left(\forall z \cdot z \neq x \wedge Y_{j}(z) \rightarrow\right. \\
&\left.\forall z^{\prime} \cdot z^{\prime} \cdot d=z \wedge Y_{i}\left(z^{\prime}\right) \vee \bigvee_{i: s_{i} \xrightarrow{d} s_{j}} \bigvee_{i} Y_{i}(z \cdot d)\right) \tag{23}
\end{align*}
$$

Figure 7: Definition of the Path Automaton formula $\Delta_{A}(x, y, \mathbf{Y})$
more than one state during the run) and that the run starts in an initial state with node $x$ and ends in a final state with node $y$. The fourth conjunct (line 22) states that, for every non-final node on the path, if the automaton visits that node by state $s_{i}$, then the node has a $d$-child visited by state $s_{j}$, where $s_{i} \xrightarrow{d} s_{j}$ is a transition of the automaton. The fifth conjunct (line[23) is the reversed flow condition on the path, needed to ensure that the sets $\mathbf{Y}$ do not contain useless nodes, being thus symmetric to the fourth. The following result stems from the classical automata-logic connection (see [17, §2.10] for a textbook presentation):
Lemma 7. Given nodes $n, m \in[1, \kappa]^{*}$ and directions $d_{1}, \ldots, d_{p} \in[1, \kappa]$ such that $m=n \cdot d_{1} \cdot \ldots \cdot d_{p}$, for each valuation $\nu$, such that $\nu(x)=n$ and $\nu(y)=m$, we have $d_{1} \ldots d_{p} \in \mathcal{L}(A) \Longleftrightarrow \models_{\text {wsts }}^{\nu} \exists \mathbf{Y} . \Delta_{A}(x, y, \mathbf{Y})$.

Our purpose is to infer, directly from the syntax of the rules in $\Delta$, path automata that recognize the path between the node where a variable is introduced (either as a formal parameter of a rule or by an existential quantifier) and the node where the variable is instantiated, in each given rewriting tree. Formally, for each rule $\rho=\left(\mathcal{A}\left(x_{1}, \ldots, x_{\# \mathcal{A}}\right) \leftarrow \exists y_{1} \ldots \exists y_{m} \cdot \psi\right)$, such that $\psi$ is a quantifie-free formula, and each variable $x \in \operatorname{fv}(\psi)$, we consider the path automaton $A_{\rho}^{x}=(S, I, F, \delta)$, where:

- $S \stackrel{\text { def }}{=}\left\{s_{\rho^{\prime}}^{z} \mid \rho^{\prime}=\left(\mathcal{A}^{\prime}\left(x_{1}, \ldots, x_{\# \mathcal{A}^{\prime}}\right) \leftarrow \exists y_{1} \ldots \exists y_{m} \cdot \phi\right), \phi\right.$ is q.f., $\left.z \in \operatorname{fv}(\phi)\right\}$; the intuition is that the automaton visits the state $s_{\rho^{\prime}}^{z}$ while tracking variable $z$ in the body of the rule $\rho^{\prime}$, that labels the node of the rewriting tree which is currently visited by the automaton,
- the initial and final states are $I \stackrel{\text { def }}{=}\left\{s_{\rho}^{x}\right\}$ and $F \stackrel{\text { def }}{=}\left\{s_{\rho^{\prime}}^{z} \mid \rho^{\prime}=\left(\mathcal{A}^{\prime}\left(x_{1}, \ldots, x_{\# \mathcal{A}^{\prime}}\right)\right.\right.$ $\left.\leftarrow \exists y_{1} \ldots \exists y_{m} \cdot \mathrm{C}^{q}(z) * \phi\right), \phi$ is q.f., $\mathrm{C} \in \mathfrak{C}, q \in$ states $\left._{\beta}(\mathrm{C})\right\}$; the automaton starts to track $x$ in $\rho$ and moves down in the rewriting tree, finally tracking a variable $z$ that occurs in a component atom in $\rho^{\prime}$,


Figure 8: Path Automata Recognizing the Paths to Component Atoms from Example 6

- the transitions are $s_{\rho}^{y_{j}} \xrightarrow{d} s_{\rho^{\prime}}^{x_{j}}$, for all rules $\rho=\left(\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \varphi\right)$ and $\rho^{\prime}=\left(\mathcal{A}^{\prime}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}^{\prime}\right)}\right) \leftarrow \varphi^{\prime}\right)$ from $\Delta$, all directions $d \in[1, \#$ pred $(\varphi)]$, such that $\operatorname{pred}_{d}(\varphi)=\mathcal{A}^{\prime}\left(y_{1}, \ldots, y_{\#\left(\mathcal{A}^{\prime}\right)}\right)$, and all $j \in\left[1, \#\left(\mathcal{A}^{\prime}\right)\right]$; if $\rho$ labels the parent of the node labeled by $\rho^{\prime}$ in the rewriting tree, the automaton moves down one step, from tracking $y_{j}$ in $\rho$ to tracking $x_{j}$ in $\rho^{\prime}$.

Example 9. The path automata recognizing the paths to component atoms, for the variables $\ell_{1}^{\epsilon}$ and $r_{1}^{\epsilon}$ in the rewriting tree from Example 6 are depicted in Figure 8. The initial states are $9 \frac{\ell_{1}}{7}$ and $9 \frac{r_{1}}{7}$, respectively, and the final state is 910 for both automata. The rule labels are the ones from Example 6 .

The lemma below shows that these automata recognize the set of paths corresponding to the recursive substitutions of a given variable down to the node where it occurs in a component atom, in all rewriting trees built using the rules from the given SID:

Lemma 8. Given a rule $\rho=\left(\mathcal{A}\left(x_{1}, \ldots, x_{\#(\mathcal{A})}\right) \leftarrow \exists y_{1} \ldots \exists y_{m} . \phi\right)$, such that $\phi$ is a quantifier-free formula, and a variable $x \in \operatorname{fv}(\phi)$, for each sequence $w \in$ $[1, \kappa]^{*}$, the following are equivalent:

1. there exists a rewriting tree $\mathcal{T}$ and nodes $n, m \in \operatorname{nodes}(\mathcal{T})$, such that $\mathcal{T}(n)=\rho, m=n \cdot w$ and $\mathfrak{s}_{\mathcal{T}}(x)=m\left(\right.$ see 11 for the definition of $\left.\mathfrak{s}_{\mathcal{T}}\right)$.
2. $w \in \mathcal{L}\left(A_{\rho}^{x}\right)$.

Proof. (11) $\Rightarrow \sqrt{2 p}$ By the definition of $\mathfrak{s} \mathcal{T}$, we have $\mathfrak{s}_{\mathcal{T}}(x)=\mathfrak{s}_{\mathcal{T}_{n}}^{\epsilon}(x)$ and $\mathfrak{s}_{\mathcal{T}_{n}}^{\epsilon}(x)=$ $m$ if there exist:

- a sequence of nodes $n=n_{1}, n_{2}, \ldots, n_{p}=m \in \operatorname{nodes}(\mathcal{T})$, such that $n_{i+1}=$ $n_{i} \cdot d_{i}$, for some $d_{i} \in[1, \kappa]$, for all $i \in[1, p-1]$,
- a sequence of variables $x=z_{1}, \ldots, z_{p} \in \mathbb{X}^{1}$, such that $z_{i}$ replaces $z_{i+1}$ in the construction of the characteristic formula $\mathfrak{F}(\mathcal{T})$, for all $i \in[1, p-1]$, and occurs in a component atom from $\mathcal{T}(m)$.
By the definition of rewriting trees, each node $n_{i}$ is labeled by a rule $\rho_{i}=$ $\left(\mathcal{A}_{i}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)}\right) \leftarrow \exists y_{1} \ldots \exists y_{m} \cdot \phi_{i}\right)$, where $\phi_{i}$ is quantifier-free, and there exists $j \in\left[1, \#\left(\mathcal{A}_{i}\right)\right]$ such that $z_{i}$ occurs on the $j$-th position in $\operatorname{pred}_{d_{i}}\left(\phi_{i}\right)$ and $z_{i+1}$ occurs $j$-th within the head $\mathcal{A}_{i+1}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{i+1}\right)}\right)$ of $\rho_{i+1}$. By the definition of $A_{\rho}^{x}$, there exist transitions $s_{\rho_{i}}^{z_{i}} \xrightarrow{d_{i}} s_{\rho_{i+1}}^{z_{i+1}}$, for all $i \in[1, p-1]$. Moreover,
$s_{\rho}^{x}=s_{\rho_{1}}^{z_{1}}$ and $s_{\rho_{p}}^{z_{p}}$ are initial state and final states of $A_{\rho}^{x}$, respectively. Since $m=n \cdot d_{1} \ldots d_{p}=n \cdot w$, we obtain that $w=d_{1} \ldots d_{p} \in \mathcal{L}\left(A_{\rho}^{x}\right)$.
(2) $\Rightarrow$ 11) Let $w=d_{1} \ldots d_{p}$, where $d_{1}, \ldots, d_{p} \in[1, \kappa]$, and $s_{\rho_{i}}^{z_{i}} \xrightarrow{d_{i}} s_{\rho_{i+1}}^{z_{i+1}}$, for all $i \in[1, p-1]$, be transitions of $A_{\rho}^{x}$, such that $s_{\rho_{1}}^{z_{1}}=s_{\rho}^{x}$ is the initial state and $s_{\rho_{p}}^{z_{p}}$ is a final state of $A_{\rho}^{x}$. Then, by the definition of $A_{\rho}^{x}$, the nodes $w_{i} \stackrel{\text { def }}{=} n \cdot d_{1} \ldots d_{i}$ are labeled by rules $\rho_{i} \stackrel{\text { def }}{=}\left(\mathcal{A}_{i}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{i}\right)}\right) \leftarrow \varphi_{i}\right)$, such that $z_{i}$ occurs in $\operatorname{pred}_{d_{i}}\left(\varphi_{i}\right)$ on the same position as $z_{i+1}$, which occurs in $\mathcal{A}_{i+1}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}_{i+1}\right)}\right)$. $m=n \cdot d_{1} \ldots d_{o}$ and $A_{\rho}^{x}$ has an accepting run over $d_{1} \ldots d_{p}$. By the definition of $\mathfrak{s}_{\mathcal{T}}$, we have $\mathfrak{s}_{\mathcal{T}}(x)=\mathfrak{s}_{\mathcal{I}_{n}}^{\epsilon}(x)=m$.

The path automata $A_{\rho}^{z}$ are used to define the $\operatorname{Path}_{\rho}^{z}(x, y, \mathbf{R})$ formulæ below:

$$
\begin{aligned}
\operatorname{Path}_{\rho}^{z}(x, y, \mathbf{R}) & \stackrel{\text { def }}{=} \exists \mathbf{Y} \cdot \Delta_{A_{\rho}^{z}}(x, y, \mathbf{Y}) \wedge \Upsilon(\mathbf{Y}, \mathbf{R}) \\
\Upsilon(\mathbf{Y}, \mathbf{R}) & \stackrel{\text { def }}{=} \bigwedge_{i: \rho_{i}=\left(\mathcal{A}^{\prime}\left(x_{1}, \ldots, x_{\#\left(\mathcal{A}^{\prime}\right)}\right) \leftarrow \varphi^{\prime}\right)} \bigwedge_{z \in \mathrm{fv}\left(\varphi^{\prime}\right)} \forall x \cdot Y_{\rho_{i}}^{z}(x) \rightarrow R_{i}(x)
\end{aligned}
$$

The formula $\Upsilon(\mathbf{Y}, \mathbf{R})$ above states that all nodes labeled with a state $q_{\rho_{i}}^{z}$ during the run must be also labeled with $\rho_{i}$ in the rewriting tree given as input to the path automaton. Here we denote by $Y_{\rho_{i}}^{z}$ the second-order variable corresponding to the state $q_{\rho_{i}}^{z}$ in a path automaton $A_{\rho}^{x}$. Intuitively, the formula $\operatorname{Path}_{\rho}^{z}(x, y, \mathbf{R})$ states that, in each rewriting tree defined by a model $\nu$ of the RewrTree ${ }_{\Delta}(\mathbf{R})$ formula, there exists a path from $\nu(x)$ to $\nu(y)$, where $\nu(x)$ is a node labeled with $\rho$, in which $z$ occurs free or existentially quantified, and $y$ is the node where a variable necessarily equal to $z$ occurs in a component atom.

### 6.4. Flow Formulae

At this point, we introduce a $W S \kappa S$ formula $\Phi\left(\mathbf{X}, \mathbf{X}^{\prime}, \mathbf{R}\right)$ that defines the structure of transitions in the Petri net $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu} \mathbf{R}}, \beta\right)$, where $\nu$ is a model of RewrTree $_{\Delta}(\mathbf{R})$. By "structure" here we mean that $\nu(\mathbf{X})$ and $\nu\left(\mathbf{X}^{\prime}\right)$ encode the pre- and post-sets of some transition in $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathbf{R}}}, \beta\right)$, as explained in $\$ 6.2$

Figure 9 shows the flow formula of a given SID, consisting of the rules $\rho_{1}, \ldots, \rho_{P}$. We denote by inter $(\rho)$ the set of interaction atoms that occur in the body of a rule $\rho$. Essentially, the formula 24 is split into a disjunction of formulæ $\Psi_{\ell, \mathrm{l}\left(x_{1}, \ldots, x_{\#(I)}\right)}$, one for each rule $\rho_{\ell} \in \Delta$ and each interaction atom $\mathrm{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)$ that occurs in the body of $\rho_{\ell}$. To understand the $\left.\Psi_{\ell, \mathrm{I}\left(x_{1}, \ldots, x_{\#}(\mathrm{I})\right.}\right)$ formulæ, recall that each of the variables $x_{i}$ is mapped to the (unique) node of the rewriting tree containing an instance atom $\mathrm{C}^{q}(x)$, such that $x_{i}$ replaces $x$ in the characterizing formula of the rewriting tree $\mathcal{T}_{\nu}^{\mathbf{R}}$, provided that $\nu$ is a model of $\operatorname{Rewr}^{\operatorname{Tree}} \Delta \Delta(\mathbf{R})$. In order to find this node, we track the variable $x_{i}$ from the node labeled by the rule $\rho$, to the node where this instance atom occurs. This is done by the $\operatorname{Path}_{\rho_{\ell}}^{x_{i}}\left(y_{0}, y_{i}, \mathbf{R}\right)$ formula $\sqrt{25}$, that holds whenever $y_{0}$ and $y_{i}$ are respectively mapped to the source and the destination of a path from a node $\nu\left(y_{0}\right) \in \operatorname{nodes}\left(\mathcal{T}_{\nu}^{\mathbf{R}}\right)$, with label $\rho_{\ell}$ to a node $\nu\left(y_{i}\right) \in \operatorname{nodes}\left(\mathcal{T}_{\nu}^{\mathbf{R}}\right)$, such that the

$$
\begin{gather*}
\Phi\left(\mathbf{X}, \mathbf{X}^{\prime}, \mathbf{R}\right) \stackrel{\text { def }}{=} \bigvee_{\ell=1 \mathrm{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right) \in \operatorname{inter}\left(\rho_{\ell}\right)}^{P} \Psi_{\ell\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)}\left(\mathbf{X}, \mathbf{X}^{\prime}, \mathbf{R}\right)  \tag{24}\\
\Psi_{\ell, \mathrm{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)}\left(\mathbf{X}, \mathbf{X}^{\prime}, \mathbf{R}\right) \stackrel{\text { def }}{=} \exists y_{0} \ldots \exists y_{\#(\mathrm{I})} \cdot R_{\ell}\left(y_{0}\right) \wedge \bigwedge_{i=1}^{\#(\mathrm{I})} \operatorname{Path}_{\rho_{\ell}}^{x_{i}}\left(y_{0}, y_{i}, \mathbf{R}\right) \wedge  \tag{25}\\
\left(\bigvee_{\left\langle\tau_{1}, \ldots, \tau_{\#(\mathrm{I})}\right\rangle \in T_{\beta}(\mathrm{I})} \bigwedge_{q \in \mathcal{Q}_{\beta}} X_{q}=\left\{y_{i} \mid i \in[1, \#(\mathrm{I})], \tau_{i}=\left(q_{i} \xrightarrow{p_{i}} q_{i}^{\prime}\right), q_{i}=q\right\} \wedge\right.  \tag{26}\\
\left.X_{q}^{\prime}=\left\{y_{i} \mid i \in[1, \#(\mathrm{I})], \tau_{i}=\left(q_{i} \xrightarrow{p_{i}} q_{i}^{\prime}\right), q_{i}^{\prime}=q\right\}\right) \wedge  \tag{27}\\
\bigwedge_{1 \leq i<j \leq \#(\mathrm{I})}\left(y_{i} \neq y_{j} \vee \operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{i}\right)=\operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{j}\right)\right) \tag{28}
\end{gather*}
$$

Figure 9: The Flow Formula
component atom that gives the value of $x_{i}{ }^{5}$ occurs in $\mathcal{T}_{\nu}^{\mathbf{R}}\left(\nu\left(y_{i}\right)\right)$. Hence, $x_{i}$ is interpreted as $\nu\left(y_{i}\right)$ in the canonical model of the characteristic formula of $\mathcal{T}_{\nu}^{\mathbf{R}}$.

The correctness of the encoding is formalized and proved by the following:
Lemma 9. Given a model $\nu$ of RewrTree $_{\Delta}(\mathbf{R})$, we have $\models_{\text {wsks }}^{\nu} \Phi\left(\mathbf{X}, \mathbf{X}^{\prime}, \mathbf{R}\right)$ if and only if $\sigma_{\nu}^{\mathbf{X}}={ }^{\bullet} t$ and $\sigma_{\nu}^{\mathbf{x}^{\prime}}=t^{\bullet}$, for some transition $t$ of $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}}, \beta\right)$.

Proof. " $\Rightarrow$ " If $\models_{\text {wsks }}^{\nu} \Phi\left(\mathbf{X}, \mathbf{X}^{\prime}, \mathbf{R}\right)$ then there exists a rule $\rho_{\ell} \in \Delta=\left\{\rho_{1}, \ldots, \rho_{N}\right\}$ and $\mathbf{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right) \in \operatorname{inter}\left(\rho_{\ell}\right)$, such that $\left.\right|_{\text {wsks }} ^{\nu} \Psi_{\ell, \mathrm{I}\left(x_{1}, \ldots, x_{\#(I)}\right)}\left(\mathbf{X}, \mathbf{X}^{\prime}, \mathbf{R}\right)$. Then, there exist nodes $w_{0}, \ldots, w_{n} \in \operatorname{nodes}\left(\mathcal{T}_{\nu}^{\mathbf{R}}\right)$, such that $\nu\left(y_{i}\right)=w_{i}$, for all $i \in[0, n]$ and let $\nu^{\prime}=\nu\left[y_{0} \leftarrow w_{0}, \ldots, y_{n} \leftarrow w_{n}\right]$. Since, moreover, $\models_{\text {wsks }}^{\nu^{\prime}} R_{\ell}\left(y_{0}\right)$, we obtain that $\mathcal{T}_{\nu}^{\mathbf{R}}\left(w_{0}\right)=\rho_{\ell}$, hence the variables $x_{1}, \ldots, x_{\#(\mathrm{I})}$ occur in the body of the rule $\rho_{\ell}$ and are assigned to the unique nodes $w_{1}^{\prime}, \ldots, w_{\#(\mathrm{I})}^{\prime} \in \operatorname{nodes}\left(\mathcal{T}_{\nu}^{\mathbf{R}}\right)$, for which $\pi\left(w_{0}, w_{i}^{\prime}\right) \in \mathcal{L}\left(A_{\rho_{\rho}}^{x_{i}}\right)$, for all $i \in[1, \#(\mathrm{I})]$, by Lemma 8 . Since $\models_{\text {wsks }}^{\nu^{\prime}}$ Path $_{\rho_{\ell}}^{x_{i}}\left(y_{0}, y_{i}, \mathbf{R}\right)$, by Lemma 7 , we obtain $\pi\left(w_{0}, w_{i}\right) \in \mathcal{L}\left(A_{\rho_{\ell}}^{x_{i}}\right)$, thus $w_{i}=w_{i}^{\prime}$, for all $i \in[1, \#(\mathrm{I})]$, by the uniqueness of these nodes. Let $\left\langle\tau_{1}, \ldots, \tau_{\#(\mathrm{I})}\right\rangle$ be a tuple of transitions from $T_{\beta}(\mathbf{I})$ synchronizing by I. Since $\models_{\text {wsks }}^{\nu^{\prime}} \Psi_{\ell, \mathrm{I}\left(x_{1}, \ldots, x_{\#(I)}\right)}\left(\mathbf{X}, \mathbf{X}^{\prime}, \mathbf{R}\right)$, we have $w_{i} \neq w_{j}$, for each $1 \leq i<j \leq \#(\mathrm{I})$, such that $\operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{i}\right)=$ ${ }_{1105} \operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{j}\right)$. By Def. [1] we have that $t=\left(\mathrm{I}\left[w_{1}, \ldots, w_{\#(\mathrm{I})}\right],\left\langle\tau_{1}, \ldots, \tau_{\#(\mathrm{I})}\right\rangle\right)$ is a transition of $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathrm{R}}}, \beta\right)$ and, moreover, $\bullet t=\left\{q_{1}, \ldots, q_{\#(\mathrm{I})}\right\}=\sigma_{\nu}^{\mathrm{x}}$ and $t^{\bullet}=$ $\left\{q_{1}^{\prime}, \ldots, q_{\#(\mathrm{I})}^{\prime}\right\}=\sigma_{\nu}^{\mathrm{X}^{\prime}}$, because $\models_{\text {wsks }}^{\nu^{\prime}} X_{q}=\left\{y_{i} \mid i \in[1, \#(\mathrm{I})], \tau_{i}=\left(q_{i} \xrightarrow{p_{i}} q_{i}^{\prime}\right), q_{i}=q\right\}$ and $\models_{\text {wsks }}^{\nu^{\prime}} X_{q}^{\prime}=\left\{y_{i} \mid i \in[1, \#(\mathrm{I})], \tau_{i}=\left(q_{i} \xrightarrow{p_{i}} q_{i}^{\prime}\right), q_{i}^{\prime}=q\right\}$, respectively.
$" \Leftarrow "$ Let $t=\left(\mathrm{I}\left[w_{1}, \ldots, w_{\#(\mathrm{I})}\right],\left\langle\tau_{1}, \ldots, \tau_{\#(\mathrm{I})}\right\rangle\right)$ be a transition of $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathrm{R}}}, \beta\right)$, where $\tau_{i}=q_{i} \xrightarrow{\langle\mathfrak{P}(\mathrm{I})\rangle_{i}} q_{i}^{\prime}$ is a transition of $\beta\left(\operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{i}\right)\right)$, for each $i \in$

[^4]$[1, \#(\mathrm{I})]$. Then there exists a node $w_{0} \in \operatorname{nodes}\left(\mathcal{T}_{\nu}^{\mathbf{R}}\right)$ and an interaction atom $\mathrm{I}\left(x_{1}, \ldots, x_{\#(\mathrm{I})}\right)$ in $\mathcal{T}_{\nu}^{\mathbf{R}}\left(w_{0}\right)$, such that $\mathfrak{s}_{\mathcal{T}_{\nu}^{\mathbf{R}}}\left(x_{i}\right)=w_{i}$, for all $i \in[1, \#(\mathbf{I})]$. Assume w.l.o.g. that $\mathcal{T}_{\nu}^{\mathbf{R}}\left(w_{0}\right)=\rho_{\ell}$, for some $\ell \in[1, P]$ and let $\nu^{\prime}=\nu\left[y_{0} \leftarrow\right.$ $\left.w_{0}, \ldots, y_{\#(\mathrm{I})} \leftarrow w_{\#(\mathrm{I})}\right]$ be a valuation. We prove the following points:

- $=_{\text {wsks }}^{\nu^{\prime}} R_{\ell}\left(y_{0}\right)$ : since $=_{\text {wsks }}^{\nu} \operatorname{RewrTree}_{\Delta}(\mathbf{R}), \mathcal{T}_{\nu}^{\mathbf{R}}\left(w_{0}\right)=\rho_{\ell}, \nu^{\prime}\left(y_{0}\right)=w_{0}$ and $\nu^{\prime}$ agrees with $\nu$ over $\mathbf{R}$.
- $=_{\text {wsks }}^{\nu^{\prime}} \operatorname{Path}_{\rho_{\ell}}^{x_{i}}\left(y_{0}, y_{i}, \mathbf{R}\right)$, for all $i \in[1, \#(\mathrm{I})]$ : since $\mathfrak{s}_{\mathcal{T}_{\nu}^{\mathrm{R}}}\left(x_{i}\right)=w_{i}$, we obtain $\pi\left(w_{0}, w_{i}\right) \in \mathcal{L}\left(A_{\rho_{\ell}}^{x_{i}}\right)$ by Lemma 8 and $\models_{\text {wsks }}^{\nu^{\prime}} \operatorname{Path}_{\rho_{\ell}}^{x_{i}}\left(y_{0}, y_{i}, \mathbf{R}\right)$ follows, by Lemma 7
- $\mid=_{\text {wsks }}^{\nu^{\prime}} y_{i} \neq y_{j}$, for all $1 \leq i<j \leq \#(\mathrm{I})$, such that $\operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{i}\right)=$ $\operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{j}\right)$ : by Def. 1, $w_{i} \neq w_{j}$ for all $i, j \in[1, \#(\mathrm{I})]$, such that $\operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{i}\right)=\operatorname{comp}\left(\langle\mathfrak{P}(\mathrm{I})\rangle_{j}\right)$.
- $=_{\text {wsks }}^{\nu^{\prime}} X_{q}=\left\{y_{i} \mid i \in[1, \#(\mathrm{I})], \tau_{i}=\left(q_{i} \xrightarrow{p_{i}} q_{i}^{\prime}\right), q_{i}=q\right\}$, for all $q \in \mathcal{Q}_{\beta}$ : because $\sigma_{\nu}^{\mathrm{x}}={ }^{\bullet} t=\left\{q_{1}, \ldots, q_{\#(\mathrm{I})}\right\}$.
- $=_{\text {wsks }}^{\nu^{\prime}} X_{q}=\left\{y_{i} \mid i \in[1, \#(\mathrm{I})], \tau_{i}=\left(q_{i} \xrightarrow{p_{i}} q_{i}^{\prime}\right), q_{i}^{\prime}=q\right\}$, for all $q \in \mathcal{Q}_{\beta}$ : because $\sigma_{\nu}^{\mathbf{x}^{\prime}}=t^{\bullet}=\left\{q_{1}^{\prime}, \ldots, q_{\#(\mathrm{I})}^{\prime}\right\}$
We obtain that $\models_{\text {wsks }}^{\nu} \Psi_{\ell, 1\left(x_{1}, \ldots, x_{\#(I)}\right)}$ and, consequently $\models_{\text {wsks }}^{\nu} \Phi\left(\mathbf{X}, \mathbf{X}^{\prime}, \mathbf{R}\right)$.


### 6.5. Structural Invariants

An invariant $\mathcal{I}$ of a marked Petri net $\mathcal{N}=\left(\mathrm{N}, \mathrm{m}_{0}\right)$, where $\mathrm{N}=(S, T, E)$ is a Petri net, is a set of markings that over-approximates the reachable states $\operatorname{Reach}(\mathcal{N})$, that is $\operatorname{Reach}(\mathcal{N}) \subseteq \mathcal{I} \subseteq \operatorname{pow}(S)$. The invariant is, moreover, said to be inductive if and only if it is closed under the transition relation of the Petri net, namely that $\left\{\mathrm{m}^{\prime} \mid \mathrm{m} \in \mathcal{I}, t \in T, \mathrm{~m} \xrightarrow{t} \mathrm{~m}^{\prime}\right\} \subseteq \mathcal{I}$.

The synthesis of inductive invariants is a notoriously difficult problem, that has been received much attention in the past. The most common method of infering inductive invariants is the iteration of an abstract (over-approximation) transition relation in an abstract domain, until a fixpoint is reached. Depending on the complexity of the abstract domain, this approach, known as abstract interpretation [9, can be quite costly. In contrast, we consider inductive invariants that can be synthesized directly from the structure of a Petri net, without iterating (an abstraction of) its transition relation up to a fixpoint. Such invariants are called structural in the literature [18. The next definition introduces two types of structural invariants:

Definition 11. Given a marked Petri net $\mathcal{N}=\left(\mathrm{N}, \mathrm{m}_{0}\right)$, where $\mathrm{N}=(S, T, E)$, a set of places $\sigma \subseteq S$ is said to be a:

- trap if $\left\|\sigma \cap \mathrm{m}_{0}\right\| \geq 1$ and, for any $t \in T$, if $\left\|\sigma \cap{ }^{\bullet} t\right\| \geq 1$ then $\left\|\sigma \cap t^{\bullet}\right\| \geq 1$,
- mutex if $\left\|\sigma \cap \mathrm{m}_{0}\right\|=1$ and, for any $t \in T$, we have $\left\|\sigma \cap{ }^{\bullet} t\right\|=\left\|\sigma \cap t^{\bullet}\right\| \leq 1$.

The trap and mutex invariant of $\mathcal{N}$ are the below sets of markings, respectively:

- $\Theta(\mathcal{N}) \stackrel{\text { def }}{=}\{\mathrm{m}$ marking of $\mathcal{N} \mid\|\mathrm{m} \cap \sigma\| \geq 1$, for each trap $\sigma$ of $\mathcal{N}\}$,
- $\Omega(\mathcal{N}) \stackrel{\text { def }}{=}\{\mathrm{m}$ marking of $\mathcal{N} \mid\|\mathrm{m} \cap \sigma\|=1$, for each mutex $\sigma$ of $\mathcal{N}\}$.

By substituting the reachable set with the intersection of the trap and mutex invariant, we obtain the following sufficient queries, that allow to prove the absence of deadlocks and unreachability of a tuple of states in a system described by a CL formula:

Lemma 10. Given a tight sentence $\phi$, interpreted over a SID $\Delta$, a behavior map $\beta$ and a tuple of states $\left\langle q_{1}, \ldots, q_{n}\right\rangle$, the following hold:

$$
\begin{aligned}
\operatorname{deadlock}(\phi, \Delta, \beta) & \Rightarrow \operatorname{deadlock}_{t}^{\sharp}(\phi, \Delta, \beta) \\
\operatorname{reach}\left(\phi,\left\langle q_{1}, \ldots, q_{n}\right\rangle, \Delta, \beta\right) & \Rightarrow \operatorname{reach}_{t}^{\sharp}\left(\phi,\left\langle q_{1}, \ldots, q_{n}\right\rangle, \Delta, \beta\right)
\end{aligned}
$$

where deadlock ${ }_{t}^{\sharp}(\phi, \Delta, \beta)$ and $\operatorname{reach}_{t}^{\sharp}\left(\phi,\left\langle q_{1}, \ldots, q_{n}\right\rangle, \Delta, \beta\right)$ are defined below:

- deadlock $k_{t}^{\sharp}(\phi, \Delta, \beta)$ : does there exist a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, such that $\Theta\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right) \cap \Omega\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right) \cap \operatorname{Dead}\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)\right) \neq \emptyset$ ?
- $\operatorname{reach}_{t}^{\sharp}\left(\phi,\left\langle q_{1}, \ldots, q_{n}\right\rangle, \Delta, \beta\right)$ : does there exist a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$ and a configuration $\left(\alpha_{\mathcal{T}}, \mathrm{m}\right) \in \Theta\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right) \cap \Omega\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right)$, such that $\left\{q_{i}\left[u_{i}\right] \mid i \in[1, n]\right\} \subseteq \mathrm{m}$, with $q_{1}\left[u_{1}\right], \ldots, q_{n}\left[u_{n}\right]$ pairwise distinct?

Proof. We show first that the trap $\Theta(\mathcal{N})$ and mutex $\Omega(\mathcal{N})$ invariants of a marked Petri net $\mathcal{N}$ are invariants of $\mathcal{N}$. If $m_{0}=\emptyset$ then $\mathcal{N}$ has no traps nor mutexes and Reach $(\mathcal{N})=\Theta(\mathcal{N})=\Omega(\mathcal{N})=\emptyset$, thus $\Theta(\mathcal{N})$ and $\Omega(\mathcal{N})$ are trivially inductive invariants. So we assume $\mathrm{m}_{0} \neq \emptyset$.

For the set $\Theta(\mathcal{N})$, let $\sigma$ be an arbitrary trap of $\mathcal{N}$. Since $m_{0} \neq \emptyset$, we have $\left\|\sigma \cap \mathrm{m}_{0}\right\| \geq 1$, hence $\mathrm{m}_{0} \in \Theta(\mathcal{N})$, by the choice of $\sigma$. For each pair of markings m and $\mathrm{m}^{\prime}$, such that $\mathrm{m} \in \Theta(\mathcal{N})$ and $\mathrm{m} \xrightarrow{t} \mathrm{~m}^{\prime}$, for some transition $t \in T$, we have $\mathrm{m}^{\prime}=(\mathrm{m} \backslash \bullet t) \cup t^{\bullet}$. If $\sigma \cap \bullet t=\emptyset$ then $\left\|\mathrm{m}^{\prime} \cap \sigma\right\| \geq\|\mathrm{m} \cap \sigma\|$ and $\|\mathrm{m} \cap \sigma\| \geq 1$, because $\mathrm{m} \in \Theta(\mathcal{N})$. Otherwise, $\|\sigma \cap \bullet t\| \geq 1$, thus $\left\|\sigma \cap t^{\bullet}\right\| \geq 1$ because $\sigma$ is a trap of $\mathcal{N}$, hence $\left\|\mathrm{m}^{\prime} \cap \sigma\right\| \geq 1$. In both cases, we obtain that $\left\|\mathrm{m}^{\prime} \cap \sigma\right\| \geq 1$, hence $\mathrm{m}^{\prime} \in \Theta(\mathcal{N})$, because $\sigma$ is an arbitrary trap of $\mathcal{N}$.

A similar argument shows that $\Omega(\mathcal{N})$ is an inductive invariant of $\mathcal{N}$. Let $\sigma$ be an arbitrary mutex of $\mathcal{N}$. Then $\left\|\sigma \cap \mathrm{m}_{0}\right\|=1$, hence $\mathrm{m}_{0} \in \Omega(\mathcal{N})$, by the choice of $\sigma$. For each pair of markings m and $\mathrm{m}^{\prime}$, such that $\mathrm{m} \in \Theta(\mathcal{N})$ and $\mathrm{m} \xrightarrow{t} \mathrm{~m}^{\prime}$, for some transition $t \in T$, we have $\mathrm{m}^{\prime}=(\mathrm{m} \backslash \bullet t) \cup t^{\bullet}$ and $\|\sigma \cap \bullet t\|=\left\|\sigma \cap t^{\bullet}\right\| \leq 1$, hence $\left\|\mathrm{m}^{\prime} \cap \sigma\right\|=1$ and $\mathrm{m}^{\prime} \in \Omega(\mathcal{N})$, because $\sigma$ is an arbitrary mutex of $\mathcal{N}$. Moreover, since inductive invariants are closed under intersection, the set $\Theta(\mathcal{N}) \cap \Omega(\mathcal{N})$ is also an inductive invariant of $\mathcal{N}$. Finally, because $\operatorname{Reach}(\mathcal{N})$ is known to be the least inductive invariant of $\mathcal{N}$, we obtain Reach $(\mathcal{N}) \subseteq \Theta(\mathcal{N}) \cap \Omega(\mathcal{N})$. The rest of the proof follows from Theorem 2 and the above fact.

Given a CL sentence $\phi$ that describes the set of initial configurations of a system, Figure 10 introduces $\mathrm{WS} \kappa S$ formulæ that define the trap and mutex invariants of the marked Petri nets $\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{0}\right)$, where $\mathcal{T} \in \mathbb{T}(\phi)$ are rewriting trees and $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{0}\right) \models \phi$ are initial configurations. This construction uses the flow formula (Figure 9) derived from the SID that interprets the predicate symbols from $\phi$ and the behavior map $\beta$ of the system. Here we denote by $\mathbf{X}$ (resp.

$$
\begin{align*}
& \operatorname{Init}(\mathbf{Y}, \mathbf{R}) \stackrel{\text { def }}{=} \mathrm{m}(\mathbf{Y}, \mathbf{R}) \wedge \forall x \cdot \bigwedge_{q \in \mathcal{Q}_{\beta}}\left(\begin{array}{c}
Y_{q}(x) \leftrightarrow \\
\begin{array}{l}
\ell \in[1, P] \text { such that } \\
\mathrm{C}^{q}(z) \text { occurs in } \rho_{\ell}
\end{array} \\
\hline \theta(\mathbf{X}, \mathbf{R}) \stackrel{\text { def }}{=} \forall \mathbf{Y} \forall \mathbf{Z} \cdot \Phi(\mathbf{Y}, \mathbf{Z}, \mathbf{R}) \rightarrow(\|\mathbf{X} \cap \mathbf{Y}\| \geq 1 \rightarrow\|\mathbf{X} \cap \mathbf{Z}\| \geq 1)
\end{array}\right.  \tag{29}\\
& \begin{aligned}
\Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R}) \stackrel{\text { def }}{=} \mathrm{m}(\mathbf{X}, \mathbf{R}) \wedge \forall \mathbf{Z} \cdot(\|\mathbf{Y} \cap \mathbf{Z}\| \geq 1 \wedge \theta(\mathbf{Z}, \mathbf{R})) \rightarrow\|\mathbf{X} \cap \mathbf{Z}\| \geq 1
\end{aligned}  \tag{30}\\
& \omega(\mathbf{X}, \mathbf{R}) \stackrel{\text { def }}{=} \forall \mathbf{Y} \forall \mathbf{Z} \cdot \Phi(\mathbf{Y}, \mathbf{Z}, \mathbf{R}) \rightarrow((\|\mathbf{X} \cap \mathbf{Y}\|=0 \leftrightarrow\|\mathbf{X} \cap \mathbf{Z}\|=0) \wedge  \tag{31}\\
& \\
& \quad(\|\mathbf{X} \cap \mathbf{Y}\|=1 \leftrightarrow\|\mathbf{X} \cap \mathbf{Z}\|=1))  \tag{32}\\
& \Omega(\mathbf{X}, \mathbf{Y}, \mathbf{R}) \stackrel{\text { def }}{=} \mathrm{m}(\mathbf{X}, \mathbf{R}) \wedge \forall \mathbf{Z} .(\|\mathbf{Y} \cap \mathbf{Z}\|=1 \wedge \omega(\mathbf{Y}, \mathbf{R})) \rightarrow\|\mathbf{X} \cap \mathbf{Z}\|=1 \tag{33}
\end{align*}
$$

Figure 10: Initial Configurations and Structural Invariants
$\mathbf{Y}$ and $\mathbf{Z}$ ) the tuple of second-order variables $\left\langle X_{q} \mid q \in \mathcal{Q}_{\beta}\right\rangle$ (resp. $\left\langle Y_{q} \mid q \in \mathcal{Q}_{\beta}\right\rangle$ and $\left\langle Z_{q} \mid q \in \mathcal{Q}_{\beta}\right\rangle$ ) and we write $\mathbf{X} \cap \mathbf{Y}$ (resp. $\mathbf{X} \cap \mathbf{Z}$ ) for $\bigcup_{q \in \mathcal{Q}_{\beta}} X_{q} \cap Y_{q}$ (resp. $\left.\bigcup_{q \in \mathcal{Q}_{\beta}} X_{q} \cap Z_{q}\right)$, where the definitions of set union and intersection are standard in MSO and omitted to avoid clutter.

Let $\nu$ be a model of $\operatorname{RewrTree}_{\Delta}(\mathbf{R})$, meaning that $\mathcal{T}_{\nu}^{\mathbf{R}}$ is a rewriting tree for $\phi$. We recall that, by letting $\rho_{1} \stackrel{\text { def }}{=}\left(\mathcal{A}_{\phi}() \leftarrow \phi\right)$ be the first rule in the SID, we capture the fact that $\mathcal{T}_{\nu}^{\mathbf{R}} \in \mathbb{T}(\phi)$, for each model $\nu$ of $\operatorname{Rewr}^{\operatorname{Tree}}{ }_{\Delta}(\mathbf{R})$ (see the second conjunct in line 15 from Figure 6). The formula $\operatorname{Init}(\mathbf{Y}, \mathbf{R})(29)$ encodes the fact that $\sigma_{\nu}^{\mathrm{Y}}$ is an initial marking of the Petri net $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathrm{R}}}, \beta\right)$ namely, that each node of the rewriting tree $\mathcal{T}_{\nu}^{\mathbf{R}}$ is the index of a component of type C in state $q$, if $\mathrm{C}^{q}(z)$ occurs in the rule that labels the node.

We assume further that $\nu$ is a model of $\operatorname{RewrTree}_{\Delta}(\mathbf{R}) \wedge \operatorname{Init}(\mathbf{Y}, \mathbf{R})$. The formulæ $\theta(\mathbf{X}, \mathbf{R})$ 30) and $\Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R})$ (31) define the traps and the trap invariant of the marked Petri net $\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathbf{R}}}, \beta\right), \sigma_{\nu}^{\mathbf{Y}}\right)$, respectively. Similarly, $\omega(\mathbf{X}, \mathbf{R}) 32$ and $\Omega(\mathbf{X}, \mathbf{R})$ 33) define the mutexes and the mutex invariant of $\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathrm{R}}}, \beta\right), \sigma_{\nu}^{\mathbf{Y}}\right)$, respectively. We intentionally use the same symbols to denote trap (mutex) invariants and their defining $\mathrm{W} S \kappa S$ formulæ, the distinction between sets of markings and formulæ being clear from the context. The following lemma shows the correctness of the definitions from Figure 10 .
Lemma 11. If $\models_{\text {wsks }}^{\nu}$ RewrTree $_{\Delta}(\mathbf{R}) \wedge \operatorname{Init}(\mathbf{Y}, \mathbf{R})$ then the following hold:

1. $\Theta\left(\mathbf{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathbf{R}}}, \beta\right), \sigma_{\nu}^{\mathbf{Y}}\right)=\left\{\sigma_{\nu^{\prime}}^{\mathbf{X}} \mid \models_{\text {wsks }}^{\nu^{\prime}} \Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R}), \nu^{\prime}(\mathbf{Y}, \mathbf{R})=\nu(\mathbf{Y}, \mathbf{R})\right\}$,
2. $\Omega\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathbf{R}}}, \beta\right), \sigma_{\nu}^{\mathbf{Y}}\right)=\left\{\sigma_{\nu^{\prime}}^{\mathbf{X}} \mid \models_{\text {wsks }}^{\nu^{\prime}} \Omega(\mathbf{X}, \mathbf{Y}, \mathbf{R}), \nu^{\prime}(\mathbf{Y}, \mathbf{R})=\nu(\mathbf{Y}, \mathbf{R})\right\}$.

Proof. We give the proof for point (1), the proof for (2) uses a similar argument and is left for the reader.
$" \subseteq "$ Let $\mathrm{m} \in \Theta\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathrm{R}}}, \beta\right), \sigma_{\nu}^{\mathrm{Y}}\right)$ be a marking and $\nu^{\prime}$ be a valuation such that $\sigma_{\nu^{\prime}}^{\mathbf{x}}=\mathrm{m}$ and $\nu^{\prime}$ agrees with $\nu$ over $\mathbf{Y}$ and $\mathbf{R}$. Clearly, such a valuation exists
and, in order to show that $\mathrm{m} \in\left\{\sigma_{\nu^{\prime}}^{\mathrm{X}} \mid \models_{\text {wsks }}^{\nu^{\prime}} \Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R}), \nu^{\prime}(\mathbf{Y}, \mathbf{R})=\nu(\mathbf{Y}, \mathbf{R})\right\}$, it suffices to prove $\models_{\text {wsks }}^{\nu^{\prime}} \Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R})$. Because $\sigma_{\nu^{\prime}}^{\mathbf{X}}=\mathrm{m}$, we have $\models_{\text {wsks }}^{\nu^{\prime}} \mathrm{m}(\mathbf{X}, \mathbf{R})$ and we are left with proving $\models_{\text {wsks }}^{\nu^{\prime}} \forall \mathbf{Z} . \theta(\mathbf{Z}, \mathbf{R}) \wedge\|\mathbf{Y} \cap \mathbf{Z}\| \geq 1 \rightarrow\|\mathbf{X} \cap \mathbf{Z}\| \geq 1$. To this end, let $\mu$ be any valuation that agrees with $\nu^{\prime}$ over $\mathbf{X}, \mathbf{Y}$ and $\mathbf{R}$, such that $\models_{\text {wsks }}^{\mu} \theta(\mathbf{Z}, \mathbf{R}) \wedge\|\mathbf{Y} \cap \mathbf{Z}\| \geq 1$. It suffices to prove that $\sigma_{\mu}^{\mathbf{Z}}$ is a trap of the marked Petri net $\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathbf{R}}}, \beta\right), \sigma_{\nu}^{\mathbf{Y}}\right)$ to obtain $=_{\text {wsks }}^{\mu}\|\mathbf{X} \cap \mathbf{Z}\| \geq 1$, using the fact that $\mathrm{m} \in \Theta\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathrm{R}}}, \beta\right), \sigma_{\nu}^{\mathbf{Y}}\right)$ and $\sigma_{\mu}^{\mathrm{X}}=\mathrm{m}$. Note that, since $\mu$ agrees with $\nu^{\prime}$ over $\mathbf{X}$, we also have $\sigma_{\mu}^{\mathbf{X}}=\sigma_{\nu^{\prime}}^{\mathbf{X}}=\mathrm{m}$. Since $\models_{\text {wsks }}^{\mu}\|\mathbf{Y} \cap \mathbf{Z}\| \geq 1$ and $\mu$ agrees with $\nu$ over $\mathbf{Y}$, we have $\sigma_{\mu}^{\mathbf{Z}} \cap \sigma_{\nu}^{\mathbf{Y}} \neq \emptyset$. Let $t$ be a transition of $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathrm{R}}}, \beta\right)$, such that $\sigma_{\mu}^{\mathbf{Z}} \cap \bullet t \neq \emptyset$ and let $\mu^{\prime}$ be a valuation that agrees with $\mu$ over $\mathbf{Z}$, such that $\sigma_{\mu^{\prime}}^{\mathbf{Y}^{\prime}}=\bullet t$ and $\sigma_{\mu}^{\mathbf{Z}^{\prime}}=t^{\bullet}$, where $\mathbf{Y}^{\prime}$ and $\mathbf{Z}^{\prime}$ are distinct copies of $\mathbf{Y}$ and $\mathbf{Z}$, respectively. By Lemma9, we obtain $\models_{\text {wsks }}^{\mu^{\prime}} \Phi\left(\mathbf{Y}^{\prime}, \mathbf{Z}^{\prime}, \mathbf{R}\right)$. Since $\models_{\text {wsks }}^{\mu} \theta(\mathbf{Z}, \mathbf{R})$ and $\mu^{\prime}$ agrees with $\mu$ over $\mathbf{Z}$, we obtain $\models_{\text {wsks }}^{\mu^{\prime}}\left\|\mathbf{Z} \cap \mathbf{Z}^{\prime}\right\| \geq 1$, thus $\sigma_{\mu}^{\mathbf{Z}} \cap \sigma_{\mu^{\prime}}^{\mathbf{Z}^{\prime}}=\sigma_{\mu}^{\mathbf{Z}} \cap t^{\bullet} \neq \emptyset$. We have showed that $\sigma_{\mu}^{\mathbf{Z}}$ is a trap of the marked Petri net $\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu} \mathrm{R}}, \beta\right), \sigma_{\nu}^{\mathbf{Y}}\right)$, which concludes this direction of the proof.
" $\supseteq$ " Let $\nu^{\prime}$ be a valuation that agrees with $\nu$ over $\mathbf{Y}$ and $\mathbf{R}$, such that $~==_{\text {wsks }}^{\nu^{\prime}}$ $\Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R})$. To show that $\sigma_{\nu^{\prime}}^{\mathbf{X}} \in \Theta\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu} \mathbf{R}}, \beta\right), \sigma_{\nu}^{\mathbf{Y}}\right)$, we must prove that $\sigma_{\nu^{\prime}}^{\mathbf{X}} \cap \theta \neq \emptyset$, for each trap $\theta$ of the marked Petri net $\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathbf{R}}}, \beta\right), \sigma_{\nu}^{\mathbf{Y}}\right)$. Let $\theta$ be such a trap and let $\mu$ be a valuation that agrees with $\nu^{\prime}$ over $\mathbf{X}, \mathbf{Y}$ and $\mathbf{R}$, such that $\sigma_{\mu}^{Z}=\theta$. Since $\sigma_{\mu}^{Z}$ is a trap of the marked Petri net $\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}^{\mathbf{R}}}, \beta\right), \sigma_{\nu}^{\mathbf{Y}}\right)$, we obtain:

- $\models_{\text {wsks }}^{\mu}\|\mathbf{Y} \cap \mathbf{Z}\| \geq 1$, since $\models_{\text {wsks }}^{\nu} \operatorname{Init}(\mathbf{Y}, \mathbf{R})$ and $\mu$ agrees with $\nu$ over $\mathbf{Y}$, and
$\bullet \models_{\text {wsks }}^{\mu} \theta(\mathbf{X}, \mathbf{R})$, by Lemma 9, since $\mu$ agrees with $\nu$ over $\mathbf{X}$ and $\mathbf{R}$.
Since $\models_{\text {wsks }}^{\mu} \Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R})$, we obtain that $\models_{\text {wsks }}^{\mu}\|\mathbf{X} \cap \mathbf{Z}\| \geq 1$, thus $\sigma_{\mu}^{\mathbf{x}} \cap \sigma_{\mu}^{\mathbf{z}}=\sigma_{\nu^{\prime}}^{\mathbf{x}} \cap$ $\theta \neq \emptyset$, which concludes this direction of the proof.


### 6.6. Verification Conditions

The final step in generating sufficient verification conditions for the deadlock and reachability problems (\$4) is the encoding of the sets of error configurations in $\mathrm{WS} \kappa S$. This is done separately, for the two kinds of errors considered, by the formulæ in Figure 11. The convention here is to use $\mathbf{X}$ to represent the configurations in the error sets. However, none of the formulæ from Figure 11 constrains $\mathbf{X}$ to be a marking, i.e. by conjunction with $\mathrm{m}(\mathbf{X})$, since the latter formula is already included in the definition of the structural invariants, by the formulæ $\Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R})$ and $\Omega(\mathbf{X}, \mathbf{Y}, \mathbf{R})$ (Figure 10).

We recall that a marking m is a deadlock of a Petri net $\mathrm{N}=(S, T, E)$ if and only if $\bullet t \nsubseteq \mathrm{~m}$, for all transitions $t \in T$ (if $\bullet t \subseteq \mathrm{~m}$ for some transition $t$, then that transition could be fired from m ). This condition is captured by the $\operatorname{DeadLock}(\mathbf{X}, \mathbf{R})$ formula (34), where $\Phi(\mathbf{Y}, \mathbf{Z}, \mathbf{R})$ is the flow formula that defines the pre- and post-sets, for some transition in the Petri net $\mathbf{N}\left(\Sigma, \alpha_{\mathcal{T}_{\nu}} \mathrm{R}, \beta\right)$, where $\nu$ is a model of $\operatorname{RewrTree}_{\Delta}(\mathbf{R})$ (Lemma 9). Here we write $\mathbf{Y} \subseteq \mathbf{X}$ as a shorthand for $\forall x . \bigwedge_{q \in \mathcal{Q}_{\beta}} Y_{q}(x) \rightarrow X_{q}(x)$, denoting the fact that the set of places defined by $\mathbf{Y}$ is included in the one defined by $\mathbf{X}$.

$$
\begin{gather*}
\operatorname{DeadLock}(\mathbf{X}, \mathbf{R}) \stackrel{\text { def }}{=} \forall \mathbf{Y} \forall \mathbf{Z} . \Phi(\mathbf{Y}, \mathbf{Z}, \mathbf{R}) \rightarrow \neg \mathbf{Y} \subseteq \mathbf{X}  \tag{34}\\
\operatorname{ErrSet}_{\left\langle q_{1}, \ldots, q_{n}\right\rangle}(\mathbf{X}) \stackrel{\text { def }}{ } \bigwedge_{q \in \mathcal{Q}_{\beta}}\left\|X_{q}\right\| \geq\left\|\left\{i \in[1, k] \mid q_{i}=q\right\}\right\| \tag{35}
\end{gather*}
$$

Figure 11: Error Configurations for the Deadlock and Reachability Problems

The set defined by the $\operatorname{ErrSet}_{\left\langle q_{1}, \ldots, q_{n}\right\rangle}(\mathbf{X})$ formula 35 captures the fact that each configuration encoded by $\mathbf{X}$ contains pairwise distinct places $q_{1}\left[w_{1}\right], \ldots$, $q_{n}\left[w_{n}\right]$ which is, essentially, the condition required by a reachability query. Note that, because the tuple of states $\left\langle q_{1}, \ldots, q_{n}\right\rangle$ is part of the input of the query, the number $\left\|\left\{i \in[1, k] \mid q_{i}=q\right\}\right\|$ of occurrences of each state $q$ in the tuple is constant, hence the cardinality of each set $X_{q}$ is compared to a constant (13).

The following theorem states the soundness of the verification conditions built throughout this section:

Theorem 3. Given a tight sentence $\phi$, interpreted over a SID $\Delta$, a behavior map $\beta$ and a tuple of states $\left\langle q_{1}, \ldots, q_{n}\right\rangle$, the following hold:

1. $\operatorname{DeadLock}(\mathbf{X}, \mathbf{R}) \wedge \Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R}) \wedge \Omega(\mathbf{X}, \mathbf{Y}, \mathbf{R}) \wedge \operatorname{Init}(\mathbf{Y}, \mathbf{R}) \wedge \operatorname{RewrTree}_{\Delta}(\mathbf{R})$ is unsatisfiable only if deadlock $(\phi, \Delta, \beta)$ has a negative answer.
2. $\operatorname{ErrSet}_{\left\langle q_{1}, \ldots q_{n}\right\rangle}(\mathbf{X}) \wedge \Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R}) \wedge \Omega(\mathbf{X}, \mathbf{Y}, \mathbf{R}) \wedge \operatorname{Init}(\mathbf{Y}, \mathbf{R}) \wedge \operatorname{RewrTree}_{\Delta}(\mathbf{R})$ is unsatisfiable only if $\operatorname{reach}\left(\phi,\left\langle q_{1}, \ldots, q_{n}\right\rangle, \Delta, \beta\right)$ has a negative answer.

Proof. (1) To prove the contrapositive statement, assume that deadlock $(\phi, \Delta, \beta)$ has a positive answer. By Lemma 10 , the query deadlock ${ }_{t}^{\sharp}(\phi, \Delta, \beta)$ has a positive answer as well, thus there exists a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$ and a marking $\mathrm{m} \in$ $\Theta\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right) \cap \Omega\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right) \cap \operatorname{Dead}\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)\right)$, where $\left(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}\right)$ is the canonical model of $\phi$ corresponding to $\mathcal{T}$. Let $\nu$ be a valuation such that $\mathcal{T}_{\nu}^{\mathbf{R}}=\mathcal{T}, \sigma_{\nu}^{\mathbf{Y}}=\mathrm{m}_{\mathcal{T}}$ and $\sigma_{\nu}^{\mathrm{x}}=\mathrm{m}$. Since the tuples of second-order variables $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{T}$ are pairwise disjoint, such a valuation exists. We prove the following points:

- $=_{\text {wsks }}^{\nu} \operatorname{Init}(\mathbf{Y}, \mathbf{R})$ follows directly from the definitions of $\mathrm{m}_{\mathcal{T}}$ (Def. 7) and $\operatorname{Init}(\mathbf{Y}, \mathbf{R}) \sqrt{29}$.
- $=_{\text {wsks }}^{\nu} \Theta(\mathbf{X}, \mathbf{Y}, \mathbf{R})$ follows from point 1 of Lemma 11, because $\sigma_{\nu}^{\mathbf{x}}=\mathrm{m}$, $\mathcal{T}_{\nu}^{\mathbf{R}}=\mathcal{T}$ and $\mathrm{m} \in \Theta\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right)$.
- $=_{\text {wsks }}^{\nu} \Omega(\mathbf{X}, \mathbf{Y}, \mathbf{R})$ follows from point 2 of Lemma 11 , because $\sigma_{\nu}^{\mathbf{X}}=\mathrm{m}$, $\mathcal{T}_{\nu}^{\text {wsks }}=\mathcal{T}$ and $\mathrm{m} \in \Omega\left(\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right), \mathrm{m}_{\mathcal{T}}\right)$.
 $\sigma_{\nu^{\prime}}^{\mathbf{x}}$ of $\mathbf{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$, such that $=_{\text {wsks }}^{\nu^{\prime}} \operatorname{DeadLock}(\mathbf{X}, \mathbf{R}) \wedge$ RewrTree $_{\Delta}(\mathbf{R})$. This follows directly from the definition of a deadlock marking and the definition of $\operatorname{DeadLock}(\mathbf{X}, \mathbf{R})(34$, using the characterization of the pre- and post-sets of the transitions from $\mathrm{N}\left(\Sigma, \alpha_{\mathcal{T}}, \beta\right)$ provided by Lemma 9 .
(2) The proof of this point follows a similar argument as for point 1, the only difference being that the set of markings containing a set $\left\{q_{i}\left[u_{i}\right] \mid i \in[1, n]\right\} \subseteq$
m , with $q_{1}\left[u_{1}\right], \ldots, q_{n}\left[u_{n}\right]$ pairwise distinct coincides with the set $\left\{\sigma_{\nu^{\prime}}^{\mathbf{X}} \mid \models_{\text {wsks }}^{\nu^{\prime}}\right.$ $\operatorname{ErrSet}_{\left\langle q_{1}, \ldots, q_{n}\right\rangle}(\mathbf{X}) \wedge \mathrm{m}(\mathbf{X}, \mathbf{R}) \wedge$ RewrTree $\left._{\Delta}(\mathbf{R})\right\}$.

Since the WS $\kappa S$ logic is decidable with non-elementary complexity, in general [12], the problem of checking the satisfiability of the verification conditions provided by Theorem 3 is decidable. By inspection of the formulæ (1) and (2), one can show that checking the verification conditions is in 4EXPTIME, where 4 is the maximum quantifier alternation depth of these formulæ. In practice, however, these checks are quite fast, as shown by the preliminary experiments performed on a similar encoding, reported in our previous work [2].

## 7. Related Work

This paper reports on a resource logic for the specification of sets of configurations of parametric distributed systems with unbounded numbers of components (processes) and a verification method for safety properties, such as absence of deadlocks and critical section violations.

Traditionally, verification of unbounded networks of parallel processes considers known architectural patterns, typically cliques or rings [19, 20. Because the price for decidability is drastic restriction on architecture styles [21], more recent works propose practical semi-algorithms, e.g. regular model checking [22, 23] or automata learning [24. Here the architectural pattern is implicitly determined by the class of language recognizers: word automata encode pipelines or rings, whereas tree automata describe trees.

A first attempt at specifying architectures by logic is the interaction logic of Konnov et al. [25], which is a combination of Presburger arithmetic with monadic uninterpreted function symbols, that can describe cliques, stars and rings. More structured architectures (pipelines and trees) can be described using a second-order extension [26. As such, these interaction logics are undecidable and have no support for automated verification. Recently, interaction logics that support the verification of safety properties, by structural invariant synthesis have been developed. These logics use fragments of first order logic with interpreted function symbols that implicitly determine the class of architectures, such as cliques [27], or pipelines, rings and trees [28].

From a theoretical point of view, our resource logic with inductive definitions is strictly more expressive: for instance, a chain of components where a certain component type occurs on all even (odd) positions is provably not
${ }_{1325}$ expressible in first order logic, but can be easily defined using our language, whose inductive definitions allow to describe second order constructs in a controlled manner (rather than using unrestricted second order quantification, as in [26]). Moreover, first order logic with successor functions can describe at most tree-like architectures, whereas our language describes structures more general than trees ${ }^{6}$, using no interpreted function symbols, other than ports involved in interactions [27, 28].

[^5]Specifying parameterized component-based systems by inductive definitions is, however, not new. Network grammars [29] use context-free grammar rules to describe systems with linear (pipeline, token-ring) architectures obtained by composition of an unbounded number of processes. In contrast, we use predicate symbols of unrestricted arities to describe architectural patterns that are, in general, more complex than trees. Such complex structures can be specified recursively using graph grammar rules with parameter variables [30, 31]. To avoid clashes, these variables must be renamed to unique names and assigned ${ }^{1340}$ unique indices at each unfolding step. Our specifications use quantifiers to avoid name clashes and separating conjunction to guarantee that the components and interactions obtained by the unfolding of the rules are unique.

Verification of network grammars against safety properties requires the synthesis of network invariants [32], computed by rather costly fixpoint iterations [33] or by abstracting (forgetting the particular values of indices in) the composition of a small bounded number of instances 34. Instead, our method uses lightweight structural invariants, that are synthesized with little computational effort and prove to be efficient in many practical examples [28, 2].

## 8. Conclusions and Future Work

The paper proposes a general framework for the practical semi-algorithmic verification of parametric systems. The framework integrates previous work of the authors and extends it through developments in two complementary directions. The first direction is modeling parametric system architectures from instances of predefined component types and interaction types. Configuration logic allows the specification of configurations characterizing architecture styles with snapshots of their component states. Parametric system behavior can be obtained, in the form of Petri nets using operational semantics, from given configuration logic specifications and behaviors of component types. We show that even for very simple linear architectures, essential safety properties such as deadlock-freedom or mutual explosion are undecidable.

The second direction proposes a method that from given configuration logic specifications and the finite-state behavior of its components, leads to formulas of $\mathrm{WS} \kappa S$ characterizing two types of structural invariants used to prove deadlock-freedom and mutual exclusion properties. The generation process avoids the complexity of traditional fixpoint computation techniques. It synthesizes constraints on the state space induced by the interactions of the parametric architecture. Verification boils down to checking the satisfiability of $\mathrm{W} \delta \kappa S$ formulæ, for which optimized solvers exist. The whole generation process involves transformations from configuration specifications into rewriting trees and then into WS $\kappa S$ through path automata. Proving its soundness requires some tedious technical developments; nonetheless, its implementation does not suffer from the usual limitations due to state space complexity and the number of components. Experimental results show that the proposed verification method is scalable and allows proving safety in a number of non-trivial cases [2].

As future work, we plan on adding support for broadcast in our specification language and develop further the invariant synthesis method to take broadcast into account. We also envisage an extension of the finite-state model of behavior to more complex automata, such as pushdown or timed automata.

## References

[1] J. Kramer, J. Magee, Analysing dynamic change in distributed software architectures, IEE Proceedings - Software 145 (5) (1998) 146-154.
[2] M. Bozga, R. Iosif, Specification and safety verification of parametric hierarchical distributed systems, in: 17th International Conference, FACS 2021, Virtual Event, October 28-29, 2021, Proceedings, Vol. 13077 of Lecture Notes in Computer Science, Springer, 2021, pp. 95-114.
[3] J. C. Reynolds, Separation logic: A logic for shared mutable data structures, in: 17th IEEE Symposium on Logic in Computer Science (LICS 2002), IEEE Computer Society, 2002, pp. 55-74.
[4] E. Ahrens, M. Bozga, R. Iosif, J. Katoen, Local reasoning about parameterized reconfigurable distributed systems, CoRR abs/2107.05253 (2021). URL https://arxiv.org/abs/2107.05253
[5] P. W. O'Hearn, Resources, concurrency, and local reasoning, Theor. Comput. Sci. 375 (1-3) (2007) 271-307.
[6] I. Sergey, A. Nanevski, A. Banerjee, G. A. Delbianco, Hoare-style specifications as correctness conditions for non-linearizable concurrent objects, SIGPLAN Not. 51 (10) (2016) 92-110.
[7] F. Farka, A. Nanevski, A. Banerjee, G. A. Delbianco, I. Fábregas, On algebraic abstractions for concurrent separation logics, Proc. ACM Program. Lang. 5 (POPL) (2021) 1-32.
[8] C. W. Barrett, I. Shikanian, C. Tinelli, An abstract decision procedure for a theory of inductive data types, J. Satisf. Boolean Model. Comput. 3 (1-2) (2007) 21-46.
[9] P. Cousot, R. Cousot, Systematic design of program analysis frameworks, in: 6th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM Press, New York, NY, 1979, pp. 269-282.
[10] E. L. Post, Finite Combinatory Processes-Formulation 1, The Journal of Symbolic Logic 1 (3) (1936) 103-105.
[11] M. Davis, What is a computation?, in: Mathematics Today Twelve Informal Essays, Conference Board of the Mathematical Sciences, 1978, pp. 241-267.
[12] J. Thatcher, J. Wright, Generalized finite automata theory with an application to a decision problem of second-order logic, Mathematical systems theory 2 (2005) 57-81.
[13] E. Emerson, A. Sistla, Symmetry and model checking, Formal Methods in System Design 9 (1996) 105-131. doi:10.1007/BF00625970.
[14] M. Bozga, L. Bueri, R. Iosif, Decision problems in a logic for reasoning about reconfigurable distributed systems, in: IJCAR, Vol. 13385 of Lecture Notes in Computer Science, Springer, 2022, pp. 691-711.
[15] J. G. Henriksen, J. L. Jensen, M. E. Jørgensen, N. Klarlund, R. Paige, T. Rauhe, A. Sandholm, Mona: Monadic second-order logic in practice, in: First International Workshop, TACAS '95, Vol. 1019 of LNCS, Springer, 1995, pp. 89-110.
[16] M. Bojańczyk, Tree-walking automata, in: Language and Automata Theory and Applications: Second International Conference, LATA 2008, Tarragona, Spain, March 13-19, 2008. Revised Papers, Springer-Verlag, Berlin, Heidelberg, 2008, p. 1-2.
[17] B. Khoussainov, A. Nerode, Automata Theory and its Applications, Springer, 2001.
[18] J. Sifakis, Structural properties of Petri nets, in: Mathematical Foundations of Computer Science 1978, Proceedings, 7th Symposium, Zakopane, Poland, September 4-8, 1978, Vol. 64 of Lecture Notes in Computer Science, Springer, 1978, pp. 474-483.
[19] S. M. German, A. P. Sistla, Reasoning about systems with many processes, J. ACM 39 (3) (1992) 675-735.
[20] M. Browne, E. Clarke, O. Grumberg, Reasoning about networks with many identical finite state processes, Information and Computation 81 (1) (1989) $13-31$.
[21] R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, J. Widder, Decidability of Parameterized Verification, Synthesis Lectures on Distributed Computing Theory, Morgan \& Claypool Publishers, 2015.
[22] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, E. Shahar, Symbolic model checking with rich assertional languages, Theoretical Computer Science 256 (1) (2001) 93-112.
[23] P. A. Abdulla, G. Delzanno, N. B. Henda, A. Rezine, Regular model checking without transducers (on efficient verification of parameterized systems), in: 13th International Conference, TACAS 2007, Vol. 4424 of LNCS, Springer, 2007, pp. 721-736.
[24] Y. Chen, C. Hong, A. W. Lin, P. Rümmer, Learning to prove safety over parameterised concurrent systems, in: 2017 Formal Methods in Computer Aided Design, FMCAD 2017, IEEE, 2017, pp. 76-83.
[25] I. V. Konnov, T. Kotek, Q. Wang, H. Veith, S. Bliudze, J. Sifakis, Parameterized systems in BIP: design and model checking, in: 27th International Conference on Concurrency Theory, CONCUR 2016, Vol. 59 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 30:1-30:16.
[26] A. Mavridou, E. Baranov, S. Bliudze, J. Sifakis, Configuration logics: Modeling architecture styles, J. Log. Algebr. Meth. Program. 86 (1) (2017) 2-29.
[27] M. Bozga, R. Iosif, J. Sifakis, Checking deadlock-freedom of parametric component-based systems, in: 25th International Conference, TACAS 2019, Vol. 11428 of LNCS, Springer, 2019, pp. 3-20.
[28] M. Bozga, J. Esparza, R. Iosif, J. Sifakis, C. Welzel, Structural invariants for the verification of systems with parameterized architectures, in: 26th International Conference, TACAS 2020, Vol. 12078 of LNCS, Springer, 2020, pp. 228-246.
[29] Z. Shtadler, O. Grumberg, Network grammars, communication behaviors and automatic verification, in: Automatic Verification Methods for Finite State Systems, International Workshop, Vol. 407 of LNCS, Springer, 1989, pp. 151-165.
[30] D. Le Metayer, Describing software architecture styles using graph grammars, IEEE Transactions on Software Engineering 24 (7) (1998) 521-533.
[31] D. Hirsch, P. Inverardi, U. Montanari, Graph grammars and constraint solving for software architecture styles, in: Proceedings of the Third International Workshop on Software Architecture, ISAW '98, Association for Computing Machinery, New York, NY, USA, 1998, p. 69-72.
[32] P. Wolper, V. Lovinfosse, Verifying properties of large sets of processes with network invariants, in: Automatic Verification Methods for Finite State Systems, International Workshop, Vol. 407 of LNCS, Springer, 1989, pp. 68-80.
[33] D. Lesens, N. Halbwachs, P. Raymond, Automatic verification of parameterized linear networks of processes, in: The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, ACM Press, 1997, pp. 346-357.
[34] Y. Kesten, A. Pnueli, E. Shahar, L. D. Zuck, Network invariants in action, in: CONCUR 2002 - Concurrency Theory, 13th International Conference, Vol. 2421 of LNCS, Springer, 2002, pp. 101-115.


[^0]:    * Corresponding author

    Email addresses: marius.bozga@univ-grenoble-alpes.fr (Marius Bozga), radu.iosif@univ-grenoble-alpes.fr (Radu Iosif), joseph.sifakis@univ-grenoble-alpes.fr (Joseph Sifakis)
    ${ }^{1}$ Institute of Engineering Univ. Grenoble Alpes

[^1]:    ${ }^{2}$ Boolean Petri nets are sometimes called 1-safe or 1-bounded in the literature.

[^2]:    ${ }^{3} w$ can be augmented with extra zeroes on the right

[^3]:    ${ }^{4}$ Typically, global permutations suffice when only one component type is considered.

[^4]:    ${ }^{5}$ In the canonical store $\mathfrak{s}_{\mathcal{T}_{\nu}} \mathbf{R}$, see $\S 5.2$ for its definition.

[^5]:    ${ }^{6}$ For instance, the tree-shaped architecture with leaves linked in a ring $\$ 5$

