Verification of component-based systems with recursive architectures

Marius Bozga, Radu Iosif, Joseph Sifakis

Verification of Component-based Systems with Recursive Architectures

Marius Bozga*, Radu Iosif, Joseph Sifakis

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 $\mathsf{WS}\kappa S$ 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 $\mathsf{WS}\kappa S$ 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 describing the behavior of the components (e.g. semaphores, monitors, compare-and-swap, 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

*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

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 finite-state 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 between (unboundedly large) sets of components. Such high-level models of real-life 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 ($\mathtt{t}$) or not ($\mathtt{n}$). The number of tokens is constant and equal to the number of stations that are initially in state $\mathtt{t}$. Each station not having a token may receive a token from the left, via the *in* port (that triggers the transition $\mathtt{n} \xrightarrow{in} \mathtt{t}$) and send its token to the right, via the *out* port (that triggers the transition $\mathtt{t} \xrightarrow{out} \mathtt{n}$). We refer to Figure 1(a) for a depiction of a token-ring system, defined by the rules:

$$\mathit{Ring}() \leftarrow \exists y_1 \exists y_2 \ . \ \mathit{Chain}(y_1, y_2) * \mathsf{T}(y_2, y_1) \tag{1}$$

$$\mathit{Chain}(x_1, x_2) \leftarrow \mathit{Comp}(x_1) * \mathsf{T}(x_1, x_2) * \mathit{Comp}(x_2) \tag{2}$$

$$\mathit{Chain}(x_1, x_2) \leftarrow \exists y_1 \ . \ \mathit{Comp}(x_1) * \mathsf{T}(x_1, y_1) * \mathit{Chain}(y_1, x_2) \tag{3}$$

$$\mathit{Comp}(x_1) \leftarrow \mathsf{S}^{\mathtt{n}}(x_1) \tag{4}$$

$$\mathit{Comp}(x_1) \leftarrow \mathsf{S}^{\mathtt{t}}(x_1) \tag{5}$$

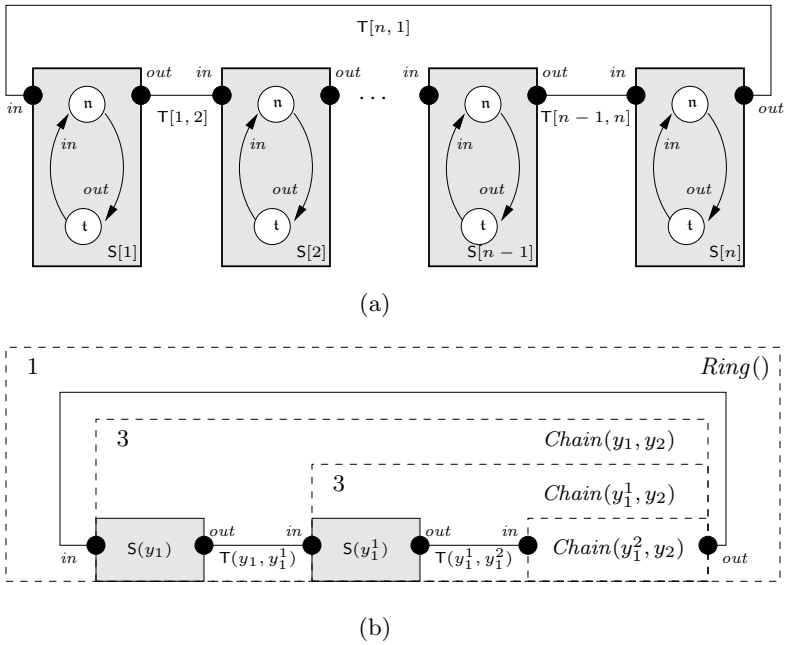

Figure 1: Recursive Specification of a Token-Ring System

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 1-5 is a commutative and associative logical connective that composes sub-systems with disjoint sets of components and interactions; for instance, in rule 2, the two components declared as $Comp(x_1)$ and $Comp(x_2)$ 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 non-deadlock actually reach a deadlock?" is challenging, because it requires a proof 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]. An extension of this resource logic has been recently developed for Hoare-style reasoning about the safety properties of programmed reconfiguration in distributed 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

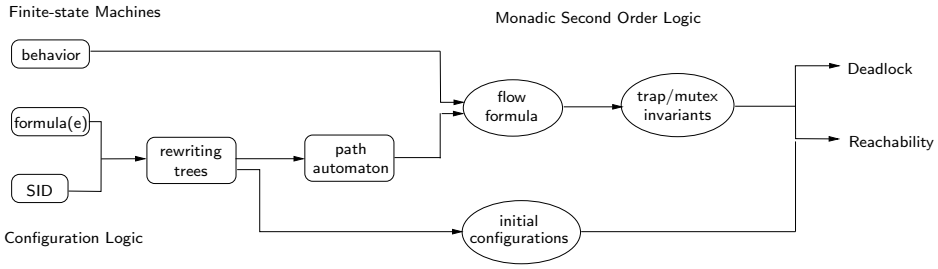

Figure 2: The Synthesis of Verification Conditions

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 (y_2) 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 over-approximation 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. By $\mathrm{pow}(A)$ 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:

- $\mathfrak{C} = \{\mathsf{C}_1, \ldots, \mathsf{C}_N\}$ is a finite set of relation symbols of arity one, called *component types*,
- $\mathfrak{I} = \{\mathsf{I}_1, \ldots, \mathsf{I}_M\}$ is a finite set of relation symbols of arity $\#(\mathsf{I}_j) \geq 1$, called *interaction types*,
- $\mathfrak{P} : \mathfrak{C} \cup \mathfrak{I} \to \mathrm{pow}(\mathbb{P}) \cup \bigcup_{i \geq 1} \mathbb{P}^i$ is an *interface* associating each component type $\mathsf{C} \in \mathfrak{C}$ a finite set of ports $\mathfrak{P}(\mathsf{C}) \in \mathrm{pow}(\mathbb{P})$ and each interaction type $\mathsf{I} \in \mathfrak{I}$ a finite tuple of ports $\mathfrak{P}(\mathsf{I}) \in \mathbb{P}^{\#(\mathsf{I})}$.

W.l.o.g., we assume that $\bigcup_{i=1}^{N} \mathfrak{P}(\mathsf{C}_i) = \mathbb{P}$ and $\mathfrak{P}(\mathsf{C}_i) \cap \mathfrak{P}(\mathsf{C}_j) = \emptyset$, for all $1 \leq i < j \leq N$, i.e. each port p belongs to the interface of exactly one component type, denoted by $comp(p)$. We denote by $\mathfrak{I}^{(k)}$ the subset of interaction types of arity k, formally $\mathfrak{I}^{(k)} \stackrel{\text{def}}{=} \{\mathsf{I} \in \mathfrak{I} \mid \#(\mathsf{I}) = k\}$, for any $k \geq 1$.

Example 1 (contd. from §1.1)**.** *The signature for the token-ring example from Figure 1a is* $\Sigma = \langle \{\mathsf{S}\}, \{\mathsf{T}\}, \mathfrak{P} \rangle$*, where* $\mathfrak{P}(\mathsf{S}) = \{in, out\}$ *and* $\mathfrak{P}(\mathsf{T}) = \langle out, in \rangle$*, 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* α over the signature $\Sigma = (\mathfrak{C}, \mathfrak{I}, \mathfrak{P})$ associates each component type C_i a set $\alpha(\mathsf{C}_i) \subseteq \mathbb{U}$ and each interaction type I_j a relation $\alpha(\mathsf{I}_j) \subseteq \mathbb{U}^{\#(\mathsf{I}_j)}$, defining:

- *components* $\mathsf{C}_i[u]$, for some $u \in \alpha(\mathsf{C}_i)$, and
- *interactions* $\mathsf{I}_j[u_1, \ldots, u_{\#(\mathsf{I}_j)}]$, for some $\langle u_1, \ldots, u_{\#(\mathsf{I}_j)} \rangle \in \alpha(\mathsf{I}_j)$.

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 $\mathsf{C}_1[u]$ and $\mathsf{C}_2[u]$ belonging to different component types and similarly, a tuple of indices $\mathbf{u}$ can refer two (or more) different interactions $\mathsf{I}_1[\mathbf{u}]$ and $\mathsf{I}_2[\mathbf{u}]$ of different interaction types, with the same arity; interactions may occur disconnected from components, e.g. $\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}]$ does not necessarily mean that each component $comp(\langle\mathfrak{P}(\mathsf{I})\rangle_i)[u_i]$, for all $i \in [1, \#(\mathsf{I})]$ is present in the structure.

Example 2 (contd. from Example 1)**.** *Letting indices be natural numbers, the structure of Figure 1a is described by the architecture α over Σ, such that $\alpha(\mathsf{S}) = [1, n]$ and $\alpha(\mathsf{T}) = \{\langle 1, 2\rangle, \langle 2, 3\rangle, \ldots, \langle n, 1\rangle\}$, for some $n \geq 2$.* ■

A *behavior* is described by a finite state machine $M = (Q, P, \rightarrow)$, where Q is a finite set of states, $P \subseteq \mathbb{P}$ is a finite set of ports and $\rightarrow \subseteq Q \times P \times Q$ is a transition relation; we write $q \xrightarrow{p} q'$ instead of $(q, p, q') \in \rightarrow$. Let $\mathbb{B}$ be the set of finite-state machines with ports from $\mathbb{P}$. The *behavior map* $\beta : \mathfrak{C} \rightarrow \mathbb{B}$ associates each component type $\mathsf{C} \in \mathfrak{C}$ with a state machine $\beta(\mathsf{C}) = (Q_{\mathsf{C}}, \mathfrak{P}(\mathsf{C}), \rightarrow_{\mathsf{C}})$, whose set of states is denoted by $states_\beta(\mathsf{C}) \stackrel{\text{def}}{=} Q_{\mathsf{C}}$. In the following, we consider w.l.o.g. that $states_\beta(\mathsf{C}_i) \cap states_\beta(\mathsf{C}_j) = \emptyset$, for all $1 \leq i < j \leq N$ and define $\mathcal{Q}_\beta \stackrel{\text{def}}{=} \bigcup_{i=1}^{N} states_\beta(\mathsf{C}_i)$.

Example 3 (contd. from Example 1)**.** *The behavior of the component type S in Figure 1a is described by the state machine $\beta(\mathsf{S}) = (\{\mathfrak{t}, \mathfrak{n}\}, \{in, out\}, \rightarrow)$, with transitions $\mathfrak{n} \xrightarrow{in} \mathfrak{t}$ and $\mathfrak{t} \xrightarrow{out} \mathfrak{n}$.* ■

For a port p we denote by $T_\beta(p)$ the set of transitions labeled by p in the finite-state machine associated by β to $comp(p)$. We extend this notation to tuples of ports by taking $T_\beta(\langle p_1, \ldots, p_n\rangle) \stackrel{\text{def}}{=} T_\beta(p_1) \times \cdots \times T_\beta(p_n)$. That is, the set $T_\beta(\langle p_1, \ldots, p_n\rangle)$ contains n-tuples of transitions $\langle t_1, \ldots, t_n\rangle$ labeled respectively with ports $\langle p_1, \ldots, p_n\rangle$. For an interaction I we use $T_\beta(\mathsf{I})$ as a shortcut for $T_\beta(\mathfrak{P}(\mathsf{I}))$. In particular, $T_\beta(\mathsf{I})$ contains the set of tuples of component's transitions that are *synchronizing* by interaction I.

A *system* $\mathfrak{S}$ is a tuple (Σ, α, β), where α is an architecture and β is a behavior associated with the signature Σ. When Σ is clear from the context, we omit it and denote a system as (α, β).

2.2. Operational Semantics

We represent the operational semantics of a system as a Petri net, recalled below for self-containment reasons. A *Petri net* is a tuple $\mathsf{N} = (S, T, E)$, where S is a set of *places*, T is a set of *transitions*, $S \cap T = \emptyset$, and $E \subseteq (S \times T) \cup (T \times S)$ 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 ${}^\bullet x \stackrel{\text{def}}{=} \{y \in S \cup T \mid E(y, x) = 1\}$, $x^\bullet \stackrel{\text{def}}{=} \{y \in S \cup T \mid E(x, y) = 1\}$ 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 only if $\mathrm{m}(s) > 0$ for each place $s \in {}^\bullet t$. We write $\mathrm{m} \xrightarrow{t} \mathrm{m}'$ whenever t is enabled in m and $\mathrm{m}'(s) = \mathrm{m}(s) - E(s, t) + E(t, s)$, for all $s \in S$ and $t \in T$. A sequence

of transitions $\sigma = t_1, \ldots, t_n$ is a *firing sequence*, written $\mathrm{m} \xrightarrow{\sigma} \mathrm{m}'$ if and only if either (i) $n = 0$ and $\mathrm{m} = \mathrm{m}'$, or (ii) $n \geq 1$ and there exist markings $\mathrm{m}_1, \ldots, \mathrm{m}_{n-1}$ such that $\mathrm{m} \xrightarrow{t_1} \mathrm{m}_1 \ldots \mathrm{m}_{n-1} \xrightarrow{t_n} \mathrm{m}'$. A marking m is a *deadlock* of a Petri net $\mathsf{N} = (S, T, E)$ if and only if no transition $t \in T$ is enabled in m and let $\mathit{Dead}(\mathsf{N})$ denote the set of deadlocks of N.

A *marked Petri net* is a pair $\mathcal{N} = (\mathsf{N}, \mathrm{m}_0)$, where m_0 is the *initial marking* of N. A firing sequence is *initial* if it starts in m_0. A marking m is *reachable* in $\mathcal{N}$ if there exists an initial firing sequence ending in m. Let $\mathit{Reach}(\mathcal{N})$ (resp. $\mathit{Step}(\mathcal{N})$) be the set of markings reachable (in one step) in $\mathcal{N}$. For simplicity, we write $\mathit{Reach}(\mathsf{N}, \mathrm{m}_0)$ and $\mathit{Step}(\mathsf{N}, \mathrm{m}_0)$ instead of $\mathit{Reach}((\mathsf{N}, \mathrm{m}_0))$ and $\mathit{Step}((\mathsf{N}, \mathrm{m}_0))$, respectively. A marked Petri net $\mathcal{N}$ is *boolean*[2] if $\mathrm{m}(s) \leq 1$, for each $s \in S$ and $\mathrm{m} \in \mathit{Reach}(\mathcal{N})$. All marked Petri nets considered in the following will be boolean and we blur the distinction between a marking $\mathrm{m} : S \to \{0, 1\}$ 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* $\mathsf{N}(\mathfrak{S}) = (S, T, E)$*, where:*

$$S \stackrel{\text{def}}{=} \bigcup_{\mathsf{C} \in \mathfrak{C}} \{\ q[u] \ \mid\ q \in \mathit{states}_\beta(\mathsf{C}),\ u \in \alpha(\mathsf{C})\ \}$$

$$T \stackrel{\text{def}}{=} \bigcup_{\mathsf{I} \in \mathfrak{I}} \{\ (\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}], \langle t_1, \ldots, t_{\#(\mathsf{I})} \rangle) \ \mid\ \langle p_1, \ldots, p_{\#(\mathsf{I})} \rangle = \mathfrak{P}(\mathsf{I}),\ \langle u_1, \ldots, u_{\#(\mathsf{I})} \rangle \in \alpha(\mathsf{I}),\ \langle t_1, \ldots, t_{\#(\mathsf{I})} \rangle \in T_\beta(\mathsf{I}),\ \forall i, j \in [1, \#(\mathsf{I})].\ i \neq j \Rightarrow u_i \neq u_j \ \mathit{or}\ \mathit{comp}(p_i) \neq \mathit{comp}(p_j)\ \}$$

$$E \stackrel{\text{def}}{=} \bigcup_{\mathsf{I} \in \mathfrak{I}} \{\ (q[u_i], (\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}], \langle t_1, \ldots, t_{\#(\mathsf{I})} \rangle)),\ ((\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}], \langle t_1, \ldots, t_{\#(\mathsf{I})} \rangle), q'[u_i]) \ \mid\ t_i = (q \xrightarrow{p_i} q'),\ i \in [1, \#(\mathsf{I})]\ \}$$

The places, transitions and edges of $\mathsf{N}(\mathfrak{S})$ are defined jointly by the architecture α and the behavior β. For every component $\mathsf{C}[u]$ in α, the Petri net contains the places $q[u]$, for each state q in $\mathit{states}_\beta(\mathsf{C})$. For every interaction $\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}]$ in α, the Petri net contains one transition for every tuple $\langle t_1, \ldots, t_{\#(\mathsf{I})} \rangle$ of transitions of component types behavior, which are *synchronizing* according to I, that is, where their labeling ports $\langle p_1, \ldots, p_{\#(\mathsf{I})} \rangle$ form the tuple $\mathfrak{P}(\mathsf{I})$. Moreover, a transition corresponding to an interaction $\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}]$ involves pairwise distinct components; the last condition in the definition of T above requires that $u_i \neq u_j$ or $\mathit{comp}(p_i) \neq \mathit{comp}(p_j)$. Finally, edges are defined according to the tuple of synchronizing transitions $\langle t_1, \ldots, t_{\#(\mathsf{I})} \rangle$ and connect to pre- and post- places, respectively $q[u_i]$ and $q'[u_i]$, for each involved component $\mathsf{C}[u_i]$. For the sake of clarity we omit writing the tuple $\langle t_1, \ldots, t_{\#(\mathsf{I})} \rangle$ when it is determined by I, namely for those behaviors where a port labels exactly one transition, in each state machine, as it is the case in the example below:

[2] Boolean Petri nets are sometimes called 1-safe or 1-bounded in the literature.

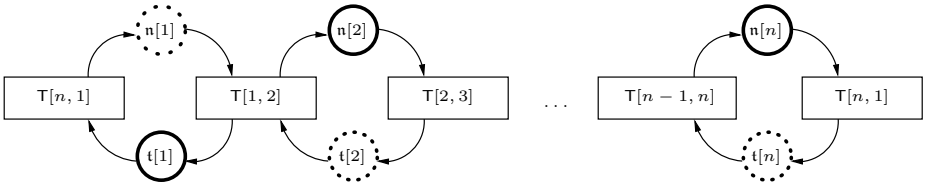

Figure 3: The Execution Semantics of a Token-Ring System

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* $\mathsf{T}[1,2], \mathsf{T}[2,3], \ldots, \mathsf{T}[n-1,n], \mathsf{T}[n,1]$ *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* $(i \mod n) + 1$ *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 $\mathsf{N}(\mathfrak{S})$, that contain exactly one state $q \in \mathit{states}_\beta(\mathsf{C})$ per component $\mathsf{C}[u]$, such that $u \in \alpha(\mathsf{C})$. 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* $\mathsf{N}(\mathfrak{S}) = (S, T, E)$ *is* precise *if and only if, for each* $\mathsf{C} \in \mathfrak{C}$ *and each* $u \in \alpha(\mathsf{C})$*, we have* $\|\mathrm{m} \cap \{q[u] \mid q \in \mathit{states}_\beta(\mathsf{C})\}\| = 1$*.*

Proposition 1. *Given a system* $\mathfrak{S}$ *and a precise initial marking* m_0 *of* $\mathsf{N}(\mathfrak{S})$*, every marking* $\mathrm{m} \in \mathit{Reach}(\mathsf{N}(\mathfrak{S}), \mathrm{m}_0)$ *is precise.*

Proof. Let $\mathfrak{S} = (\Sigma, \alpha)$ and $\mathrm{m}' \in \mathit{Reach}(\mathsf{N}(\mathfrak{S}), \mathrm{m}_0)$ be a marking. The proof goes by the length $n \geq 0$ of the firing sequence taking m_0 into m'. For the base case $n = 0$, we have that $\mathrm{m}' = \mathrm{m}_0$ is precise, by hypothesis. For the inductive step $n \geq 1$, let m be the predecessor of m' in the sequence, hence m is precise for $\mathfrak{S}$, by the inductive hypothesis. Then, by Def. 1, there exists a transition $(\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}], \langle t_1, \ldots, t_{\#(\mathsf{I})} \rangle)$ in T, such that

$$\mathrm{m}' = (\mathrm{m} \setminus \{q_1[u_1], \ldots, q_{\#(\mathsf{I})}[u_{\#(\mathsf{I})}]\}) \cup \{q'_1[u_1], \ldots, q'_{\#(\mathsf{I})}[u_{\#(\mathsf{I})}]\} \tag{6}$$

where $\mathfrak{P}(\mathsf{I}) = \langle p_1, \ldots, p_{\#(\mathsf{I})} \rangle$ and $t_i = (q_i \xrightarrow{p_i} q'_i) \in \rightarrow_{\mathit{comp}(p_i)}$ for all $i \in [1, \#(\mathsf{I})]$ and moreover $u_i \neq u_j$ or $\mathit{comp}(p_i) \neq \mathit{comp}(p_j)$ for all distinct $i, j \in [1, \#(\mathsf{I})]$.

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

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

  $$\mathrm{m}' \cap \{q[u] \mid q \in \mathit{states}_\beta(\mathsf{C})\} = \{q'_i[u]\}, \text{ by definition of } \mathrm{m}' \text{ in (6)}$$

- $u = u_i$ for some $i \in [1, \#(\mathsf{I})]$ and $q^* \neq q_i$. In this case, as the transition $(\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}], \langle t_1, \ldots, t_{\#(\mathsf{I})} \rangle)$ fires in m it follows that $q_i[u] \in \mathrm{m}$. But

then $q_i[u] \in \mathrm{m} \cap \{q[u] \mid q \in \mathit{states}_\beta(\mathsf{C})\} = \{q^*[u]\}$ implies $q_i = q^*$, which is a contradiction.

- $u \notin \{u_1, \ldots, u_{\#(\mathsf{I})}\}$. In this case, we have:

$$\mathrm{m} \cap \{q[u] \mid q \in \mathit{states}_\beta(\mathsf{C})\} = \mathrm{m}' \cap \{q[u] \mid q \in \mathit{states}_\beta(\mathsf{C})\}$$

In all non-contradictory cases, we obtain that $\|\mathrm{m} \cap \{q[u] \mid q \in \mathit{states}_\beta(\mathsf{C})\}\| = 1$. Because $\mathsf{C} \in \mathfrak{C}$ and $u \in \alpha(\mathsf{C})$ 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 $\beta : \mathfrak{C} \to \mathbb{B}$ be a behavior map, fixed for the rest of this section.

Definition 3. *A* configuration *of a system* $\mathfrak{S} = (\Sigma, \alpha, \beta)$ *is a pair* (α, m)*, where* α *is an architecture over the signature* Σ *and* m *is a precise marking of* $\mathsf{N}(\mathfrak{S})$*.*

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:

Definition 4. *Two architectures* α_1 *and* α_2*, over the signature* $\Sigma = (\mathfrak{C}, \mathfrak{I}, \mathfrak{P})$*, are* disjoint *if and only if* $\alpha_1(\mathsf{C}) \cap \alpha_2(\mathsf{C}) = \emptyset$*, for all* $\mathsf{C} \in \mathfrak{C}$ *and* $\alpha_1(\mathsf{I}) \cap \alpha_2(\mathsf{I}) = \emptyset$*, for all* $\mathsf{I} \in \mathfrak{I}$*. If* α_1 *and* α_2 *are disjoint, the* composition *of configurations* (α_1, m_1) *and* (α_2, m_2) *is* $(\alpha_1, \mathrm{m}_1) \bullet (\alpha_2, \mathrm{m}_2) \stackrel{\text{def}}{=} (\alpha_1 \cup \alpha_2, \mathrm{m}_1 \cup \mathrm{m}_2)$*, where* $\alpha_1 \cup \alpha_2$ *denotes the pointwise union of the architectures* α_1 *and* α_2*:*

- $(\alpha_1 \cup \alpha_2)(\mathsf{C}) = \alpha_1(\mathsf{C}) \cup \alpha_2(\mathsf{C})$*, for each* $\mathsf{C} \in \mathfrak{C}$*,*
- $(\alpha_1 \cup \alpha_2)(\mathsf{I}) = \alpha_1(\mathsf{I}) \cup \alpha_2(\mathsf{I})$*, for each* $\mathsf{I} \in \mathfrak{I}$*.*

The composition $(\alpha_1, \mathrm{m}_1) \bullet (\alpha_2, \mathrm{m}_2)$ *is undefined if* α_1 *and* α_2 *are not disjoint.*

It is easy to check that the composition of configurations (α_1, m_1) and (α_2, m_2) 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 $\mathsf{N}(\Sigma, \alpha_1 \cup \alpha_2, \beta)$.

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 CL are inductively defined by the following syntax:

$$\phi ::= \mathsf{emp} \mid \mathsf{C}(x) \mid \mathsf{C}^q(x) \mid \mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})}) \mid \mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \mid \phi * \phi \mid \exists x \,.\, \phi$$

where $\mathsf{C} \in \mathfrak{C}$ are component types, $q \in \mathit{states}_\beta(\mathsf{C})$ are states, $\mathsf{I} \in \mathfrak{I}$ are interaction types, $\mathcal{A} \in \mathbb{A}$ are predicate symbols and $x_1, x_2, \ldots \in \mathbb{X}^1$ denote first-order variables. Atomic formulæ of the form $\mathsf{C}(x)$ or $\mathsf{C}^q(x)$, $\mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})$ and $\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})})$ are called *component*, *interaction* and *predicate atoms*, respectively. The logical connective $*$ is an associative and commutative *separating conjunction* operator.

By $\mathrm{fv}(\phi)$ we denote the set of free variables that do not occur within the scope of a quantifier; ϕ is called *quantifier-free* (q.f.) if and only if it has no quantifiers and a *sentence* if and only if $\mathrm{fv}(\phi) = \emptyset$, respectively. A *substitution* is a partial mapping $\theta : \mathbb{X}^1 \to \mathbb{X}^1$ and $\phi\theta$ is the formula obtained by replacing each variable $x \in \mathrm{fv}(\phi) \cap \mathrm{dom}(\theta)$ by $\theta(x)$ in ϕ, where $\mathrm{dom}(\theta) \stackrel{\text{def}}{=} \{x \in \mathbb{X}^1 \mid \theta \text{ is defined at } x\}$. We denote by $[x_1/y_1, \ldots, x_k/y_k]$ 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 $\mathbf{x} = \langle x_1, \ldots, x_k \rangle$ and $\mathbf{y} = \langle y_1, \ldots, y_k \rangle$.

Intuitively, a formula emp describes configurations with empty architecture, $\mathsf{C}(x)$ (resp. $\mathsf{C}^q(x)$) 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 $\mathsf{I}(x_1, \ldots, x_k)$ describes a single interaction of type I, between components indexed by $x_1, \ldots, x_k$, respectively. The formula $\mathsf{C}^{q_1}(x_1) * \ldots * \mathsf{C}^{q_k}(x_k) * \mathsf{I}(x_1, \ldots, x_k)$ 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 I. The formula $\mathsf{I}(x_1, \ldots, x_k) * \mathsf{I}(x'_1, \ldots, x'_k)$ states the existence of two interactions of type I, with *distinct* tuples of indices, given by the values of $\langle x_1, \ldots, x_k \rangle$ and $\langle x'_1, \ldots, x'_k \rangle$, respectively, i.e. the values of x_i and x'_i must differ for at least one $i \in [1, k]$.

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

- a *store* $\mathfrak{s} : \mathbb{X}^1 \to \mathbb{U}$, i.e. a function mapping variables to indices, and
- a *set of inductive definitions* (SID) Δ consisting of rules of the form $\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \phi$, where ϕ is a CL formula such that $\mathrm{fv}(\phi) = \{x_1, \ldots, x_{\#(\mathcal{A})}\}$.

The satisfaction relation is defined inductively on the structure of the formulæ:

$(\alpha, \mathrm{m}) \models^{\mathfrak{s}}_{\Delta} \mathsf{emp} \iff \alpha(\mathsf{C}) = \emptyset$, for all $\mathsf{C} \in \mathfrak{C}$, $\alpha(\mathsf{I}) = \emptyset$, for all $\mathsf{I} \in \mathfrak{I}$ and $\mathrm{m} = \emptyset$

$(\alpha, \mathrm{m}) \models^{\mathfrak{s}}_{\Delta} \mathsf{C}(x) \iff \alpha(\mathsf{C}) = \{\mathfrak{s}(x)\}$, $\alpha(\mathsf{C}') = \emptyset$, for all $\mathsf{C}' \in \mathfrak{C} \setminus \{\mathsf{C}\}$, $\alpha(\mathsf{I}) = \emptyset$, for all $\mathsf{I} \in \mathfrak{I}$ and $\mathrm{m} = \{q[\mathfrak{s}(x)]\}$, for some $q \in \mathit{states}_\beta(\mathsf{C})$

$(\alpha, \mathrm{m}) \models^{\mathfrak{s}}_{\Delta} \mathsf{C}^q(x) \iff \alpha(\mathsf{C}) = \{\mathfrak{s}(x)\}$, $\alpha(\mathsf{C}') = \emptyset$, for all $\mathsf{C}' \in \mathfrak{C} \setminus \{\mathsf{C}\}$, $\alpha(\mathsf{I}) = \emptyset$, for all $\mathsf{I} \in \mathfrak{I}$ and $\mathrm{m} = \{q[\mathfrak{s}(x)]\}$

$(\alpha, \mathrm{m}) \models^{\mathfrak{s}}_{\Delta} \mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I}_k)}) \iff \alpha(\mathsf{C}) = \emptyset$, for all $\mathsf{C} \in \mathfrak{C}$, $\alpha(\mathsf{I}) = \{\langle \mathfrak{s}(x_1), \ldots, \mathfrak{s}(x_{\#(\mathsf{I}_k)}) \rangle\}$, $\alpha(\mathsf{I}') = \emptyset$, for all $\mathsf{I}' \in \mathfrak{I} \setminus \{\mathsf{I}\}$ and $\mathrm{m} = \emptyset$

$(\alpha, \mathrm{m}) \models^{\mathfrak{s}}_{\Delta} \mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \iff (\alpha, \mathrm{m}) \models^{\mathfrak{s}}_{\Delta} \phi[y_1/x_1, \ldots, y_{\#(\mathcal{A})}/x_{\#(\mathcal{A})}]$, for some rule $\mathcal{A}(y_1, \ldots, y_{\#(\mathcal{A})}) \leftarrow \phi \in \Delta$

$(\alpha, \mathrm{m}) \models^{\mathfrak{s}}_{\Delta} \phi_1 * \phi_2 \iff$ there exist configurations (α_1, m_1), (α_2, m_2) such that $(\alpha, \mathrm{m}) = (\alpha_1, \mathrm{m}_1) \bullet (\alpha_2, \mathrm{m}_2)$ and $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}}_{\Delta} \phi_i$, for $i = 1, 2$.

$(\alpha, \mathrm{m}) \models^{\mathfrak{s}}_{\Delta} \exists x \,.\, \phi_1 \iff (\alpha, \mathrm{m}) \models^{\mathfrak{s}'}_{\Delta} \phi_1$, for some store $\mathfrak{s}'$ that agrees with $\mathfrak{s}$ on all variables from $\mathbb{X}^1 \setminus \{x\}$

To simplify the notation, we consider that Δ is fixed and omit the Δ subscript in the following. If the formula ϕ 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, (α, m) is said to be a *model* of ϕ and denote by $[\![\phi]\!]$ the set of models of the sentence ϕ. For two sentences ϕ and ψ, we say that ϕ *entails* ψ if $[\![\phi]\!] \subseteq [\![\psi]\!]$, 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:*

$$
\begin{array}{lcl}
\mathit{Chain}_{0,0}(x, x) & \leftarrow & \mathsf{S}(x) \\
\mathit{Chain}_{0,1}(x, x) & \leftarrow & \mathsf{S}^{\mathfrak{t}}(x) \\
\mathit{Chain}_{1,0}(x, x) & \leftarrow & \mathsf{S}^{\mathfrak{n}}(x) \\
\mathit{Chain}_{n,t}(x, z) & \leftarrow & \exists y.\ \mathsf{S}^{\mathfrak{t}}(x) * \mathsf{T}(x, y) * \mathit{Chain}_{n,t \dot{-} 1}(y, z) \\
\mathit{Chain}_{n,t}(x, z) & \leftarrow & \exists y.\ \mathsf{S}^{\mathfrak{n}}(x) * \mathsf{T}(x, y) * \mathit{Chain}_{n \dot{-} 1,t}(y, z)
\end{array}
$$

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* ϕ *of* CL*, if* $(\alpha, \mathrm{m}) \in [\![\phi]\!]$ *then* m *is a precise marking of the Petri net* $\mathsf{N}(\Sigma, \alpha, \beta)$*.*

Proof. We prove a more general statement: for each formula ϕ and each store $\mathfrak{s}$, if $(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \phi$ then m is a precise marking of $\mathsf{N}(\Sigma, \alpha, \beta)$. The proof is by induction on the definition of the $\models^{\mathfrak{s}}$ relation, by distinguishing the following cases:

- $\phi \in \{\mathsf{emp}, \mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})\}$: $\mathrm{m} = \emptyset$ and $\alpha(\mathsf{C}) = \emptyset$, for all $\mathsf{C} \in \mathfrak{C}$, thus m is trivially precise for $\mathsf{N}(\Sigma, \alpha, \beta)$.
- $\phi = \mathsf{C}(x)$: $\mathrm{m} = \{q[\mathfrak{s}(x)]\}$, for some $q \in \mathit{states}_{\beta}(\mathsf{C})$, $\alpha(\mathsf{C}) = \{\mathfrak{s}(x)\}$ and $\alpha(\mathsf{C}') = \emptyset$, for all $\mathsf{C}' \in \mathfrak{C} \setminus \{\mathsf{C}\}$, hence m is precise for $\mathsf{N}(\Sigma, \alpha, \beta)$.
- $\phi = \mathsf{C}^q(x)$: $\mathrm{m} = \{q[\mathfrak{s}(x)]\}$, $\alpha(\mathsf{C}) = \{\mathfrak{s}(x)\}$ and $\alpha(\mathsf{C}') = \emptyset$, for all $\mathsf{C}' \in \mathfrak{C} \setminus \{\mathsf{C}\}$, hence m is precise for $\mathsf{N}(\Sigma, \alpha, \beta)$.
- $\phi = \mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})})$: $(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \psi[y_1/x_1, \ldots, y_{\#(\mathcal{A})}/x_{\#(\mathcal{A})}]$, for some rule $\mathcal{A}(y_1, \ldots, y_{\#(\mathcal{A})}) \leftarrow \psi \in \Delta$ 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 $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}} \phi_i$, for $i = 1, 2$, such that α_1 and α_2 are disjoint, $\alpha = \alpha_1 \cup \alpha_2$ and $\mathrm{m} = \mathrm{m}_1 \cup \mathrm{m}_2$. By the inductive hypothesis, m_i is a precise marking of $\mathsf{N}(\Sigma, \alpha_i, \beta)$, for $i = 1, 2$. Let $\mathsf{C} \in \mathfrak{C}$ be a component type and let $u \in \alpha(\mathsf{C})$ be an index. Then either $u \in \alpha_1(\mathsf{C})$ or $u \in \alpha_2(\mathsf{C})$, but not both. Let us consider the case $u \in \alpha_1(\mathsf{C}) \setminus \alpha_2(\mathsf{C})$, the other case being symmetric. Then $\mathrm{m} \cap \{q[u] \mid q \in \mathit{states}(\mathsf{C})\} = \mathrm{m}_1 \cap \{q[u] \mid q \in \mathit{states}(\mathsf{C})\}$, hence $\|\mathrm{m} \cap \{q[u] \mid q \in \mathit{states}(\mathsf{C})\}\| = \|\mathrm{m}_1 \cap \{q[u] \mid q \in \mathit{states}(\mathsf{C})\}\| = 1$. 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 ϕ, a SID Δ, a behavior map $\beta : \mathfrak{C} \to \mathbb{B}$ and a tuple of states $\langle q_1, \ldots, q_n \rangle \in \mathcal{Q}_\beta^n$ for some $n \in \mathbb{N}$. They are defined as follows:

- $\mathit{deadlock}(\phi, \Delta, \beta)$: is there a configuration (α, m), such that $(\alpha, \mathrm{m}) \models_\Delta \phi$ and $\mathit{Reach}(\mathsf{N}(\Sigma, \alpha, \beta), \mathrm{m}) \cap \mathit{Dead}(\mathsf{N}(\Sigma, \alpha, \beta)) \neq \emptyset$?
- $\mathit{reach}(\phi, \langle q_1, \ldots, q_k \rangle, \Delta, \beta)$: are there configurations (α, m), (α, m') and indices $u_1, \ldots, u_k \in \mathbb{U}$, such that $(\alpha, \mathrm{m}) \models_\Delta \phi$, $\mathrm{m}' \in \mathit{Reach}(\mathsf{N}(\Sigma, \alpha, \beta), \mathrm{m})$ and $\{q_i[u_i] \mid i \in [1, k]\} \subseteq \mathrm{m}'$, with $q_1[u_1], \ldots, q_k[u_k]$ pairwise distinct?

The above queries occur typically as correctness conditions in system verification. For instance, proving that, in every system described by a formula ϕ, no freeze configuration is reachable means showing that $\mathit{deadlock}(\phi, \Delta, \beta)$ does not hold. Similarly, proving that each system described by ϕ stays clear of a set of error configurations e.g., at most one component is in some critical state q_{crit} at any time amounts to proving that $\mathit{reach}(\phi, \langle q_{crit}, q_{crit} \rangle, \Delta, \beta)$ 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 $\Sigma = (\{\mathsf{C}\}, \mathfrak{I}, \mathfrak{P})$, a sentence ϕ, interpreted over a SID Δ, is* linear *if and only if $\phi \models_{\Delta \cup \Lambda} \phi_{\mathcal{L}}$, where Λ consists of the rules:*

$$\begin{array}{lcl} \mathcal{L}(x, x) & \leftarrow & \mathsf{C}(x) * \mathop{\Large *}_{\mathsf{I} \in \mathfrak{I}^{(1)}} \mathsf{I}(x) \\ \mathcal{L}(x, y) & \leftarrow & \exists z \,.\, \mathsf{C}(x) * \mathop{\Large *}_{\mathsf{I} \in \mathfrak{I}^{(1)}} \mathsf{I}(x) * \mathop{\Large *}_{\mathsf{I} \in \mathfrak{I}^{(2)}} \mathsf{I}(x, z) * \mathcal{L}(z, y) \end{array}$$

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

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:*

$$\begin{array}{rcl} \mathit{LinearDeadlock} & \stackrel{\text{def}}{=} & \{\mathit{deadlock}(\phi, \Delta, \beta) \mid \phi \models_{\Delta \cup \Lambda} \phi_{\mathcal{L}}\} \\ \mathit{LinearReachability} & \stackrel{\text{def}}{=} & \{\mathit{reach}(\phi, \langle q_1 \rangle, \Delta, \beta) \mid \phi \models_{\Delta \cup \Lambda} \phi_{\mathcal{L}}\} \end{array}$$

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 $1 : \ \mathit{stmt}_1;\ 2 : \ \mathit{stmt}_2;\ \ldots;\ m : \ \mathit{stmt}_m$, where each statement stmt_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 $\Sigma_{PT} = (\mathfrak{C}_{PT}, \mathfrak{I}_{PT}, \mathfrak{P}_{PT})$ is presented in Figure 4(a) and consists of one component type C_{PT} and ten interaction types $\mathfrak{I}_{PT}$, 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 β_M of the component type C_{PT}. 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 γ_k and the presence ($\top$) or absence

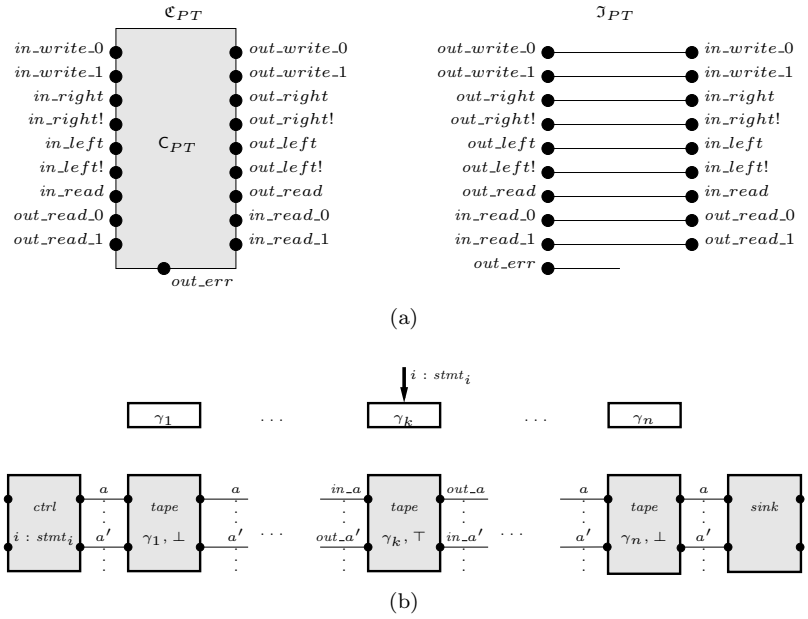

Figure 4: Undecidability of Deadlock and Reachability for Linear Component-based Systems

($\perp$) of the machine head at that cell. They react to commands (received through *in*-ports) by changing their state and/or further issuing commands 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 Σ_{PT} and the behavior β_M as defined in the previous section. Furthermore, consider the following set Δ_w of rules, where $\mathfrak{I}_1 \stackrel{\text{def}}{=} \mathfrak{I}_{PT}^{(1)}$, $\mathfrak{I}_2 \stackrel{\text{def}}{=} \mathfrak{I}_{PT}^{(2)}$:

$$\begin{array}{lcl} \mathit{Zeroes}(x,x) & \leftarrow & \mathsf{C}_{PT}^{(0,\perp)}(z) * \circledast_{\mathsf{I} \in \mathfrak{I}_1} \mathsf{I}(x) \\ \mathit{Zeroes}(x,y) & \leftarrow & \exists z \,.\, \mathsf{C}_{PT}^{(0,\perp)}(x) * \circledast_{\mathsf{I} \in \mathfrak{I}_1} \mathsf{I}(x) * \circledast_{\mathsf{I} \in \mathfrak{I}_2} \mathsf{I}(x,z) * \mathit{Zeroes}(z,y) \end{array}$$

[3] w can be augmented with extra zeroes on the right

control state machine

$Q^M_{ctrl} \stackrel{\text{def}}{=} \{i, (i, \_) \mid i \in [1, m+1]\}$ represents the program M

$i:$ write 0	$i \xrightarrow{out\_write\_0} i{+}1$
$i:$ write 1	$i \xrightarrow{out\_write\_1} i{+}1$
$i:$ go right	$i \xrightarrow{out\_right} i{+}1$
$i:$ go left	$i \xrightarrow{out\_left} i{+}1,\ i \xrightarrow{out\_left!} (i, \_),\ (i, \_) \xrightarrow{out\_err} (i, \_)$
$i:$ goto step j if read 0	$i \xrightarrow{out\_read} (i, \_),\ (i, \_) \xrightarrow{in\_read\_0} j,\ (i, \_) \xrightarrow{in\_read\_1} i{+}1$
$i:$ goto step j if read 1	$i \xrightarrow{out\_read} (i, \_),\ (i, \_) \xrightarrow{in\_read\_1} j,\ (i, \_) \xrightarrow{in\_read\_0} i{+}1$
$i:$ stop	$/ * \mathit{nothing} * /$

tape state machine

$Q_{tape} \stackrel{\text{def}}{=} \{(\gamma, \pi), (\gamma, \pi, a) \mid \gamma \in \{0,1\},\ \pi \in \{\bot, \top\},\ a \in \mathfrak{I}_{PT} \setminus \{err\}\}$

$(\gamma, \bot) \xrightarrow{in\_a} (\gamma, \bot, a)$	$(\gamma, \bot, a) \xrightarrow{out\_a} (\gamma, \bot)$	$\forall a \neq \mathit{right}!, \mathit{left}!, \mathit{err}$
$(\gamma, \bot) \xrightarrow{in\_right!} (\gamma, \top)$	$(\gamma, \bot, \mathit{left}) \xrightarrow{out\_left!} (\gamma, \top)$	
$(\gamma, \top) \xrightarrow{in\_write\_\gamma'} (\gamma', \top)$		
$(\gamma, \top) \xrightarrow{in\_read} (\gamma, \top, \mathit{read})$	$(\gamma, \top, \mathit{read}) \xrightarrow{out\_read\_\gamma} (\gamma, \top)$	
$(\gamma, \top) \xrightarrow{in\_right} (\gamma, \top, \mathit{right})$	$(\gamma, \top, \mathit{right}) \xrightarrow{out\_right!} (\gamma, \bot)$	
$(\gamma, \top) \xrightarrow{in\_left!} (\gamma, \bot)$		

sink state machine with states $Q_{sink} \stackrel{\text{def}}{=} \{\mathit{idle}, \mathit{busy}\}$

$\mathit{idle} \xrightarrow{in\_right!} \mathit{busy}$	$\mathit{busy} \xrightarrow{out\_err} \mathit{busy}$

Table 1: The behavior β_M of C_{PT} for a Post-Turing machine executing a program M

$$
\begin{array}{rcl}
\mathit{Word\_w}(x,y) & \leftarrow & \exists z_2\ \ldots\ \exists z_{n-1}\ .\ \mathsf{C}_{PT}^{(\gamma_1,\top)}(x) * \circledast_{\mathsf{I} \in \mathfrak{I}_1} \mathsf{I}(x) * \\
& & \quad \circledast_{\mathsf{I} \in \mathfrak{I}_2} \mathsf{I}(x, z_2) * \mathsf{C}_{PT}^{(\gamma_2,\bot)}(z_2) * \circledast_{\mathsf{I} \in \mathfrak{I}_1} \mathsf{I}(z_2) * \\
& & \quad \circledast_{\mathsf{I} \in \mathfrak{I}_2} \mathsf{I}(z_2, z_3) * \mathsf{C}_{PT}^{(\gamma_3,\bot)}(z_3) * \circledast_{\mathsf{I} \in \mathfrak{I}_1} \mathsf{I}(z_3) * \\
& & \quad \ldots \\
& & \quad \circledast_{\mathsf{I} \in \mathfrak{I}_2} \mathsf{I}(z_{n-1}, y) * \mathsf{C}_{PT}^{(\gamma_n,\bot)}(y) * \circledast_{\mathsf{I} \in \mathfrak{I}_1} \mathsf{I}(y) \\
\mathit{Tape\_w}(x,y) & \leftarrow & \exists z_1\ \exists w_1\ \exists z_2\ \exists w_2\ .\ \mathit{Zeroes}(x, z_1) * \\
& & \quad \circledast_{\mathsf{I} \in \mathfrak{I}_2} \mathsf{I}(z_1, w_1) * \mathit{Word\_w}(w_1, w_2) \\
& & \quad \circledast_{\mathsf{I} \in \mathfrak{I}_2} \mathsf{I}(w_2, z_2) * \mathit{Zeroes}(z_2, y) \\
\mathit{Init\_w}(x,y) & \leftarrow & \exists z_1\ \exists z_2\ .\ \mathsf{C}_{PT}^{1}(x) * \circledast_{\mathsf{I} \in \mathfrak{I}_1} \mathsf{I}(x) \\
& & \quad \circledast_{\mathsf{I} \in \mathfrak{I}_2} \mathsf{I}(x, z_1) * \mathit{Tape\_w}(z_1, z_2) * \\
& & \quad \circledast_{\mathsf{I} \in \mathfrak{I}_2} \mathsf{I}(z_2, y) * \mathsf{C}_{PT}^{\mathit{idle}}(y) * \circledast_{\mathsf{I} \in \mathfrak{I}_1} \mathsf{I}(y)
\end{array}
$$

and let $\phi_w \stackrel{\text{def}}{=} \exists x\ \exists y\ .\ \mathit{Init\_w}(x,y)$. Intuitively, ϕ_w is a linear sentence which defines the valid (initial) configurations of the linear system encoding the Post-Turing 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 $deadlock(\phi_w, \Delta_w, \beta_M)$ 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 continuously 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], 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 the previous assertions to the following one:

3. the answer to $reach(\phi_w, \langle m \rangle, \Delta_w, \beta_M)$ 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 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 1). 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 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 §6 uses rewriting trees as backbones for the encoding of sets of configurations in MSO interpreted over trees [12]. In particular, the component indices from the set $\mathbb{U}$ are going to be interpreted as nodes of rewriting trees. As will be shown below, this interpretation of logical variables loses no generality, because component indices can only be compared for equality; their particular nature is of no importance from the point of view of the verification problems considered. Furthermore, the interpretation of component indices as tree nodes enables the use of a decidable fragment of MSO, to encode verification conditions.

The mapping of variables to the nodes of a rewriting tree is uniquely defined by the occurrences of the component atoms in the tree. More precisely, because all atomic propositions that occur in the rewriting tree are joined by separating conjunctions, the variables of all component atoms $\mathsf{C}(x)$ with the same component type C must be interpreted as different indices, or else the formula corresponding to the tree would not be satisfiable. Hence, a rewriting tree uniquely defines an architecture α over a given signature Σ and, implicitly, a Petri net $\mathsf{N}(\Sigma, \alpha, \beta)$, for a given behavior map β (Def. 1).

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 κ, called *nodes* in the following. We denote the concatenation of nodes $w, u \in [1, \kappa]^*$ as $w \cdot u$ or simply wu, when no confusion arises. A set of nodes $T \subseteq [1, \kappa]^*$ is said to be:

1. *prefix-closed* if $wi \in T$, for some $i \in [1, \kappa]$, only if $w \in T$, and
2. *complete* if $wi \in T$, for some $i \in [1, \kappa]$, only if $wj \in T$, for all $j \in [1, i-1]$.

A *κ-ary tree* $\mathcal{T}$ is a function mapping a complete prefix-closed set of nodes $\mathrm{nodes}(\mathcal{T})$ into a finite set of labels. The *root* of $\mathcal{T}$ is the empty sequence ϵ, the *children* of a node $w \in \mathrm{nodes}(\mathcal{T})$ are $\mathrm{childs}_{\mathcal{T}}(w) \stackrel{\text{def}}{=} \{wi \in \mathrm{nodes}(\mathcal{T}) \mid i \in [1, \kappa]\}$ and the *parent* of a node $wi \in \mathrm{nodes}(\mathcal{T})$, where $i \in [1, \kappa]$, is w. The *leaves* of $\mathcal{T}$ are $\mathrm{leaves}(\mathcal{T}) \stackrel{\text{def}}{=} \{w \in \mathrm{nodes}(\mathcal{T}) \mid w \cdot 1 \notin \mathrm{nodes}(\mathcal{T})\}$. The *subtree* of $\mathcal{T}$ rooted at w is defined by $\mathrm{nodes}(\mathcal{T}{\downarrow_w}) \stackrel{\text{def}}{=} \{w' \mid ww' \in \mathrm{nodes}(\mathcal{T})\}$ and $\mathcal{T}{\downarrow_w}\,(w') \stackrel{\text{def}}{=} \mathcal{T}(ww')$, for all $w' \in \mathrm{nodes}(\mathcal{T}{\downarrow_w})$. A *prefix* of a tree $\mathcal{T}$ is the restriction of $\mathcal{T}$ to a complete prefix-closed subset of $\mathrm{nodes}(\mathcal{T})$. We say that $\mathcal{T}$ is finite if $\mathrm{nodes}(\mathcal{T})$ is finite.

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

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

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 Δ.

Definition 6. *Given a CL formula ϕ, a* rewriting tree for ϕ *is a tree $\mathcal{T}$ with labels from Δ, such that $\mathcal{T}(\epsilon) = \big(\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \phi\big)$, for some predicate symbol $\mathcal{A}$ and, for each $w \in \mathrm{nodes}(\mathcal{T})$ such that $\mathcal{T}(w) = \big(\mathcal{A}_0(x_1, \ldots, x_{\#(\mathcal{A}_0)}) \leftarrow \psi_0\big)$, the following hold:*

1. *for all $i \in [1, \#_{\mathsf{pred}}(\psi_0)]$, if $\mathsf{pred}_i(\psi_0) = \mathcal{A}_i(y_1^i, \ldots, y_{\#(\mathcal{A}_i)}^i)$, then $wi \in \mathrm{nodes}(\mathcal{T})$ and $\mathcal{T}(wi) = \big(\mathcal{A}_i(x_1, \ldots, x_{\#(\mathcal{A}_i)}) \leftarrow \psi_i\big)$ is a rule from Δ,*
2. *$wi \notin \mathrm{nodes}(\mathcal{T})$, for all $i > \#_{\mathsf{pred}}(\psi_i)$.*

We denote by $\mathbb{T}(\phi)$ the set of rewriting trees for ϕ.

By slight abuse of notation, we write $\phi[\psi_1/\varphi_1 \ldots \psi_n/\varphi_n]$ for the result of replacing each subformula ψ_i of ϕ by the formula φ_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:

$$\mathfrak{F}(\mathcal{T}) \stackrel{\mathsf{def}}{=} \begin{cases} \phi, \text{ if } \mathrm{nodes}(\mathcal{T}) = \{\epsilon\} \\ \phi\left[\mathcal{A}_1(\mathbf{y}_1)/\mathfrak{F}(\mathcal{T}{\downarrow_1})[\mathbf{x}_1/\mathbf{y}_1] \ \ldots \ \mathcal{A}_n(\mathbf{y}_n)/\mathfrak{F}(\mathcal{T}{\downarrow_n})[\mathbf{x}_n/\mathbf{y}_n]\right], \\ \text{if } n = \#_{\mathsf{pred}}(\phi) \geq 1 \text{ and } \mathsf{pred}_i(\phi) = \mathcal{A}_i(\mathbf{y}_i), \text{ for all } i \in [1, n] \end{cases}$$

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 $\mathfrak{M}(\mathcal{T}) \stackrel{\mathsf{def}}{=} \eta$ denote the matrix of the characteristic formula $\mathfrak{F}(\mathcal{T})$ i.e., the largest quantifier-free formula of $\mathfrak{F}(\mathcal{T}) = \exists x_1 \ldots \exists x_n \ . \ \eta$ 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 $\Sigma = (\{\mathsf{N}, \mathsf{L}\}, \{\mathsf{R}, \mathsf{I}\}, \mathfrak{P})$ be a signature, where $\mathfrak{P}(\mathsf{N}) = \{req, reply\}$, $\mathfrak{P}(\mathsf{L}) = \{reply, in, out\}$, $\mathfrak{P}(\mathsf{R}) = \langle req, reply, reply\rangle$ and $\mathfrak{P}(\mathsf{I}) = \langle out, in\rangle$. Consider a behavior map β, such that $states_\beta(\mathsf{N}) = \{q_0, q_1\}$ and $states_\beta(\mathsf{L}) = \{s_0, s_1, s_2\}$. The SID below defines tree-shaped architectures, whose leaves are*

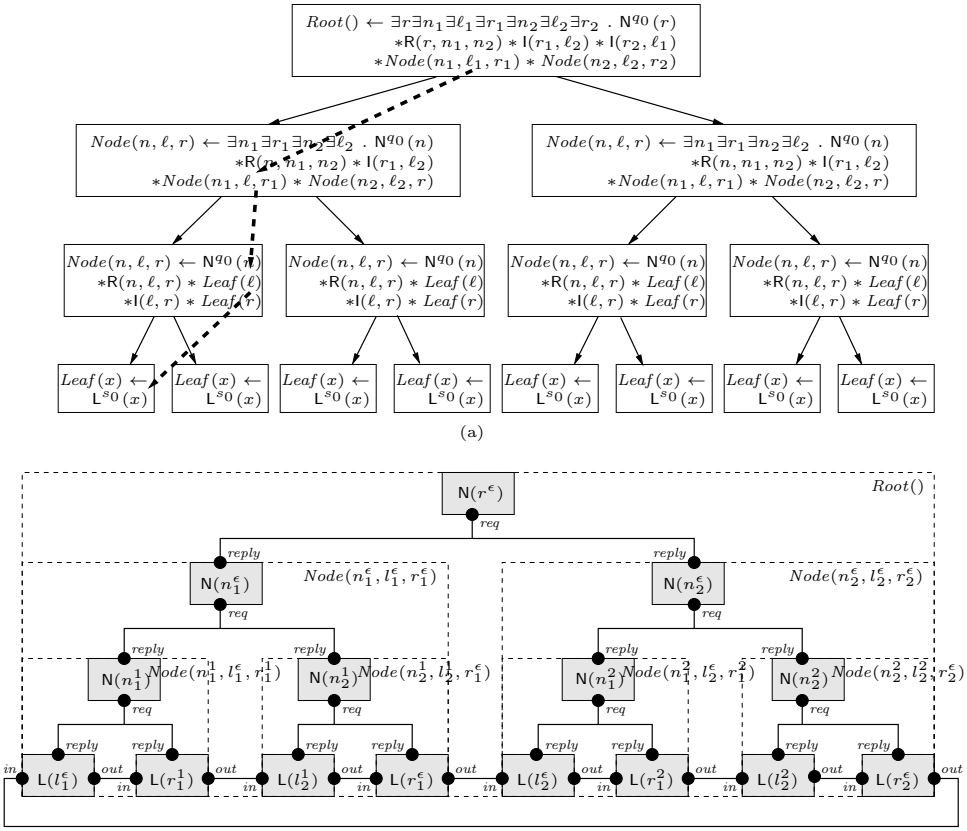

Figure 5: Trees with Leaves Linked in a Token Ring

connected in a token ring:

$$Root() \leftarrow \exists r \exists n_1 \exists \ell_1 \exists r_1 \exists n_2 \exists \ell_2 \exists r_2 \ . \ \mathsf{N}^{q_0}(r) * \mathsf{R}(r, n_1, n_2) * \mathsf{I}(r_1, \ell_2) * \mathsf{I}(r_2, \ell_1) * Node(n_1, \ell_1, r_1) * Node(n_2, \ell_2, r_2) \quad (7)$$

$$Node(n, \ell, r) \leftarrow \exists n_1 \exists r_1 \exists n_2 \exists \ell_2 \ . \ \mathsf{N}^{q_0}(n) * \mathsf{R}(n, n_1, n_2) * \mathsf{I}(r_1, \ell_2) * Node(n_1, \ell, r_1) * Node(n_2, \ell_2, r) \quad (8)$$

$$Node(n, \ell, r) \leftarrow \mathsf{N}^{q_0}(n) * \mathsf{R}(n, \ell, r) * Leaf(\ell) * \mathsf{I}(\ell, r) * Leaf(r) \quad (9)$$

$$Leaf(x) \leftarrow \mathsf{L}^{s_0}(x) \quad (10)$$

Figure 5(a) shows an unfolding tree for $\phi = \exists r \exists n_1 \exists \ell_1 \exists r_1 \exists n_2 \exists \ell_2 \exists r_2 \ . \ \mathsf{N}^{q_0}(r) * \mathsf{R}(r, n_1, n_2) * \mathsf{I}(r_1, \ell_2) * \mathsf{I}(r_2, \ell_1) * Node(n_1, \ell_1, r_1) * Node(n_2, \ell_2, r_2)$*, 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^ϵ, n_1^ϵ *and* n_2^ϵ *are introduced at the root, whereas* $n_1^1, n_2^1, \ell_2^1, r_1^1$ $(n_1^2, n_2^2, \ell_2^2, r_1^2)$ *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 Δ. In general, such rewriting trees are not unique, as different rewriting trees can have equivalent characteristic formulæ.

Lemma 1. *For each* CL *sentence* ϕ*, we have* $[\![\phi]\!] = \bigcup_{\mathcal{T} \in \mathbb{T}(\phi)} [\![\mathfrak{F}(\mathcal{T})]\!]$.

Proof. We prove a slightly more general statement: for each formula ϕ, each rule $\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \phi$ from Δ, each configuration (α, m) and each store $\mathfrak{s}$:

$$(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \phi \iff (\alpha, \mathrm{m}) \models^{\mathfrak{s}} \mathfrak{F}(\mathcal{T}), \text{ for some } \mathcal{T} \in \mathbb{T}(\phi), \text{ such that } \mathcal{T}(\epsilon) = \big(\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \phi\big)$$

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

- $\phi \in \{\mathsf{emp}, \mathsf{C}^q(x), \mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})\}$: we define $\mathcal{T}$ by $\mathrm{nodes}(\mathcal{T}) \stackrel{\text{def}}{=} \{\epsilon\}$ and $\mathcal{T}(\epsilon) \stackrel{\text{def}}{=} \big(\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \phi\big)$. By Def. 6, we have $\mathcal{T} \in \mathbb{T}(\phi)$ and $\mathfrak{F}(\mathcal{T}) = \phi$, thus $(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \mathfrak{F}(\mathcal{T})$.
- $\phi = \exists y_1 \ldots \exists y_n \,.\, \phi_0 * \circledast_{i=1}^{k} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)$, where ϕ_0 is quantifier- and predicate-free: since $(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \exists y_1 \ldots \exists y_n \,.\, \phi_0 * \circledast_{i=1}^{k} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)$, there exists a store $\mathfrak{s}'$ that agrees with $\mathfrak{s}$ over $\mathbb{X}^1 \setminus \{y_1, \ldots, y_n\}$ and configurations (α_i, m_i), such that $(\alpha, \mathrm{m}) = (\alpha_0, \mathrm{m}_0) \bullet \ldots \bullet (\alpha_k, \mathrm{m}_k)$, $(\alpha_0, \mathrm{m}_0) \models^{\mathfrak{s}'} \phi_0$ and $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}'} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)$, for all $i \in [1, k]$. For each $i \in [1, k]$, since $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}'} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)$, there exists $\mathcal{A}_i(x_1, \ldots, x_{\#(\mathcal{A}_i)}) \leftarrow \psi_i$ in Δ, such that $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}'} \psi_i[x_1/z_1^i, \ldots, x_{\#(\mathcal{A}_i)}/z_{\#(\mathcal{A}_i)}^i]$, by the definition of $\models$. Let $\mathfrak{s}_i' \stackrel{\text{def}}{=} \mathfrak{s}' \circ [x_1/z_1^i, \ldots, x_{\#(\mathcal{A}_i)}/z_{\#(\mathcal{A}_i)}^i]$ be a store. We have $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}_i'} \psi_i$ and, by the induction hypothesis, there exists a rewriting tree $\mathcal{T}_i \in \mathbb{T}(\psi_i)$, such that $\mathcal{T}_i(\epsilon) = \big(\mathcal{A}_i(x_1, \ldots, x_{\#(\mathcal{A}_i)}) \leftarrow \psi_i\big)$ and $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}_i'} \mathfrak{F}(\mathcal{T}_i)$, hence $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}'} \mathfrak{F}(\mathcal{T}_i)[x_1/z_1^i, \ldots, x_{\#(\mathcal{A}_i)}/z_{\#(\mathcal{A}_i)}^i]$. We define the rewriting tree $\mathcal{T}$ as:
  - $\mathrm{nodes}(\mathcal{T}) = \{\epsilon\} \cup \bigcup_{i=1}^{k} i \cdot \mathrm{nodes}(\mathcal{T}_i)$,
  - $\mathcal{T}(\epsilon) = \mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \phi$,
  - $\mathcal{T}(iw) = \mathcal{T}_i(w)$, for each $i \in [1, k]$ and each $w \in \mathrm{nodes}(\mathcal{T}_i)$.

  We obtain $\mathfrak{F}(\mathcal{T}) = \exists y_1 \ldots \exists y_n \,.\, \phi_0 * \circledast_{i=1}^{k} \mathfrak{F}(\mathcal{T}_i)[x_1/z_1^i, \ldots, x_{\#(\mathcal{A}_i)}/z_{\#(\mathcal{A}_i)}^i]$, 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 $\mathcal{T}(\epsilon) = \big(\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \phi\big)$, by distinguishing the following cases:

- $\mathrm{nodes}(\mathcal{T}) = \{\epsilon\}$: since $\mathcal{T}(\epsilon) = \big(\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \phi\big)$, we have $\mathfrak{F}(\mathcal{T}) = \phi$, hence $(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \phi$.

- $\mathrm{childs}_{\mathcal{T}}(\epsilon) = \{1, \ldots, k\}$, for some $k \geq 1$: assume w.l.o.g. that $\phi = \exists y_1 \ldots \exists y_n \ . \ \phi_0 * \mathbin{\raisebox{0.25ex}{\scalebox{0.8}{$\bigstar$}}}_{i=1}^{k} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)$, where ϕ_0 is a quantifier- and predicate-free formula. By the definition of characteristic formulæ, we have $\mathfrak{F}(\mathcal{T}) = \exists y_1 \ldots \exists y_n \ . \ \phi_0 * \mathbin{\raisebox{0.25ex}{\scalebox{0.8}{$\bigstar$}}}_{i=1}^{k} \mathfrak{F}(\mathcal{T}{\downarrow_i})[x_1/z_1^i, \ldots, x_{\#(\mathcal{A}_i)}/z_{\#(\mathcal{A}_i)}^i]$, where $\mathcal{T}(i) = \big(\mathcal{A}_i(x_1, \ldots, x_{\#(\mathcal{A}_i)}) \leftarrow \psi_i\big)$, hence $\mathcal{T}{\downarrow_i} \in \mathbb{T}(\psi_i)$. Since $(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \mathfrak{F}(\mathcal{T})$, there exists a store $\mathfrak{s}'$ that agrees with $\mathfrak{s}$ over $\mathbb{X}^1 \setminus \{y_1, \ldots, y_n\}$ and configurations $(\alpha_0, \mathrm{m}_0), \ldots, (\alpha_k, \mathrm{m}_k)$, such that $(\alpha_0, \mathrm{m}_0) \models^{\mathfrak{s}'} \phi_0$ and $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}'} \mathfrak{F}(\mathcal{T}{\downarrow_i})[x_1/z_1^i, \ldots, x_{\#(\mathcal{A}_i)}/z_{\#(\mathcal{A}_i)}^i]$, for all $i \in [1, m]$. Consider an arbitrary index $i \in [1, m]$ and let $\mathfrak{s}_i' \stackrel{\text{def}}{=} \mathfrak{s}' \circ [x_1/z_1^i, \ldots, x_{\#(\mathcal{A}_i)}/z_{\#(\mathcal{A}_i)}^i]$, hence $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}_i'} \mathfrak{F}(\mathcal{T}{\downarrow_i})$. By the inductive hypothesis, we have $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}_i'} \psi_i$, hence $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}_i'} \mathcal{A}_i(x_1, \ldots, x_{\#(\mathcal{A}_i)})$ and $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}'} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)$ follows. Because the above holds for all $i \in [1, k]$, we conclude that $(\alpha, \mathrm{m}) \models^{\mathfrak{s}} \phi$. $\square$

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 κ. 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* $(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}})$ 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 Δ is of one of the forms:*

$$\mathcal{A}(x_1) \leftarrow \phi, \text{ such that } \phi \neq \mathsf{emp} \tag{I}$$

$$\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \exists y_1 \ldots \exists y_n \ . \ \phi \ * \ \psi \ * \mathbin{\raisebox{0.25ex}{\scalebox{0.8}{$\bigstar$}}}_{i=1}^{p} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i) \tag{II}$$

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

- $\phi \in \{\mathsf{C}^q(x_1) \mid \mathsf{C} \in \mathfrak{C}, \ q \in \mathit{states}_\beta(\mathsf{C})\} \cup \{\mathsf{emp}\}$,
- *ψ is a separating conjunction of interaction atoms,*
- $\bigcup_{i=1}^{p} \{z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i\} = \big(\{x_1, \ldots, x_{\#(\mathcal{A})}\} \setminus \mathrm{fv}(\phi)\big) \cup \{y_1, \ldots, y_n\}$, *and*
- $\{z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i\} \cap \{z_1^j, \ldots, z_{\#(\mathcal{A}_j)}^j\} = \emptyset$, *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 $\mathit{Chain}_{n,t}$ 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 ϕ be a* SL *formula and $\mathcal{T} \in \mathbb{T}(\phi)$ be a rewriting tree for ϕ. For each variable x, that occurs free or existentially quantified in $\mathfrak{F}(\mathcal{T})$, there exists exactly one node $w \in \mathrm{nodes}(\mathcal{T})$ and one component atom $\mathsf{C}^q(x)$ 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 $w \in \mathrm{nodes}(\mathcal{T}) \setminus \{\epsilon\}$ 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 $\mathrm{nodes}(\mathcal{T}) = \{\epsilon\}$, the only possibility is that $\mathcal{T}(\epsilon)$ is a rule of the form (I) and the proof is immediate. For the inductive step, let $\mathcal{T}(\epsilon)$ be the following rule:

$$\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \exists y_1 \ldots \exists y_n \,.\, \phi \;*\; \psi \;* \circledast_{i=1}^{p} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)$$

If ϕ is of the form $\mathsf{C}^q(x_1)$ and $x = x_1$, then ϵ is the unique node such that x occurs in a component atom. Else, $x \in \{x_1, \ldots, x_{\#(\mathcal{A})}\} \cup \{y_1, \ldots, y_n\} \setminus \mathrm{fv}(\phi)$, thus $x \in \{z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i\}$, for exactly one $i \in [1, p]$, by Assumption 2. In this case the inductive hypothesis applies, thus x occurs in exactly one component atom $\mathsf{C}^q(x)$ in $\mathcal{T}(w)$, for exactly one $w \in \mathrm{nodes}(\mathcal{T}{\downarrow_i})$. □

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 $w \in \mathrm{nodes}(\mathcal{T})$, such that x occurs in a single component atom $\mathsf{C}^q(x)$ in $\mathcal{T}(w)$ (Lemma 2). Formally, let $\phi = \exists y_1 \ldots \exists y_n \,.\, \psi \;* \circledast_{i=1}^{p} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)$ be a formula, where ψ is quantifier- and predicate-free, and define, for each rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, a store $\mathfrak{s}_{\mathcal{T}}^{\epsilon} : \mathrm{fv}(\psi) \cup \bigcup_{i=1}^{p} \{z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i\} \to \mathrm{nodes}(\mathcal{T})$, as follows:

$$\mathfrak{s}_{\mathcal{T}}^{\epsilon}(x) \stackrel{\mathsf{def}}{=} \begin{cases} \epsilon & \text{if } x \text{ occurs in a component atom from } \psi \\ i \cdot \mathfrak{s}_{\mathcal{T}{\downarrow_i}}^{\epsilon}(x_j) & \text{if } x = z_j^i, \text{ for some } j \in [1, \#(\mathcal{A}_i)], \text{ where} \\ & \mathcal{T}(i) = \big(\mathcal{A}_i(x_1, \ldots, x_{\#(\mathcal{A}_i)}) \leftarrow \varphi_i\big), \text{ for all } i \in [1, p] \end{cases}$$

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 $\psi \;* \circledast_{i=1}^{p} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)$ of ϕ.

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 $\mathfrak{s}_{\mathcal{T}} : \mathrm{fv}(\mathfrak{M}(\mathcal{T})) \to \mathrm{nodes}(\mathcal{T})$ mapping all variables (free or existentially quantified) that occur in the characteristic formula $\mathfrak{F}(\mathcal{T})$:

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

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 α-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 $(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}})$, defined as follows:*

- $\alpha_{\mathcal{T}}(\mathsf{C}) \stackrel{\text{def}}{=} \{\mathfrak{s}_{\mathcal{T}}(x) \mid \mathsf{C}^q(x) \text{ occurs in } \mathfrak{M}(\mathcal{T})\}$,
- $\alpha_{\mathcal{T}}(\mathsf{I}) \stackrel{\text{def}}{=} \{\langle \mathfrak{s}_{\mathcal{T}}(x_1), \ldots, \mathfrak{s}_{\mathcal{T}}(x_{\#(\mathsf{I})})\rangle \mid \mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})}) \text{ occurs in } \mathfrak{M}(\mathcal{T})\}$, *and*
- $\mathrm{m}_{\mathcal{T}} \stackrel{\text{def}}{=} \{q[\mathfrak{s}_{\mathcal{T}}(x)] \mid \mathsf{C}^q(x) \text{ occurs in } \mathfrak{M}(\mathcal{T})\}$.

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

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

- $\mathrm{nodes}(\mathcal{T}) = \{\epsilon\}$ and $\mathcal{T}(\epsilon) = (\mathcal{A}(x) \leftarrow \mathsf{C}^q(x))$ is a rule of type (I): then $\alpha_{\mathcal{T}}(\mathsf{C}) = \{\mathfrak{s}_{\mathcal{T}}(x)\} = \{\epsilon\}$, $\mathrm{m}_{\mathcal{T}} = \{q[\mathfrak{s}_{\mathcal{T}}(x)]\} = \{q[\epsilon]\}$, $\alpha_{\mathcal{T}}(\mathsf{C}') = \emptyset$, for all $\mathsf{C}' \in \mathfrak{C} \setminus \{\mathsf{C}\}$ and $\alpha_{\mathcal{T}}(\mathsf{I}) = \emptyset$, for all $\mathsf{I} \in \mathfrak{I}$, thus $(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}) \models^{\mathfrak{s}_{\mathcal{T}}} \mathsf{C}^q(x)$.
- $\mathrm{nodes}(\mathcal{T}) \neq \{\epsilon\}$ and $\mathcal{T}(\epsilon) = \big(\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \exists y_1 \ldots \exists y_n \,.\, \phi * \psi * \mathop{\ast}_{i=1}^{p} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)\big)$ is a rule of type (II). By the definition of $\mathfrak{s}_{\mathcal{T}}$, we have $\mathfrak{s}_{\mathcal{T}}(x) = i \cdot \mathfrak{s}_{\mathcal{T}_{\downarrow_i}}(x)$, for all $x \in \mathrm{fv}(\mathfrak{M}(\mathcal{T}_{\downarrow_i}))$ and $i \in [1, p]$. We define the structures $(\alpha_0, \mathrm{m}_0), (\alpha_1, \mathrm{m}_1), \ldots, (\alpha_p, \mathrm{m}_p)$ as follows:
  - $\alpha_0(\mathsf{C}) \stackrel{\text{def}}{=} \{\mathfrak{s}_{\mathcal{T}}(x)\} = \{\epsilon\}$ and $\mathrm{m}_0 \stackrel{\text{def}}{=} \{q[\mathfrak{s}_{\mathcal{T}}(x)]\} = \{q[\epsilon]\}$ if $\phi = \mathsf{C}^q(x_1)$, else $\alpha_0(\mathsf{C}) = \emptyset$ and $\mathrm{m}_0 = \emptyset$ and, moreover, $\alpha_0(\mathsf{I}) \stackrel{\text{def}}{=} \{\langle \mathfrak{s}_{\mathcal{T}}(x_1), \ldots, \mathfrak{s}_{\mathcal{T}}(x_{\#(\mathsf{I})})\rangle \mid \mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})}) \text{ occurs in } \psi\}$, thus we have $(\alpha_0, \mathrm{m}_0) \models^{\mathfrak{s}_{\mathcal{T}}} \phi * \psi$.
  - $\alpha_i(\mathsf{C}) \stackrel{\text{def}}{=} \{\mathfrak{s}_{\mathcal{T}}(x) \mid \mathsf{C}^q(x) \text{ occurs in } \mathfrak{M}(\mathcal{T}_{\downarrow_i})\}$, $\alpha_i(\mathsf{I}) \stackrel{\text{def}}{=} \{\langle \mathfrak{s}_{\mathcal{T}}(x_1), \ldots, \mathfrak{s}_{\mathcal{T}}(x_{\#(\mathsf{I})})\rangle \mid \mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})}) \text{ occurs in } \mathfrak{M}(\mathcal{T}_{\downarrow_i})\}$ and $\mathrm{m}_i \stackrel{\text{def}}{=} \{q[\mathfrak{s}_{\mathcal{T}}(x)] \mid \mathsf{C}^q(x) \text{ occurs in } \mathfrak{M}(\mathcal{T}_{\downarrow_i})\}$, for each $i \in [1, p]$. Since $\mathfrak{s}_{\mathcal{T}}(x) = i \cdot \mathfrak{s}_{\mathcal{T}_{\downarrow_i}}(x)$, for all $x \in \eta_i$ and $i \in [1, p]$, we have $\alpha_i(\mathsf{C}) = \{i \cdot u \mid u \in \alpha_{\mathcal{T}_{\downarrow_i}}(\mathsf{C})\}$, for all $\mathsf{C} \in \mathfrak{C}$, $\mathrm{m}_i = \{q[i \cdot u] \mid q[u] \in \mathrm{m}_{\mathcal{T}_{\downarrow_i}}\}$ and $\alpha_i(\mathsf{I}) = \{\langle i \cdot u_1, \ldots, i \cdot u_{\#(\mathsf{I})}\rangle \mid \langle u_1, \ldots, u_{\#(\mathsf{I})}\rangle \in \alpha_{\mathcal{T}_{\downarrow_i}}(\mathsf{I})\}$, for all $\mathsf{I} \in \mathfrak{I}$. By the inductive hypothesis, we have $(\alpha_{\mathcal{T}_{\downarrow_i}}, \mathrm{m}_{\mathcal{T}_{\downarrow_i}}) \models^{\mathfrak{s}_{\mathcal{T}_i}} \eta_i$, thus $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}_{\mathcal{T}}} \eta_i$, 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 $(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}) = (\alpha_0, \mathrm{m}_0) \bullet \ldots \bullet (\alpha_p, \mathrm{m}_p)$. Consequently, we obtain $(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}) \models^{\mathfrak{s}_{\mathcal{T}}} \phi * \psi * \mathop{\ast}_{i=1}^{p} \mathcal{A}_i(z_1^i, \ldots, z_{\#(\mathcal{A}_i)}^i)$, 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:*

$$\begin{array}{llll} \mathfrak{s}_{\mathcal{T}}(r^{\epsilon}) = \epsilon & \mathfrak{s}_{\mathcal{T}}(n_1^{\epsilon}) = 1 & \mathfrak{s}_{\mathcal{T}}(n_2^{\epsilon}) = 2 & \\ \mathfrak{s}_{\mathcal{T}}(n_1^1) = 11 & \mathfrak{s}_{\mathcal{T}}(n_2^1) = 12 & \mathfrak{s}_{\mathcal{T}}(n_1^2) = 21 & \mathfrak{s}_{\mathcal{T}}(n_2^2) = 22 \\ \mathfrak{s}_{\mathcal{T}}(\ell_1^{\epsilon}) = 111 & \mathfrak{s}_{\mathcal{T}}(r_1^1) = 112 & \mathfrak{s}_{\mathcal{T}}(\ell_2^1) = 121 & \mathfrak{s}_{\mathcal{T}}(r_1^{\epsilon}) = 122 \\ \mathfrak{s}_{\mathcal{T}}(\ell_2^{\epsilon}) = 211 & \mathfrak{s}_{\mathcal{T}}(r_1^2) = 212 & \mathfrak{s}_{\mathcal{T}}(\ell_2^2) = 221 & \mathfrak{s}_{\mathcal{T}}(r_2^{\epsilon}) = 222 \end{array}$$

The configuration $(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}})$ corresponding to the rewriting tree $\mathcal{T}$ from Fig-

ure 5(a) is given below:

$$\begin{aligned}
\alpha_{\mathcal{T}}(\mathsf{N}) &= \{\epsilon, 1, 2, 11, 12, 21, 22\} \\
\alpha_{\mathcal{T}}(\mathsf{L}) &= \{111, 112, 121, 122, 211, 212, 221, 222\} \\
\alpha_{\mathcal{T}}(\mathsf{R}) &= \{\langle \epsilon, 1, 2\rangle, \langle 1, 11, 12\rangle, \langle 11, 111, 112\rangle, \langle 12, 121, 122\rangle, \langle 21, 211, 212\rangle, \\
&\qquad \langle 22, 221, 222\rangle\} \\
\alpha_{\mathcal{T}}(\mathsf{I}) &= \{\langle 111, 112\rangle, \langle 112, 121\rangle, \langle 121, 122\rangle, \langle 211, 212\rangle, \langle 212, 221\rangle, \langle 221, 222\rangle, \\
&\qquad \langle 222, 111\rangle\} \\
\mathrm{m}_{\mathcal{T}} &= \{q_0[\epsilon], q_0[1], q_0[2], q_0[11], q_0[12], q_0[21], q_0[22], s_0[111], s_0[112], \\
&\qquad s_0[121], s_0[122], s_0[211], s_0[212], s_0[221], s_0[222]\}
\end{aligned}$$ ■

5.3. Symmetry Reduction

As shown in Lemma 1, each model of a sentence is a model of the characteristic formula of some rewriting tree for that sentence. In this section, we prove a symmetry property of the models of a characteristic formula corresponding to a given rewriting tree, that makes them indistinguishable from the point of view of the decision problems considered previously (§4). Let us fix a signature Σ, with component types $\mathfrak{C} = \{\mathsf{C}_1, \ldots, \mathsf{C}_N\}$ and interaction types $\mathfrak{I} = \{\mathsf{I}_1, \ldots, \mathsf{I}_M\}$, and a behavior map β, in the rest of this section.

Intuitively, two architectures are symmetric if they differ only by a renaming of indices used to interpret the component and interaction types. For instance, the architectures α_1 and α_2, where $\alpha_1(\mathsf{C}_1) = \alpha_1(\mathsf{C}_2) = \{1\}$, $\alpha_1(\mathsf{I}_1) = \{\langle 1, 1\rangle\}$ and $\alpha_2(\mathsf{C}_1) = \{1\}$, $\alpha_2(\mathsf{C}_2) = \{2\}$, $\alpha_2(\mathsf{I}_1) = \{\langle 1, 2\rangle\}$ 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 case[4] in the literature [13], because the sets $\alpha_1(\mathsf{C}_1) \cup \alpha_1(\mathsf{C}_2)$ and $\alpha_2(\mathsf{C}_1) \cup \alpha_2(\mathsf{C}_2)$ have different cardinalities. For this reason, our definition of symmetry considers one permutation per component type.

Formally, given an architecture α over Σ and a tuple of bijections $\mathbf{f} = \langle f_1, \ldots, f_N\rangle$, where each $f_i : \mathbb{U} \to \mathbb{U}$ renames the indices of the component type C_i, for all $i \in [1, N]$, we define:

$$\begin{aligned}
(\mathbf{f}(\alpha))(\mathsf{C}_i) &\stackrel{\text{def}}{=} f_i(\alpha(\mathsf{C}_i)), \text{ for all } i \in [1, N] \\
(\mathbf{f}(\alpha))(\mathsf{I}) &\stackrel{\text{def}}{=} \{\langle f_{i_1}(u_1), \ldots, f_{i_{\#(\mathsf{I})}}(u_{\#(\mathsf{I})})\rangle \mid \langle u_1, \ldots, u_{\#(\mathsf{I})}\rangle \in \alpha(\mathsf{I}), \\
&\qquad\qquad \forall k \in [1, \#(\mathsf{I})] \;.\; \mathit{comp}(\langle \mathfrak{P}(\mathsf{I})\rangle_k) = \mathsf{C}_{i_k}\}
\end{aligned}$$

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

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

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

[4] Typically, global permutations suffice when only one component type is considered.

Definition 8. *Two configurations are* symmetric*, denoted* $(\alpha_1, \mathrm{m}_1) \sim (\alpha_2, \mathrm{m}_2)$*, if and only if there exists a tuple of bijections* $\mathbf{f} = \langle f_1, \ldots, f_N \rangle$*, where* $f_i : \mathbb{U} \to \mathbb{U}$*, for all* $i \in [1, N]$*, such that* $\mathbf{f}(\alpha_1) = \alpha_2$ *and* $\mathbf{f}(\mathrm{m}_1) = \mathrm{m}_2$*.*

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 $(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}})$. 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* α *is* tight *if and only if, for each interaction* $\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}]$ *from* α *and each* $k \in [1, \#(\mathsf{I})]$*, we have* $u_k \in \alpha(\mathsf{C})$ *where* $\mathsf{C} = comp(\langle \mathfrak{P}(\mathsf{I}) \rangle_k))$*, that is, the unique component type such that* $\langle \mathfrak{P}(\mathsf{I}) \rangle_k \in \mathfrak{P}(\mathsf{C})$*.*

Note that the interactions from an architecture α having unconnected ports cannot fire in the Petri net $\mathsf{N}(\Sigma, \alpha, \beta)$ (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* $\lambda : \mathbb{A} \to \bigcup_{i \geq 1} \mathfrak{C}^i$*, that associates each predicate symbol* $\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})})$ *of non-zero arity with a tuple of component types of length* $\#(\mathcal{A}) \geq 1$*. A formula* ϕ *is* tight for a profile λ *if and only if it contains, for each interaction atom* $\mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})$ *and each* $k \in [1, \#(\mathsf{I})]$*:*

- *a component atom* $\mathsf{C}^q(x_k)$*, such that* $\langle \mathfrak{P}(\mathsf{I}) \rangle_k \in \mathfrak{P}(\mathsf{C})$*, or*
- *a predicate atom* $\mathcal{A}(y_1, \ldots, y_{\#(\mathcal{A})})$*, such that* $x_k = y_\ell$ *and* $\langle \mathfrak{P}(\mathsf{I}) \rangle_k \in \mathfrak{P}(\langle \lambda(\mathcal{A}) \rangle_\ell)$*, for some* $\ell \in [1, \#(\mathcal{A})]$*.*

A SID Δ *is* tight *if and only if there exists a profile* λ_Δ*, such that, for each rule* $\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \phi$ *from* Δ*, the formula* ϕ *is tight for* λ_Δ *and contains, for each* $k \in [1, \#(\mathcal{A})]$*:*

- *a component atom* $\mathsf{C}^q(x_k)$*, such that* $\langle \lambda_\Delta(\mathcal{A}) \rangle_k = \mathsf{C}$*, or*
- *a predicate atom* $\mathcal{B}(y_1, \ldots, y_{\#(\mathcal{A})})$*, such that* $x_k = y_\ell$ *and* $\langle \lambda_\Delta(\mathcal{A}) \rangle_k = \langle \lambda_\Delta(\mathcal{B}) \rangle_\ell$*, for some* $\ell \in [1, \#(\mathcal{B})]$*.*

A formula ϕ*, interpreted over a tight SID* Δ*, is* tight *if and only if it is tight for the profile* λ_Δ*.*

For instance, one can check that the predicate atoms $Chain_{n,t}(x, y)$ are tight, by taking the profile $\lambda_\Delta(Chain_{n,t}) = \langle \mathsf{S}, \mathsf{S} \rangle$, for all constants $n, t \geq 0$ (Example 5). Analogously, the predicate atoms $Node(n, \ell, r)$ are tight, by taking the profile $\lambda_\Delta(Node) = \langle \mathsf{N}, \mathsf{L}, \mathsf{L} \rangle$ and $\lambda_\Delta(Leaf) = \langle \mathsf{L} \rangle$ (Example 6). The above conditions guarantee the tightness of architectures described by tight formulæ:

Lemma 4. *For each model* (α, m) *of a tight sentence, the architecture* α *is tight.*

Proof. Let ϕ be a tight sentence, such that $(\alpha, \mathrm{m}) \models \phi$. 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 $\mathsf{I}(z_1, \ldots, z_{\#(\mathsf{I})})$ be an interaction atom from $\mathfrak{F}(\mathcal{T})$ and $k \in [1, \#(\mathsf{I})]$. Also, let $w \in \mathrm{nodes}(\mathcal{T})$ be the node where this interaction atom was introduced, such that $\mathcal{T}(w) = \big(\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \varphi\big)$ and $\mathsf{I}(y_1, \ldots, y_{\#(\mathsf{I})})$ occurs in φ, where $y_1, \ldots, y_{\#(\mathsf{I})}$ are substituted by $z_1, \ldots, z_{\#(\mathcal{A})}$, respectively, during the construction of $\mathfrak{F}(\mathcal{T})$. Since the SID Δ, that interprets the formula ϕ, is tight, the rule body φ is tight, hence one of the following holds:

- there exists a component atom $\mathsf{C}^q(y_k)$ in φ, such that $\langle \mathfrak{P}(\mathsf{I}) \rangle_k = \mathsf{C}$, or
- there exists a predicate atom $\mathcal{B}(\xi_1, \ldots, \xi_{\#(\mathcal{B})})$ in φ, such that $y_k = \xi_\ell$ and $\langle \mathfrak{P}(\mathsf{I}) \rangle_k = \langle \lambda_\Delta(\mathcal{B}) \rangle_\ell$, where λ_Δ is the profile from Def. 10. In this case, we apply induction to prove the existence of a component atom $\mathsf{C}^q(\xi_k)$ in $\mathfrak{F}(\mathcal{T}{\downarrow_i})$, such that $\langle \mathfrak{P}(\mathsf{I}) \rangle_k = \mathsf{C}$, where $i \in [1, \#_{\mathsf{pred}}(\varphi)]$ is the child of w corresponding to $\mathcal{B}(\xi_1, \ldots, \xi_{\#(\mathcal{B})})$ in $\mathcal{T}$.

In both cases, $\mathfrak{F}(\mathcal{T})$ contains a component atom $\mathsf{C}^q(z_k)$, such that $\langle \mathfrak{P}(\mathsf{I}) \rangle_k = \mathsf{C}$, hence (α, 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 ϕ, for each $\mathcal{T} \in \mathbb{T}(\phi)$, if $(\alpha, \mathrm{m}) \models \mathfrak{F}(\mathcal{T})$, then $(\alpha, \mathrm{m}) \sim (\alpha_\mathcal{T}, \mathrm{m}_\mathcal{T})$.*

Proof. Since $(\alpha_\mathcal{T}, \mathrm{m}_\mathcal{T}) \models \mathfrak{F}(\mathcal{T})$, by Lemma 3, it is sufficient to prove that $(\alpha_1, \mathrm{m}_1) \sim (\alpha_2, \mathrm{m}_2)$, for any two configurations (α_i, m_i), such that $(\alpha_i, \mathrm{m}_i) \models \mathfrak{F}(\mathcal{T})$. Let $\mathfrak{F}(\mathcal{T}) \stackrel{\mathsf{def}}{=} \exists x_1 \ldots \exists x_n \ . \ \eta$, where η is a quantifier- and predicate-free formula. Then $(\alpha_i, \mathrm{m}_i) \models \mathfrak{F}(\mathcal{T})$ if and only if there exist stores $\mathfrak{s}_i$, such that $(\alpha_i, \mathrm{m}_i) \models^{\mathfrak{s}_i} \eta$, for $i = 1, 2$. We define the sets $U_j^i \stackrel{\mathsf{def}}{=} \{\mathfrak{s}_i(x) \mid \mathsf{C}_j^q(x) \text{ occurs in } \eta\}$ and the bijections $f_j : U_j^1 \to U_j^2$ as $f_j(\mathfrak{s}_1(x)) = \mathfrak{s}_2(x)$, for all variables x, such that $\mathsf{C}_j^q(x)$ occurs in η. These bijections are extended to bijections $\overline{f}_j : \mathbb{U} \to \mathbb{U}$, by the following fact:

Fact 1. *Given finite sets $U_1, U_2 \subseteq \mathbb{U}$, such that $f : U_1 \to U_2$ is a bijection, there exists a bijection $\overline{f} : \mathbb{U} \to \mathbb{U}$, such that $\overline{f}(u) = f(u)$, for all $u \in U_1$.*

Proof. Since f is a bijection, we have $\|U_1\| = \|U_2\|$ and, since $\|U_i\| = \|U_i \setminus U_{3-i}\| + \|U_1 \cap U_2\|$, for $i = 1, 2$, we obtain $\|U_1 \setminus U_2\| = \|U_2 \setminus U_1\|$. Hence there exists a bijection $\overline{f} : \mathbb{U} \to \mathbb{U}$, such that $\overline{f}(u) = f(u)$, for all $u \in U_1$, $\overline{f}(U_2 \setminus U_1) = U_1 \setminus U_2$ and $\overline{f}(\mathbb{U} \setminus (U_1 \cup U_2)) = f(\mathbb{U} \setminus (U_1 \cup U_2))$. □

For $\mathbf{f} \stackrel{\mathsf{def}}{=} \langle \overline{f}_1, \ldots, \overline{f}_N \rangle$, we prove the following points:

- $\mathbf{f}(\alpha_1) = \alpha_2$: by the definition of $\overline{f}_j$, we have $\overline{f}_j(\alpha_1(\mathsf{C}_j)) = \alpha_2(\mathsf{C}_j)$, for all $j \in [1, N]$. Let $\mathsf{I} \in \mathfrak{I}$ and we prove $\alpha_2(\mathsf{I}) = \{\langle \overline{f}_{i_1}(u_1), \ldots, \overline{f}_{i_{\#(\mathsf{I})}}(u_{\#(\mathsf{I})}) \rangle \mid \langle u_1, \ldots, u_{\#(\mathsf{I})} \rangle \in \alpha_1(\mathsf{I}),\ \forall k \in [1, \#(\mathsf{I})] \ . \ comp(\langle \mathfrak{P}(\mathsf{I}) \rangle_k) = \mathsf{C}_{i_k}\}$. "$\subseteq$" Let $\langle u_1, \ldots, u_{\#(\mathsf{I})} \rangle \in \alpha_2(\mathsf{I})$. Then there exists an interaction atom $\mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})$ in η, such that $u_i = \mathfrak{s}_2(x_i)$, for $i \in [1, \#(\mathsf{I})]$. Since ϕ is a tight sentence and $(\alpha_2, \mathrm{m}_2) \models \phi$, by Lemma 4, the architecture α_2 is tight (Def. 9). Then, for each $k \in [1, \#(\mathsf{I})]$, we have $u_k \in \alpha_2(\mathsf{C}_{i_k})$, where $\mathsf{C}_{i_k} \in \mathfrak{C}$ is

the unique component type such that $\langle\mathfrak{P}(\mathsf{I})\rangle_k \in \mathfrak{P}(\mathsf{C}_{i_k})$. By the previous point, there exists $u'_k \in \alpha_1(\mathsf{C}_{i_k})$, such that $\overline{f}_{i_k}(u'_k) = u_k = \mathfrak{s}_2(x_k)$. By the definition of $\overline{f}_{i_k}$, we have $u'_k = \mathfrak{s}_1(x_k)$, hence $\langle u'_1, \ldots, u'_{\#(\mathsf{I})}\rangle \in \alpha_1(\mathsf{I})$. "$\supseteq$" Let $\langle u_1, \ldots, u_{\#(\mathsf{I})}\rangle \in \alpha_1(\mathsf{I})$, such that $comp(\langle\mathfrak{P}(\mathsf{I})\rangle_k) = \mathsf{C}_{i_k}$, for all $k \in [1, \#(\mathsf{I})]$. Then there exists an interaction atom $\mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})$ in η, such that $u_k = \mathfrak{s}_1(x_k)$, for all $k \in [1, \#(\mathsf{I})]$. Since ϕ is a tight sentence and $(\alpha_1, \mathrm{m}_1) \models \phi$, by Lemma 4, the architecture α_1 is tight, thus $\mathfrak{s}_1(x_k) = u_k \in \alpha_1(\mathsf{C}_{i_k})$. By the definition of $\overline{f}_{i_k}$, we have $\mathfrak{s}_2(x_k) = \overline{f}_{i_k}(u_k)$, for all $k \in [1, \#(\mathsf{I})]$, hence $\langle \overline{f}_{i_1}(u_1), \ldots, \overline{f}_{i_{\#(\mathsf{I})}}(u_{\#(\mathsf{I})})\rangle \in \alpha_2(\mathsf{I})$.

- $\mathbf{f}(\mathrm{m}_1) = \mathrm{m}_2$: we prove $\mathrm{m}_2 = \{q[\overline{f}_i(u)] \mid q[u] \in \mathrm{m}_1, q \in states_\beta(\mathsf{C}_i), i \in [1, N]\}$. "$\subseteq$" Let $q[u] \in \mathrm{m}_2$. Since $(\alpha_2, \mathrm{m}_2) \models^{\mathfrak{s}_2} \eta$, there exists a component atom $\mathsf{C}_i^q(x)$ in η, such that $u = \mathfrak{s}_2(x) \in \alpha_2(\mathsf{C}_i)$ and C_i is the unique component type, such that $q \in states_\beta(\mathsf{C}_i)$. By the definition of $\overline{f}_i$, we have $\mathfrak{s}_1(x) \in \alpha_1(\mathsf{C}_i)$ and $\mathfrak{s}_2(x) = \overline{f}_i(\mathfrak{s}_1(x))$. Moreover, since $(\alpha_1, \mathrm{m}_1) \models^{\mathfrak{s}_1} \eta$, we have $q[\mathfrak{s}_1(x)] \in \mathrm{m}_1$. "$\supseteq$" Let $q[u] \in \mathrm{m}_1$ and $q \in states_\beta(\mathsf{C}_i)$, for some $i \in [1, N]$. Since $(\alpha_1, \mathrm{m}_1) \models^{\mathfrak{s}_1} \eta$, there exists a component atom $\mathsf{C}^q(x)$ in η, such that $u = \mathfrak{s}_1(x)$. By the definition of $\overline{f}_i$, we have $\overline{f}_i(\mathfrak{s}_1(x)) = \mathfrak{s}_2(x)$, hence $q[\overline{f}_i(u)] = q[\mathfrak{s}_2(x)] \in \mathrm{m}_2$, since $(\alpha_2, \mathrm{m}_2) \models^{\mathfrak{s}_2} \eta$. □

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 α and a behavior β, over a signature $\Sigma = (\{\mathsf{C}_1, \ldots, \mathsf{C}_N\}, \mathfrak{I}, \mathfrak{P})$, for any tuple of bijections $\mathbf{f} = \langle f_1, \ldots, f_N\rangle$, where $f_i : \mathbb{U} \to \mathbb{U}$, for all $i \in [1, N]$, the following hold:*

1. $\mathrm{m}' \in Reach(\mathsf{N}(\Sigma, \alpha, \beta), \mathrm{m}) \iff \mathbf{f}(\mathrm{m}') \in Reach(\mathsf{N}(\Sigma, \mathbf{f}(\alpha), \beta), \mathbf{f}(\mathrm{m}))$,
2. $\mathrm{m} \in Dead(\mathsf{N}(\Sigma, \alpha, \beta)) \iff \mathbf{f}(\mathrm{m}) \in Dead(\mathsf{N}(\Sigma, \mathbf{f}(\alpha), \beta))$,

for any two markings m, m' of the Petri net $\mathsf{N}(\Sigma, \alpha, \beta)$.

Proof. By Def. 1, let $\mathsf{N}(\Sigma, \mathbf{f}(\alpha), \beta)$ be the Petri net $(S^{\mathbf{f}}, T^{\mathbf{f}}, E^{\mathbf{f}})$, where:

$$
\begin{aligned}
S^{\mathbf{f}} &= \bigcup\nolimits_{i=1}^{N} \{q[u] \mid q \in states(\mathsf{C}_i),\ u \in (\mathbf{f}(\alpha))(\mathsf{C}_i)\} \\
&= \bigcup\nolimits_{i=1}^{N} \{q[f_i(u)] \mid q \in states(\mathsf{C}_i),\ u \in \alpha(\mathsf{C}_i)\} \\
T^{\mathbf{f}} &= \bigcup\nolimits_{\mathsf{I} \in \mathfrak{I}} \{\ (\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}], \langle t_1, \ldots, t_{\#(\mathsf{I})}\rangle)\ \mid\ \langle p_1, \ldots, p_{\#(\mathsf{I})}\rangle = \mathfrak{P}(\mathsf{I}), \\
&\qquad \langle u_1, \ldots, u_{\#(\mathsf{I})}\rangle \in (\mathbf{f}(\alpha))(\mathsf{I}),\ \langle t_1, \ldots, t_{\#(\mathsf{I})}\rangle \in T_\beta(\mathsf{I}), \\
&\qquad \forall i, j \in [1, \#(\mathsf{I})].\ i \neq j \Rightarrow u_i \neq u_j \text{ or } comp(p_i) \neq comp(p_j)\ \} \\
&= \bigcup\nolimits_{\mathsf{I} \in \mathfrak{I}} \{\ (\mathsf{I}[f_{k_1}(u_1), \ldots, f_{k_{\#(\mathsf{I})}}(u_{\#(\mathsf{I})})], \langle t_1, \ldots, t_{\#(\mathsf{I})}\rangle)\ \mid \\
&\qquad \langle p_1, \ldots, p_{\#(\mathsf{I})}\rangle = \mathfrak{P}(\mathsf{I}),\ \forall i \in [1, \#(\mathsf{I})].\ \mathsf{C}_{k_i} = comp(p_i), \\
&\qquad \langle u_1, \ldots, u_{\#(\mathsf{I})}\rangle \in \alpha(\mathsf{I}),\ \langle t_1, \ldots, t_{\#(\mathsf{I})}\rangle \in T_\beta(\mathsf{I}), \\
&\qquad \forall i, j \in [1, \#(\mathsf{I})].\ i \neq j \Rightarrow f_{k_i}(u_i) \neq f_{k_j}(u_j) \text{ or } comp(p_i) \neq comp(p_j)\ \}
\end{aligned}
$$

$$E^{\mathbf{f}} = \bigcup_{\mathsf{I}\in\mathfrak{I}} \{\ (q[f_{k_i}(u_i)], (\mathsf{I}[f_{k_1}(u_1), \ldots, f_{k_{\#(\mathsf{I})}}(u_{\#(\mathsf{I})})], \langle t_1, \ldots, t_{\#(\mathsf{I})}\rangle)),$$
$$((\mathsf{I}[f_{k_1}(u_1), \ldots, f_{k_{\#(\mathsf{I})}}(u_{\#(\mathsf{I})})], \langle t_1, \ldots, t_{\#(\mathsf{I})}\rangle), q'[f_{k_i}(u_i)]) \ \mid$$
$$t_i = (q \xrightarrow{p_i}_{comp(p_i)} q'),\ i \in [1, \#(\mathsf{I})]\ \}$$

We obtain the following equivalence:

$$\mathrm{m} \xrightarrow{(\mathsf{I}[u_1,\ldots,u_{\#(\mathsf{I})}], \langle t_1,\ldots,t_{\#(\mathsf{I})}\rangle)} \mathrm{m}' \text{ in } \mathsf{N}(\Sigma, \alpha, \beta)$$
$$\iff$$
$$\mathbf{f}(\mathrm{m}) \xrightarrow{(\mathsf{I}[f_{k_1}(u_1),\ldots,f_{k_{\#(\mathsf{I})}}(u_{\#(\mathsf{I})})], \langle t_1,\ldots,t_{\#(\mathsf{I})}\rangle)} \mathbf{f}(\mathrm{m}') \text{ in } \mathsf{N}(\Sigma, \mathbf{f}(\alpha), \beta)$$

where $k_1, \ldots, k_{\#(\mathsf{I})} \in [1, N]$ are such that $\mathsf{C}_{k_i} = comp(\langle \mathfrak{P}(\mathsf{I})\rangle_i)$, for all $i \in [1, \#(\mathsf{I})]$. Point (1) uses this fact inductively, on the length of the firing sequence leading from m to m′. For point (2) we use, moreover, that $q[u] \in \mathrm{m} \iff q[f_i(u)] \in \mathbf{f}(\mathrm{m})$ and $q[u] \in {}^\bullet(\mathsf{I}[u_1, \ldots, u_{\#(\mathsf{I})}], \langle t_1, \ldots, t_{\#(\mathsf{I})}\rangle) \iff q[f_i(u)] \in {}^\bullet(\mathsf{I}[f_{k_1}(u_1), \ldots, f_{k_{\#(\mathsf{I})}}(u_{\#(\mathsf{I})})], \langle t_1, \ldots, t_{\#(\mathsf{I})}\rangle)$, where C_i is the unique component type such that $q \in states_\beta(\mathsf{C}_i)$. □

Theorem 2. *Given a tight sentence ϕ, interpreted over a SID Δ, a behavior map β and a tuple of states $\langle q_1, \ldots, q_k\rangle$, the following equivalences hold:*

$$deadlock_t(\phi, \Delta, \beta) \iff deadlock(\phi, \Delta, \beta)$$
$$reach_t(\phi, \langle q_1, \ldots, q_k\rangle, \Delta, \beta) \iff reach(\phi, \langle q_1, \ldots, q_k\rangle, \Delta, \beta)$$

where $deadlock_t(\phi, \Delta, \beta)$ and $reach_t(\phi, \langle q_1, \ldots, q_k\rangle, \Delta, \beta)$ are defined below:

- *$deadlock_t(\phi, \Delta, \beta)$: does there exist rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, such that $Reach(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta), \mathrm{m}_\mathcal{T}) \cap Dead(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta)) \neq \emptyset$?*
- *$reach_t(\phi, \langle q_1, \ldots, q_k\rangle, \Delta, \beta)$: does there exist a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$ and a configuration $(\alpha_\mathcal{T}, \mathrm{m}) \in Reach(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta), \mathrm{m}_\mathcal{T})$, such that $\{q_i[u_i] \mid i \in [1, k]\} \subseteq \mathrm{m}$, with $q_1[u_1], \ldots, q_k[u_k]$ pairwise distinct?*

Proof. We prove the following points:

- $deadlock_t(\phi, \Delta, \beta) \Rightarrow deadlock(\phi, \Delta, \beta)$: assume that $deadlock_t(\phi, \Delta, \beta)$ has a positive answer and let $\mathcal{T} \in \mathbb{T}(\phi)$ be a rewriting tree, such that $Reach(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta), \mathrm{m}_\mathcal{T}) \cap Dead(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta)) \neq \emptyset$, the existence of which is stated by $deadlock_t(\phi, \Delta, \beta)$. By Lemma 3, $(\alpha_\mathcal{T}, \mathrm{m}_\mathcal{T}) \models \mathfrak{F}(\mathcal{T})$ and, by Lemma 1, we obtain $(\alpha_\mathcal{T}, \mathrm{m}_\mathcal{T}) \models \phi$, thus $deadlock(\phi, \Delta, \beta)$ has a positive answer.
- $deadlock(\phi, \Delta, \beta) \Rightarrow deadlock_t(\phi, \Delta, \beta)$: assume that $deadlock(\phi, \Delta, \beta)$ has a positive answer and let (α, m) be a model of ϕ and m′ be a marking such that $\mathrm{m}' \in Reach(\mathsf{N}(\Sigma, \alpha, \beta), \mathrm{m}) \cap Dead(\mathsf{N}(\Sigma, \alpha, \beta))$. 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 ϕ is a tight sentence, we obtain that $(\alpha, \mathrm{m}) \sim (\alpha_\mathcal{T}, \mathrm{m}_\mathcal{T})$ and let $\mathbf{f} = \langle f_1, \ldots, f_N\rangle$ be the bijections from Def. 8, such that $(\alpha_\mathcal{T}, \mathrm{m}_\mathcal{T}) = (\mathbf{f}(\alpha), \mathbf{f}(\mathrm{m}))$. By Lemma 5, we obtain that $\mathbf{f}(\mathrm{m}') \in Reach(\mathsf{N}(\Sigma, \mathbf{f}(\alpha), \beta), \mathbf{f}(\mathrm{m})) \cap Dead(\mathsf{N}(\Sigma, \mathbf{f}(\alpha), \beta))$, thus $deadlock_t(\phi, \Delta, \beta)$ has a positive answer.

- $reach_t(\phi, \langle q_1, \ldots, q_k \rangle, \Delta, \beta) \Rightarrow reach(\phi, \langle q_1, \ldots, q_k \rangle, \Delta, \beta)$: assume that $reach_t(\phi, \langle q_1, \ldots, q_k \rangle, \Delta, \beta)$ has a positive answer and let $\mathcal{T} \in \mathbb{T}(\phi)$ be a rewriting tree and m be a marking of $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}}, \beta)$, such that $(\alpha_{\mathcal{T}}, \mathrm{m}) \in Reach(\mathsf{N}(\Sigma, \alpha_{\mathcal{T}}, \beta), \mathrm{m}_{\mathcal{T}})$ and $\{q_i[u_i] \mid i \in [1,k]\} \subseteq \mathrm{m}$, where $q_1[u_1]$, …, $q_k[u_k]$ are pairwise distinct places. By Lemma 3, $(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}) \models \mathfrak{F}(\mathcal{T})$ and, by Lemma 1, we obtain $(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}) \models \phi$, thus $reach(\phi, \langle q_1, \ldots, q_k \rangle, \Delta, \beta)$ has a positive answer.
- $reach(\phi, \langle q_1, \ldots, q_k \rangle, \Delta, \beta) \Rightarrow reach_t(\phi, \langle q_1, \ldots, q_k \rangle, \Delta, \beta)$: assume that $reach(\phi, \langle q_1, \ldots, q_k \rangle, \Delta, \beta)$ has a positive answer and let (α, m) be a model of ϕ and $\mathrm{m}' \in Reach(\mathsf{N}(\Sigma, \alpha, \beta), \mathrm{m})$ be a marking, such that $\{q_i[u_i] \mid i \in [1,k]\} \subseteq \mathrm{m}'$, where $q_1[u_1], \ldots, q_k[u_k]$ are pairwise distinct places of $\mathsf{N}(\Sigma, \alpha, \beta)$. 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 ϕ is a tight sentence, we obtain that $(\alpha, \mathrm{m}) \sim (\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}})$ and let $\mathbf{f} = \langle f_1, \ldots, f_N \rangle$ be the bijections from Def. 8, such that $(\alpha_{\mathcal{T}}, \mathrm{m}_{\mathcal{T}}) = (\mathbf{f}(\alpha), \mathbf{f}(\mathrm{m}))$. By Lemma 5, we obtain that $\mathbf{f}(\mathrm{m}') \in Reach(\mathsf{N}(\Sigma, \mathbf{f}(\alpha), \beta), \mathbf{f}(\mathrm{m}))$. Moreover, if C_{ℓ_i} is the unique component type, such that $q_i \in states_\beta(\mathsf{C}_{\ell_i})$, for all $i \in [1,k]$, since $\{q_i[u_i] \mid i \in [1,k]\} \subseteq \mathrm{m}'$, we obtain $\{q_i[f_{\ell_i}(u_i)] \mid i \in [1,k]\} \subseteq \mathbf{f}(\mathrm{m}')$ and $reach_t(\phi, \langle q_1, \ldots, q_k \rangle, \Delta, \beta)$ 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 κ, with second-order set variables ranging over finite sets. This logic is known as the Weak Second-order calculus of κ Successors ($\mathsf{WS}\kappa S$). Well-known automata-theoretic decision procedures exist for $\mathsf{WS}\kappa S$ [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 $\mathcal{A}(y_1, \ldots, y_{\#(\mathcal{A})})$ with one of its definitions $\varphi[x_1/y_1 \ldots x_{\#(\mathcal{A})}/y_{\#(\mathcal{A})}]$, for some rule $\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \varphi$ from the SID) before it is instantiated by a component atom $\mathsf{C}^q(x)$, that occurs in some node $w \in \mathrm{nodes}(\mathcal{T})$. 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 $\mathsf{C}^q(x)$ occurs.

Example 8. *Let us consider the rewriting tree $\mathcal{T}$ depicted in Fig. 5(a). The variable ℓ_1^{ϵ} introduced by an existential quantifier at the root of the tree replaces the ℓ parameter of the rule that labels the left successor of the root, i.e. the*

node 1*, that replaces the* ℓ *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* ℓ_1^ϵ *in the canonical store is* $\mathfrak{s}_\mathcal{T}(\ell_1^\epsilon) = 111$*. 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 $\mathsf{WS}\kappa S$ *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 $\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta)$ corresponding to the canonical architecture $\alpha_\mathcal{T}$, defined by the rewriting tree $\mathcal{T}$, and the given behavior map β. Moreover, path formulæ allow to define $\mathsf{WS}\kappa S$ 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 $\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta)$. These invariants are defined by the sets of places of $\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta)$ 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 $\mathsf{WS}\kappa S$ formula.

6.1. Weak Second Order Calculus of κ *Successors*

We define the logic $\mathsf{WS}\kappa S$, which is a fragment of MSO interpreted over a finite prefix of an infinite κ-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 $\mathsf{WS}\kappa S$ are defined inductively by the following syntax:

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

where $x \in \mathbb{X}^1$, $X \in \mathbb{X}^2$ and $i \in [1, \kappa]$. As usual, we write $x \neq y \stackrel{\text{def}}{=} \neg x = y$, $\xi_1 \vee \xi_2 \stackrel{\text{def}}{=} \neg(\neg\xi_1 \wedge \neg\xi_2)$, $\xi_1 \rightarrow \xi_2 \stackrel{\text{def}}{=} \neg\xi_1 \vee \xi_2$, $\xi_1 \leftrightarrow \xi_2 \stackrel{\text{def}}{=} (\xi_1 \rightarrow \xi_2) \wedge (\xi_2 \rightarrow \xi_1)$, $\forall x \,.\, \xi \stackrel{\text{def}}{=} \neg\exists x \,.\, \neg\xi$ and $\forall X \,.\, \xi \stackrel{\text{def}}{=} \neg\exists X \,.\, \neg\xi$. For a constant $n \in \mathbb{N}$, we define:

$$X = \{y_1, \ldots, y_n\} \iff \forall x \,.\, X(x) \leftrightarrow \bigvee_{i=1}^{n} x = y_i \tag{12}$$

$$\|X\| \geq n \iff \exists y_1 \ldots \exists y_n \,.\, \bigwedge_{1 \leq i < j \leq n} y_i \neq y_j \wedge \bigwedge_{i=1}^{n} X(y_i) \tag{13}$$

$$\|X\| = n \iff \|X\| \geq n \wedge \neg\|X\| \geq n+1 \tag{14}$$

The variables of a $\mathsf{WS}\kappa S$ 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 $\mathsf{WS}\kappa S$ are interpreted inductively, as $\nu(\epsilon) \stackrel{\text{def}}{=} \epsilon$ and $\nu(\tau.i) \stackrel{\text{def}}{=} \nu(\tau) \cdot i$. We denote by $\models^{\nu}_{\mathsf{wsks}} \xi$ the fact that the $\mathsf{WS}\kappa S$ formula ξ is *valid* for the valuation ν. This relation defined below, by induction on the structure of the formulæ:

$$
\begin{array}{lcl}
\models^{\nu}_{\mathsf{wsks}} \tau_1 = \tau_2 & \iff & \nu(\tau_1) = \nu(\tau_2) \\
\models^{\nu}_{\mathsf{wsks}} X(\tau) & \iff & \nu(\tau) \in \nu(X) \\
\models^{\nu}_{\mathsf{wsks}} \xi_1 \wedge \xi_2 & \iff & \models^{\nu}_{\mathsf{wsks}} \xi_1 \text{ and } \models^{\nu}_{\mathsf{wsks}} \xi_2 \\
\models^{\nu}_{\mathsf{wsks}} \neg\xi & \iff & \text{not } \models^{\nu}_{\mathsf{wsks}} \xi \\
\models^{\nu}_{\mathsf{wsks}} \exists x \ . \ \xi & \iff & \models^{\nu[x \leftarrow w]}_{\mathsf{wsks}} \xi, \text{ for some node } w \in [1,\kappa]^* \\
\models^{\nu}_{\mathsf{wsks}} \exists X \ . \ \xi & \iff & \models^{\nu[X \leftarrow S]}_{\mathsf{wsks}} \xi, \text{ for some finite set } S \subseteq [1,\kappa]^*
\end{array}
$$

A $\mathsf{WS}\kappa S$ formula ξ is *satisfiable* if and only if there exists a store ν, such that $\models^{\nu}_{\mathsf{wsks}} \xi$. Such a store is said to be a *model* of ξ. 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 $\mathsf{WS}\kappa S$ formula that describes an infinite κ-ary tree, whose finite prefix is a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, for a CL formula ϕ. Let $\Delta = \{\rho_1, \ldots, \rho_P\}$ be a fixed SID in the following, such that ρ_1 is the rule that labels the root of $\mathcal{T}$, by convention. We use a designated tuple of second order variables $\mathbf{R} = \langle R_1, \ldots, R_P \rangle$, where each variable R_i is interpreted as the set of tree nodes labeled with the rule ρ_i in some rewriting tree, for all $i \in [1,P]$. With this convention, the $\mathit{RewrTree}_{\Delta}(\mathbf{R})$ 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 $\mathcal{A}'(y_1, \ldots, y_{\#(\mathcal{A}')})$ from a rule that labels a node w in the rewriting tree there is exactly one child of that node, labeled with a rule $\mathcal{A}'(x_1, \ldots, x_{\#(\mathcal{A}')}) \leftarrow \psi$ and w has no other children.

It is not very hard to show that, for each model ν of $\mathit{RewrTree}_{\Delta}(\mathbf{R})$ there exists a unique rewriting tree, denoted by $\mathcal{T}^{\mathbf{R}}_{\nu}$, such that $\mathrm{nodes}(\mathcal{T}^{\mathbf{R}}_{\nu}) = \bigcup_{i=1}^{P} \nu(R_i)$ and $\mathcal{T}^{\mathbf{R}}_{\nu}(w) = \rho_i \iff w \in \nu(R_i)$, for all $i \in [1,P]$ and $w \in \mathrm{nodes}(\mathcal{T}^{\mathbf{R}}_{\nu})$.

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

$$RewrTree_{\Delta}(\mathbf{R}) \stackrel{\text{def}}{=} \forall x \ . \bigwedge_{1 \leq i < j \leq P} \big(\neg R_i(x) \vee \neg R_j(x)\big) \wedge \big(R_1(x) \leftrightarrow x = \epsilon\big) \wedge \tag{15}$$

$$\forall x \ . \bigwedge_{i=1}^{P} \bigwedge_{\ell=1}^{\kappa} R_i(x.\ell) \rightarrow \bigvee_{j=1}^{P} R_j(x) \wedge \tag{16}$$

$$\forall x \ . \bigwedge_{i:\rho_i = \left(\mathcal{A}(x_1,\ldots,x_{\#(\mathcal{A})}) \leftarrow \varphi\right)} \bigwedge_{j=1}^{\#_{\mathsf{pred}}(\varphi)} R_i(x) \rightarrow \bigvee_{\substack{j:\mathsf{pred}_j(\varphi) = \mathcal{A}'(y_1,\ldots,y_{\#(\mathcal{A}')}) \\ \ell:\rho_\ell = \left(\mathcal{A}'(x_1,\ldots,x_{\#(\mathcal{A}')}) \leftarrow \psi\right)}} R_\ell(x.j) \wedge \tag{17}$$

$$\forall x \ . \bigwedge_{i:\rho_i = \left(\mathcal{A}(x_1,\ldots,x_{\#(\mathcal{A})}) \leftarrow \varphi\right)} \bigwedge_{j=\#_{\mathsf{pred}}(\varphi)+1}^{\kappa} R_i(x) \rightarrow \bigwedge_{\ell=1}^{P} \neg R_\ell(x.j) \tag{18}$$

Figure 6: Rewriting Trees

net $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}}, \beta)$, for a given signature $\Sigma = (\{\mathsf{C}_1, \ldots, \mathsf{C}_N\}, \{\mathsf{I}_1, \ldots, \mathsf{I}_M\}, \mathfrak{P})$ and behavior map β. In the rest of this section, we consider Σ and β to be fixed and recall that $\mathcal{Q}_\beta$ denotes the finite set of states used to describe the behaviors $\beta(\mathsf{C}_1), \ldots, \beta(\mathsf{C}_N)$.

Our next concern is defining the precise markings of $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}}, \beta)$ (Def. 2) by a $\mathsf{WS}\kappa S$ formula. We define sets of places of $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}}, \beta)$ by means of a tuple of second-order variables $\mathbf{X} \stackrel{\text{def}}{=} \langle X_q \mid q \in \mathcal{Q}_\beta \rangle$. Any valuation ν of the variables $\mathbf{X}$ corresponds to a set of places $\sigma_\nu^{\mathbf{X}} \stackrel{\text{def}}{=} \{q[u] \mid u \in \nu(X_q)\}$. When needed, we shall use distinct copies of $\mathbf{X}$, such as $\mathbf{X'} \stackrel{\text{def}}{=} \langle X'_q \mid q \in \mathcal{Q}_\beta \rangle$, $\mathbf{Y} \stackrel{\text{def}}{=} \langle Y_q \mid q \in \mathcal{Q}_\beta \rangle$ and $\mathbf{Z} \stackrel{\text{def}}{=} \langle Z_q \mid q \in \mathcal{Q}_\beta \rangle$. The $\mathsf{WS}\kappa S$ formula below defines the precise markings of a Petri net $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}}, \beta)$, assuming that $\mathcal{T}$ is the rewriting tree corresponding to some model of $RewrTree_{\Delta}(\mathbf{R})$:

$$\mathrm{m}(\mathbf{X}, \mathbf{R}) \stackrel{\text{def}}{=} \ \forall x \ . \bigwedge_{i=1}^{N} \Big(\bigwedge_{q \neq q' \in states_\beta(\mathsf{C}_i)} \big(\neg X_q(x) \vee \neg X_{q'}(x)\big) \wedge \tag{19}$$

$$\Big(\bigvee_{q \in states_\beta(\mathsf{C}_i)} X_q(x) \leftrightarrow \bigvee_{\rho_j \in inst(\mathsf{C}_i)} R_j(x) \Big) \Big) \tag{20}$$

Intuitively, line (19) above states that a component of type C_i may not be in two different states from $states_\beta(\mathsf{C}_i)$, 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 $\mathsf{C}_i^q(x)$ occurs (see Assumption 2 on the syntax of the rules labeling a rewriting tree). We write $inst(\mathsf{C})$ for the subset of Δ consisting of those rules that contain a component atom of the form $\mathsf{C}^q(x)$. The correctness of the encoding is stated and proved below:

Lemma 6. *For any model ν of $RewrTree_{\Delta}(\mathbf{R}) \wedge \mathrm{m}(\mathbf{X}, \mathbf{R})$, the set $\sigma^{\mathbf{X}}_{\nu}$ is a precise marking of the Petri net $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^{\mathbf{R}}_{\nu}}, \beta)$.*

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

- $\sigma^{\mathbf{X}}_{\nu} \cap \{q[u] \mid q \in states_{\beta}(\mathsf{C})\} \neq \emptyset$: since $\models^{\nu}_{\mathsf{wsks}} RewrTree_{\Delta}(\mathbf{R})$, we have that $\mathcal{T}^{\mathbf{R}}_{\nu}$ is a rewriting tree, such that $\mathrm{nodes}(\mathcal{T}^{\mathbf{R}}_{\nu}) = \bigcup_{i=1}^{P} \nu(R_i)$ and $\nu(R_i)$ is the set of nodes of $\mathcal{T}^{\mathbf{R}}_{\nu}$ that are labeled with the rule ρ_i, for all $i \in [1, P]$. Since $u \in \alpha_{\mathcal{T}^{\mathbf{R}}_{\nu}}(\mathsf{C}) \subseteq \mathrm{nodes}(\mathcal{T}^{\mathbf{R}}_{\nu})$, there exists a rule ρ_i, such that $\mathsf{C} \in inst(\rho_i)$ and $u \in \nu(R_i)$, hence by line (20), we have $u \in \nu(X_q)$, i.e. $q[u] \in \sigma^{\mathbf{X}}_{\nu}$, for some $q \in states_{\beta}(\mathsf{C})$. Consequently, $q[u] \in \sigma^{\mathbf{X}}_{\nu} \cap \{q[u] \mid q \in states_{\beta}(\mathsf{C})\}$.
- $\|\sigma^{\mathbf{X}}_{\nu} \cap \{q[u] \mid q \in states_{\beta}(\mathsf{C})\}\| < 2$: suppose that $q[u], q'[u] \in \sigma^{\mathbf{X}}_{\nu}$, for $q \neq q' \in states_{\beta}(\mathsf{C})$. Hence $u \in \nu(X_q) \cap \nu(X_{q'})$, which contradicts line (19) and the fact that $\models^{\nu}_{\mathsf{wsks}} \mathrm{m}(\mathbf{X}, \mathbf{R})$. □

6.3. Path Automata

The next step in building $\mathsf{WS}\kappa S$ 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 (${}^{\bullet}t$) and post-set ($t^{\bullet}$) of a transition (t) from the Petri net $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}}, \beta)$, 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 $n_1, \ldots, n_\ell \in \mathrm{nodes}(\mathcal{T})$, 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 $s \xrightarrow{d} s'$, where $s, s' \in S$ 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 $A = (\{s_1, \ldots, s_L\}, I, F, \delta)$ 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 $\mathbf{Y} = \langle Y_1, \ldots, Y_L \rangle$ 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

$$\Delta_A(x, y, \mathbf{Y}) \stackrel{\text{def}}{=} \bigwedge_{1 \leq i \neq j \leq L} \forall z. \big(\neg Y_i(z) \vee \neg Y_j(z)\big) \wedge \bigvee_{s_i \in I} Y_i(x) \ \wedge \ \bigvee_{s_j \in F} Y_j(y) \quad (21)$$

$$\wedge \bigwedge_{i=1}^{L} \Big(\forall z \ . \ z \neq y \wedge Y_i(z) \rightarrow \bigvee_{j : s_i \xrightarrow{d} s_j} Y_j(z.d) \vee \bigvee_{j : s_i \xrightarrow{d} s_j} \exists z' \ . \ z'.d = z \wedge Y_j(z')\Big) \quad (22)$$

$$\wedge \bigwedge_{j=1}^{L} \Big(\forall z \ . \ z \neq x \wedge Y_j(z) \rightarrow \bigvee_{i : s_i \xrightarrow{d} s_j} \exists z' \ . \ z'.d = z \wedge Y_i(z') \vee \bigvee_{i : s_i \xrightarrow{d} s_j} Y_i(z.d)\Big) \quad (23)$$

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 ν, such that $\nu(x) = n$ and $\nu(y) = m$, we have $d_1 \ldots d_p \in \mathcal{L}(A) \iff \models^{\nu}_{\mathsf{wsks}} \exists \mathbf{Y} \ . \ \Delta_A(x, y, \mathbf{Y})$.*

Our purpose is to infer, directly from the syntax of the rules in Δ, 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 $\rho = (\mathcal{A}(x_1, \ldots, x_{\#\mathcal{A}}) \leftarrow \exists y_1 \ldots \exists y_m \ . \ \psi)$, such that ψ is a quantifie-free formula, and each variable $x \in \mathrm{fv}(\psi)$, we consider the path automaton $A^x_\rho = (S, I, F, \delta)$, where:

- $S \stackrel{\text{def}}{=} \{s^z_{\rho'} \mid \rho' = (\mathcal{A}'(x_1, \ldots, x_{\#\mathcal{A}'}) \leftarrow \exists y_1 \ldots \exists y_m . \phi) \, , \ \phi \text{ is q.f.}, \ z \in \mathrm{fv}(\phi)\}$; the intuition is that the automaton visits the state $s^z_{\rho'}$ while tracking variable z in the body of the rule ρ', that labels the node of the rewriting tree which is currently visited by the automaton,
- the initial and final states are $I \stackrel{\text{def}}{=} \{s^x_\rho\}$ and $F \stackrel{\text{def}}{=} \{s^z_{\rho'} \mid \rho' = \big(\mathcal{A}'(x_1, \ldots, x_{\#\mathcal{A}'}) \leftarrow \exists y_1 \ldots \exists y_m \ . \ \mathsf{C}^q(z) * \phi\big), \ \phi \text{ is q.f.}, \ \mathsf{C} \in \mathfrak{C}, q \in \mathit{states}_\beta(\mathsf{C})\}$; the automaton starts to track x in ρ and moves down in the rewriting tree, finally tracking a variable z that occurs in a component atom in ρ',

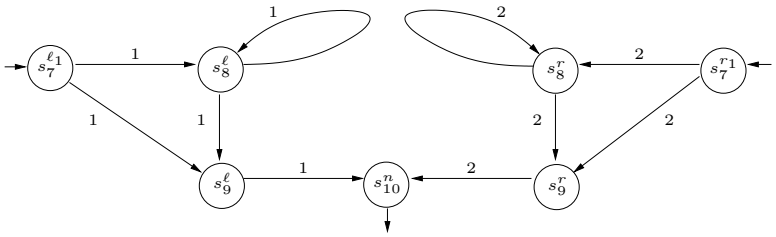

Figure 8: Path Automata Recognizing the Paths to Component Atoms from Example 6

- the transitions are $s_{\rho}^{y_j} \xrightarrow{d} s_{\rho'}^{x_j}$, for all rules $\rho = \big(\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \varphi\big)$ and $\rho' = \big(\mathcal{A}'(x_1, \ldots, x_{\#(\mathcal{A}')}) \leftarrow \varphi'\big)$ from Δ, all directions $d \in [1, \#_{\mathsf{pred}}(\varphi)]$, such that $\mathsf{pred}_d(\varphi) = \mathcal{A}'(y_1, \ldots, y_{\#(\mathcal{A}')})$, and all $j \in [1, \#(\mathcal{A}')]$; if ρ labels the parent of the node labeled by ρ' in the rewriting tree, the automaton moves down one step, from tracking y_j in ρ to tracking x_j in ρ'.

Example 9. *The path automata recognizing the paths to component atoms, for the variables ℓ_1^{ϵ} and r_1^{ϵ} in the rewriting tree from Example 6 are depicted in Figure 8. The initial states are $s_7^{\ell_1}$ and $s_7^{r_1}$, respectively, and the final state is s_{10}^{n} 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 $\rho = \big(\mathcal{A}(x_1, \ldots, x_{\#(\mathcal{A})}) \leftarrow \exists y_1 \ldots \exists y_m \,.\, \phi\big)$, such that ϕ is a quantifier-free formula, and a variable $x \in \mathrm{fv}(\phi)$, for each sequence $w \in [1, \kappa]^*$, the following are equivalent:*

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

Proof. (1) $\Rightarrow$ (2) By the definition of $\mathfrak{s}_{\mathcal{T}}$, we have $\mathfrak{s}_{\mathcal{T}}(x) = \mathfrak{s}^{\epsilon}_{\mathcal{T}\downarrow_n}(x)$ and $\mathfrak{s}^{\epsilon}_{\mathcal{T}\downarrow_n}(x) = m$ if there exist:

- a sequence of nodes $n = n_1, n_2, \ldots, n_p = m \in \mathrm{nodes}(\mathcal{T})$, 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 $\rho_i = \big(\mathcal{A}_i(x_1, \ldots, x_{\#(\mathcal{A}_i)}) \leftarrow \exists y_1 \ldots \exists y_m \,.\, \phi_i\big)$, where ϕ_i is quantifier-free, and there exists $j \in [1, \#(\mathcal{A}_i)]$ such that z_i occurs on the j-th position in $\mathsf{pred}_{d_i}(\phi_i)$ and z_{i+1} occurs j-th within the head $\mathcal{A}_{i+1}(x_1, \ldots, x_{\#(\mathcal{A}_{i+1})})$ of ρ_{i+1}. By the definition of A_{ρ}^{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,

$s^x_\rho = s^{z_1}_{\rho_1}$ and $s^{z_p}_{\rho_p}$ are initial state and final states of A^x_ρ, respectively. Since $m = n \cdot d_1 \ldots d_p = n \cdot w$, we obtain that $w = d_1 \ldots d_p \in \mathcal{L}(A^x_\rho)$.

(2) $\Rightarrow$ (1) Let $w = d_1 \ldots d_p$, where $d_1, \ldots, d_p \in [1, \kappa]$, and $s^{z_i}_{\rho_i} \xrightarrow{d_i} s^{z_{i+1}}_{\rho_{i+1}}$, for all $i \in [1, p-1]$, be transitions of A^x_ρ, such that $s^{z_1}_{\rho_1} = s^x_\rho$ is the initial state and $s^{z_p}_{\rho_p}$ is a final state of A^x_ρ. Then, by the definition of A^x_ρ, the nodes $w_i \stackrel{\text{def}}{=} n \cdot d_1 \ldots d_i$ are labeled by rules $\rho_i \stackrel{\text{def}}{=} \big(\mathcal{A}_i(x_1, \ldots, x_{\#(\mathcal{A}_i)}) \leftarrow \varphi_i\big)$, such that z_i occurs in $\mathsf{pred}_{d_i}(\varphi_i)$ on the same position as z_{i+1}, which occurs in $\mathcal{A}_{i+1}(x_1, \ldots, x_{\#(\mathcal{A}_{i+1})})$. Then, there exists a rewriting tree $\mathcal{T}$ and nodes $n, m \in \mathrm{nodes}(T)$, such that $m = n \cdot d_1 \ldots d_o$ and A^x_ρ has an accepting run over $d_1 \ldots d_p$. By the definition of $\mathfrak{s}_\mathcal{T}$, we have $\mathfrak{s}_\mathcal{T}(x) = \mathfrak{s}^\epsilon_{\mathcal{T}\downarrow_n}(x) = m$. □

The path automata A^z_ρ are used to define the $\mathit{Path}^z_\rho(x, y, \mathbf{R})$ formulæ below:

$$\begin{array}{rcl} \mathit{Path}^z_\rho(x, y, \mathbf{R}) & \stackrel{\text{def}}{=} & \exists \mathbf{Y} \,.\, \Delta_{A^z_\rho}(x, y, \mathbf{Y}) \wedge \Upsilon(\mathbf{Y}, \mathbf{R}) \\ \Upsilon(\mathbf{Y}, \mathbf{R}) & \stackrel{\text{def}}{=} & \bigwedge_{i:\rho_i = (\mathcal{A}'(x_1, \ldots, x_{\#(\mathcal{A}')}) \leftarrow \varphi')} \bigwedge_{z \in \mathrm{fv}(\varphi')} \forall x \,.\, Y^z_{\rho_i}(x) \rightarrow R_i(x) \end{array}$$

The formula $\Upsilon(\mathbf{Y}, \mathbf{R})$ above states that all nodes labeled with a state $q^z_{\rho_i}$ during the run must be also labeled with ρ_i in the rewriting tree given as input to the path automaton. Here we denote by $Y^z_{\rho_i}$ the second-order variable corresponding to the state $q^z_{\rho_i}$ in a path automaton A^x_ρ. Intuitively, the formula $\mathit{Path}^z_\rho(x, y, \mathbf{R})$ states that, in each rewriting tree defined by a model ν of the $\mathit{RewrTree}_\Delta(\mathbf{R})$ formula, there exists a path from $\nu(x)$ to $\nu(y)$, where $\nu(x)$ is a node labeled with ρ, 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 Formulæ

At this point, we introduce a $\mathsf{WS}\kappa S$ formula $\Phi(\mathbf{X}, \mathbf{X}', \mathbf{R})$ that defines the *structure* of transitions in the Petri net $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^\mathbf{R}_\nu}, \beta)$, where ν is a model of $\mathit{RewrTree}_\Delta(\mathbf{R})$. By "structure" here we mean that $\nu(\mathbf{X})$ and $\nu(\mathbf{X}')$ encode the pre- and post-sets of some transition in $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^\mathbf{R}_\nu}, \beta)$, 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 $\mathit{inter}(\rho)$ the set of interaction atoms that occur in the body of a rule ρ. Essentially, the formula (24) is split into a disjunction of formulæ $\Psi_{\ell, \mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})}$, one for each rule $\rho_\ell \in \Delta$ and each interaction atom $\mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})$ that occurs in the body of ρ_ℓ. To understand the $\Psi_{\ell, \mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})}$ formulæ, recall that each of the variables x_i is mapped to the (unique) node of the rewriting tree containing an instance atom $\mathsf{C}^q(x)$, such that x_i replaces x in the characterizing formula of the rewriting tree $\mathcal{T}^\mathbf{R}_\nu$, provided that ν is a model of $\mathit{RewrTree}_\Delta(\mathbf{R})$. In order to find this node, we track the variable x_i from the node labeled by the rule ρ, to the node where this instance atom occurs. This is done by the $\mathit{Path}^{x_i}_{\rho_\ell}(y_0, y_i, \mathbf{R})$ formula (25), that holds whenever y_0 and y_i are respectively mapped to the source and the destination of a path from a node $\nu(y_0) \in \mathrm{nodes}(\mathcal{T}^\mathbf{R}_\nu)$, with label ρ_ℓ to a node $\nu(y_i) \in \mathrm{nodes}(\mathcal{T}^\mathbf{R}_\nu)$, such that the

$$\Phi(\mathbf{X},\mathbf{X}',\mathbf{R}) \stackrel{\text{def}}{=} \bigvee_{\ell=1}^{P} \bigvee_{\mathsf{I}(x_1,\ldots,x_{\#(\mathsf{I})})\in inter(\rho_\ell)} \Psi_{\ell,\mathsf{I}(x_1,\ldots,x_{\#(\mathsf{I})})}(\mathbf{X},\mathbf{X}',\mathbf{R}) \quad (24)$$

$$\Psi_{\ell,\mathsf{I}(x_1,\ldots,x_{\#(\mathsf{I})})}(\mathbf{X},\mathbf{X}',\mathbf{R}) \stackrel{\text{def}}{=} \exists y_0 \ldots \exists y_{\#(\mathsf{I})} \,.\, R_\ell(y_0) \wedge \bigwedge_{i=1}^{\#(\mathsf{I})} Path_{\rho_\ell}^{x_i}(y_0,y_i,\mathbf{R}) \wedge \quad (25)$$

$$\Big(\bigvee_{\langle\tau_1,\ldots,\tau_{\#(\mathsf{I})}\rangle\in T_\beta(\mathsf{I})} \bigwedge_{q\in\mathcal{Q}_\beta} X_q = \{y_i \mid i\in[1,\#(\mathsf{I})], \tau_i = (q_i \xrightarrow{p_i} q_i'), q_i = q\} \wedge \quad (26)$$

$$X_q' = \{y_i \mid i\in[1,\#(\mathsf{I})], \tau_i = (q_i \xrightarrow{p_i} q_i'), q_i' = q\}\Big) \wedge \quad (27)$$

$$\bigwedge_{1\le i<j\le\#(\mathsf{I})} \big(y_i \neq y_j \vee comp(\langle\mathfrak{P}(\mathsf{I})\rangle_i) = comp(\langle\mathfrak{P}(\mathsf{I})\rangle_j)\big) \quad (28)$$

Figure 9: The Flow Formula

component atom that gives the value of x_i[5] occurs in $\mathcal{T}_\nu^{\mathbf{R}}(\nu(y_i))$. Hence, x_i is interpreted as $\nu(y_i)$ 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 ν of $RewrTree_\Delta(\mathbf{R})$, we have $\models_{\mathsf{wsks}}^{\nu} \Phi(\mathbf{X},\mathbf{X}',\mathbf{R})$ if and only if $\sigma_\nu^{\mathbf{X}} = {}^\bullet t$ and $\sigma_\nu^{\mathbf{X}'} = t^\bullet$, for some transition t of $\mathsf{N}(\Sigma,\alpha_{\mathcal{T}_\nu^{\mathbf{R}}},\beta)$.*

Proof. "$\Rightarrow$" If $\models_{\mathsf{wsks}}^{\nu} \Phi(\mathbf{X},\mathbf{X}',\mathbf{R})$ then there exists a rule $\rho_\ell \in \Delta = \{\rho_1,\ldots,\rho_N\}$ and $\mathsf{I}(x_1,\ldots,x_{\#(\mathsf{I})}) \in inter(\rho_\ell)$, such that $\models_{\mathsf{wsks}}^{\nu} \Psi_{\ell,\mathsf{I}(x_1,\ldots,x_{\#(\mathsf{I})})}(\mathbf{X},\mathbf{X}',\mathbf{R})$. Then, there exist nodes $w_0,\ldots,w_n \in \mathrm{nodes}(\mathcal{T}_\nu^{\mathbf{R}})$, such that $\nu(y_i) = w_i$, for all $i\in[0,n]$ and let $\nu' = \nu[y_0 \leftarrow w_0,\ldots,y_n \leftarrow w_n]$. Since, moreover, $\models_{\mathsf{wsks}}^{\nu'} R_\ell(y_0)$, we obtain that $\mathcal{T}_\nu^{\mathbf{R}}(w_0) = \rho_\ell$, hence the variables $x_1,\ldots,x_{\#(\mathsf{I})}$ occur in the body of the rule ρ_ℓ and are assigned to the unique nodes $w_1',\ldots,w_{\#(\mathsf{I})}' \in \mathrm{nodes}(\mathcal{T}_\nu^{\mathbf{R}})$, for which $\pi(w_0,w_i') \in \mathcal{L}(A_{\rho_\ell}^{x_i})$, for all $i\in[1,\#(\mathsf{I})]$, by Lemma 8. Since $\models_{\mathsf{wsks}}^{\nu'} Path_{\rho_\ell}^{x_i}(y_0,y_i,\mathbf{R})$, by Lemma 7, we obtain $\pi(w_0,w_i) \in \mathcal{L}(A_{\rho_\ell}^{x_i})$, thus $w_i = w_i'$, for all $i\in[1,\#(\mathsf{I})]$, by the uniqueness of these nodes. Let $\langle\tau_1,\ldots,\tau_{\#(\mathsf{I})}\rangle$ be a tuple of transitions from $T_\beta(\mathsf{I})$ synchronizing by I. Since $\models_{\mathsf{wsks}}^{\nu'} \Psi_{\ell,\mathsf{I}(x_1,\ldots,x_{\#(\mathsf{I})})}(\mathbf{X},\mathbf{X}',\mathbf{R})$, we have $w_i \neq w_j$, for each $1\le i<j\le\#(\mathsf{I})$, such that $comp(\langle\mathfrak{P}(\mathsf{I})\rangle_i) = comp(\langle\mathfrak{P}(\mathsf{I})\rangle_j)$. By Def. 1, we have that $t = (\mathsf{I}[w_1,\ldots,w_{\#(\mathsf{I})}],\langle\tau_1,\ldots,\tau_{\#(\mathsf{I})}\rangle)$ is a transition of $\mathsf{N}(\Sigma,\alpha_{\mathcal{T}_\nu^{\mathbf{R}}},\beta)$ and, moreover, ${}^\bullet t = \{q_1,\ldots,q_{\#(\mathsf{I})}\} = \sigma_\nu^{\mathbf{X}}$ and $t^\bullet = \{q_1',\ldots,q_{\#(\mathsf{I})}'\} = \sigma_\nu^{\mathbf{X}'}$, because $\models_{\mathsf{wsks}}^{\nu'} X_q = \{y_i \mid i\in[1,\#(\mathsf{I})], \tau_i = (q_i \xrightarrow{p_i} q_i'), q_i = q\}$ and $\models_{\mathsf{wsks}}^{\nu'} X_q' = \{y_i \mid i\in[1,\#(\mathsf{I})], \tau_i = (q_i \xrightarrow{p_i} q_i'), q_i' = q\}$, respectively.

"$\Leftarrow$" Let $t = (\mathsf{I}[w_1,\ldots,w_{\#(\mathsf{I})}],\langle\tau_1,\ldots,\tau_{\#(\mathsf{I})}\rangle)$ be a transition of $\mathsf{N}(\Sigma,\alpha_{\mathcal{T}_\nu^{\mathbf{R}}},\beta)$, where $\tau_i = q_i \xrightarrow{\langle\mathfrak{P}(\mathsf{I})\rangle_i} q_i'$ is a transition of $\beta(comp(\langle\mathfrak{P}(\mathsf{I})\rangle_i))$, for each $i \in$

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

$[1, \#(\mathsf{I})]$. Then there exists a node $w_0 \in \mathrm{nodes}(\mathcal{T}_\nu^{\mathbf{R}})$ and an interaction atom $\mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})$ in $\mathcal{T}_\nu^{\mathbf{R}}(w_0)$, such that $\mathfrak{s}_{\mathcal{T}_\nu^{\mathbf{R}}}(x_i) = w_i$, for all $i \in [1, \#(\mathsf{I})]$. Assume w.l.o.g. that $\mathcal{T}_\nu^{\mathbf{R}}(w_0) = \rho_\ell$, for some $\ell \in [1, P]$ and let $\nu' = \nu[y_0 \leftarrow w_0, \ldots, y_{\#(\mathsf{I})} \leftarrow w_{\#(\mathsf{I})}]$ be a valuation. We prove the following points:

- $\models^{\nu'}_{\mathsf{wsks}} R_\ell(y_0)$: since $\models^{\nu}_{\mathsf{wsks}} \mathit{RewrTree}_\Delta(\mathbf{R})$, $\mathcal{T}_\nu^{\mathbf{R}}(w_0) = \rho_\ell$, $\nu'(y_0) = w_0$ and ν' agrees with ν over $\mathbf{R}$.
- $\models^{\nu'}_{\mathsf{wsks}} \mathit{Path}^{x_i}_{\rho_\ell}(y_0, y_i, \mathbf{R})$, for all $i \in [1, \#(\mathsf{I})]$: since $\mathfrak{s}_{\mathcal{T}_\nu^{\mathbf{R}}}(x_i) = w_i$, we obtain $\pi(w_0, w_i) \in \mathcal{L}(A^{x_i}_{\rho_\ell})$ by Lemma 8 and $\models^{\nu'}_{\mathsf{wsks}} \mathit{Path}^{x_i}_{\rho_\ell}(y_0, y_i, \mathbf{R})$ follows, by Lemma 7.
- $\models^{\nu'}_{\mathsf{wsks}} y_i \neq y_j$, for all $1 \leq i < j \leq \#(\mathsf{I})$, such that $\mathit{comp}(\langle \mathfrak{P}(\mathsf{I}) \rangle_i) = \mathit{comp}(\langle \mathfrak{P}(\mathsf{I}) \rangle_j)$: by Def. 1, $w_i \neq w_j$ for all $i, j \in [1, \#(\mathsf{I})]$, such that $\mathit{comp}(\langle \mathfrak{P}(\mathsf{I}) \rangle_i) = \mathit{comp}(\langle \mathfrak{P}(\mathsf{I}) \rangle_j)$.
- $\models^{\nu'}_{\mathsf{wsks}} X_q = \{y_i \mid i \in [1, \#(\mathsf{I})], \tau_i = (q_i \xrightarrow{p_i} q'_i), q_i = q\}$, for all $q \in \mathcal{Q}_\beta$: because $\sigma^{\mathbf{X}}_\nu = {}^\bullet t = \{q_1, \ldots, q_{\#(\mathsf{I})}\}$.
- $\models^{\nu'}_{\mathsf{wsks}} X_q = \{y_i \mid i \in [1, \#(\mathsf{I})], \tau_i = (q_i \xrightarrow{p_i} q'_i), q'_i = q\}$, for all $q \in \mathcal{Q}_\beta$: because $\sigma^{\mathbf{X}'}_\nu = t^\bullet = \{q'_1, \ldots, q'_{\#(\mathsf{I})}\}$

We obtain that $\models^{\nu}_{\mathsf{wsks}} \Psi_{\ell, \mathsf{I}(x_1, \ldots, x_{\#(\mathsf{I})})}$ and, consequently $\models^{\nu}_{\mathsf{wsks}} \Phi(\mathbf{X}, \mathbf{X}', \mathbf{R})$. □

6.5. Structural Invariants

An *invariant* $\mathcal{I}$ of a marked Petri net $\mathcal{N} = (\mathsf{N}, \mathrm{m}_0)$, where $\mathsf{N} = (S, T, E)$ is a Petri net, is a set of markings that over-approximates the reachable states $\mathit{Reach}(\mathcal{N})$, that is $\mathit{Reach}(\mathcal{N}) \subseteq \mathcal{I} \subseteq \mathrm{pow}(S)$. 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 $\{\mathrm{m}' \mid \mathrm{m} \in \mathcal{I},\ t \in T,\ \mathrm{m} \xrightarrow{t} \mathrm{m}'\} \subseteq \mathcal{I}$.

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 $\mathcal{N} = (\mathsf{N}, \mathrm{m}_0)$, where $\mathsf{N} = (S, T, E)$, a set of places $\sigma \subseteq S$ is said to be a:*

- trap *if $\|\sigma \cap \mathrm{m}_0\| \geq 1$ and, for any $t \in T$, if $\|\sigma \cap {}^\bullet t\| \geq 1$ then $\|\sigma \cap t^\bullet\| \geq 1$,*
- mutex *if $\|\sigma \cap \mathrm{m}_0\| = 1$ and, for any $t \in T$, we have $\|\sigma \cap {}^\bullet t\| = \|\sigma \cap t^\bullet\| \leq 1$.*

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

- $\Theta(\mathcal{N}) \stackrel{\mathsf{def}}{=} \{\mathrm{m}$ *marking of* $\mathcal{N} \mid \|\mathrm{m} \cap \sigma\| \geq 1$, *for each trap* σ *of* $\mathcal{N}\}$,
- $\Omega(\mathcal{N}) \stackrel{\mathsf{def}}{=} \{\mathrm{m}$ *marking of* $\mathcal{N} \mid \|\mathrm{m} \cap \sigma\| = 1$, *for each mutex* σ *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 ϕ, interpreted over a SID Δ, a behavior map β and a tuple of states $\langle q_1, \ldots, q_n \rangle$, the following hold:*

$$\begin{array}{rcl} \mathit{deadlock}(\phi, \Delta, \beta) & \Rightarrow & \mathit{deadlock}^\sharp_t(\phi, \Delta, \beta) \\ \mathit{reach}(\phi, \langle q_1, \ldots, q_n \rangle, \Delta, \beta) & \Rightarrow & \mathit{reach}^\sharp_t(\phi, \langle q_1, \ldots, q_n \rangle, \Delta, \beta) \end{array}$$

where $\mathit{deadlock}^\sharp_t(\phi, \Delta, \beta)$ and $\mathit{reach}^\sharp_t(\phi, \langle q_1, \ldots, q_n \rangle, \Delta, \beta)$ are defined below:

- *$\mathit{deadlock}^\sharp_t(\phi, \Delta, \beta)$: does there exist a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$, such that $\Theta(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta), \mathrm{m}_\mathcal{T}) \cap \Omega(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta), \mathrm{m}_\mathcal{T}) \cap \mathit{Dead}(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta)) \neq \emptyset$?*
- *$\mathit{reach}^\sharp_t(\phi, \langle q_1, \ldots, q_n \rangle, \Delta, \beta)$: does there exist a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$ and a configuration $(\alpha_\mathcal{T}, \mathrm{m}) \in \Theta(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta), \mathrm{m}_\mathcal{T}) \cap \Omega(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta), \mathrm{m}_\mathcal{T})$, such that $\{q_i[u_i] \mid i \in [1, n]\} \subseteq \mathrm{m}$, with $q_1[u_1], \ldots, q_n[u_n]$ 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 $\mathrm{m}_0 = \emptyset$ then $\mathcal{N}$ has no traps nor mutexes and $\mathit{Reach}(\mathcal{N}) = \Theta(\mathcal{N}) = \Omega(\mathcal{N}) = \emptyset$, 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 σ be an arbitrary trap of $\mathcal{N}$. Since $\mathrm{m}_0 \neq \emptyset$, we have $\|\sigma \cap \mathrm{m}_0\| \geq 1$, hence $\mathrm{m}_0 \in \Theta(\mathcal{N})$, by the choice of σ. For each pair of markings m and m', such that $\mathrm{m} \in \Theta(\mathcal{N})$ and $\mathrm{m} \xrightarrow{t} \mathrm{m}'$, for some transition $t \in T$, we have $\mathrm{m}' = (\mathrm{m} \setminus {}^\bullet t) \cup t^\bullet$. If $\sigma \cap {}^\bullet t = \emptyset$ then $\|\mathrm{m}' \cap \sigma\| \geq \|\mathrm{m} \cap \sigma\|$ and $\|\mathrm{m} \cap \sigma\| \geq 1$, because $\mathrm{m} \in \Theta(\mathcal{N})$. Otherwise, $\|\sigma \cap {}^\bullet t\| \geq 1$, thus $\|\sigma \cap t^\bullet\| \geq 1$ because σ is a trap of $\mathcal{N}$, hence $\|\mathrm{m}' \cap \sigma\| \geq 1$. In both cases, we obtain that $\|\mathrm{m}' \cap \sigma\| \geq 1$, hence $\mathrm{m}' \in \Theta(\mathcal{N})$, because σ is an arbitrary trap of $\mathcal{N}$.

A similar argument shows that $\Omega(\mathcal{N})$ is an inductive invariant of $\mathcal{N}$. Let σ be an arbitrary mutex of $\mathcal{N}$. Then $\|\sigma \cap \mathrm{m}_0\| = 1$, hence $\mathrm{m}_0 \in \Omega(\mathcal{N})$, by the choice of σ. For each pair of markings m and m', such that $\mathrm{m} \in \Theta(\mathcal{N})$ and $\mathrm{m} \xrightarrow{t} \mathrm{m}'$, for some transition $t \in T$, we have $\mathrm{m}' = (\mathrm{m} \setminus {}^\bullet t) \cup t^\bullet$ and $\|\sigma \cap {}^\bullet t\| = \|\sigma \cap t^\bullet\| \leq 1$, hence $\|\mathrm{m}' \cap \sigma\| = 1$ and $\mathrm{m}' \in \Omega(\mathcal{N})$, because σ 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 $\mathit{Reach}(\mathcal{N})$ is known to be the least inductive invariant of $\mathcal{N}$, we obtain $\mathit{Reach}(\mathcal{N}) \subseteq \Theta(\mathcal{N}) \cap \Omega(\mathcal{N})$. The rest of the proof follows from Theorem 2 and the above fact. □

Given a CL sentence ϕ that describes the set of initial configurations of a system, Figure 10 introduces $\mathsf{WS}\kappa S$ formulæ that define the trap and mutex invariants of the marked Petri nets $(\mathsf{N}(\Sigma, \alpha_\mathcal{T}, \beta), \mathrm{m}_0)$, where $\mathcal{T} \in \mathbb{T}(\phi)$ are rewriting trees and $(\alpha_\mathcal{T}, \mathrm{m}_0) \models \phi$ are initial configurations. This construction uses the flow formula (Figure 9) derived from the SID that interprets the predicate symbols from ϕ and the behavior map β of the system. Here we denote by $\mathbf{X}$ (resp.

$$Init(\mathbf{Y},\mathbf{R}) \stackrel{\text{def}}{=} \mathrm{m}(\mathbf{Y},\mathbf{R}) \wedge \forall x \,.\, \bigwedge_{q\in\mathcal{Q}_\beta} \Big(Y_q(x) \leftrightarrow \bigvee_{\substack{\ell\in[1,P] \text{ such that} \\ \mathsf{C}^q(z) \text{ occurs in } \rho_\ell}} R_\ell(x)\Big) \tag{29}$$

$$\theta(\mathbf{X},\mathbf{R}) \stackrel{\text{def}}{=} \forall\mathbf{Y}\forall\mathbf{Z} \,.\, \Phi(\mathbf{Y},\mathbf{Z},\mathbf{R}) \rightarrow (\|\mathbf{X}\cap\mathbf{Y}\| \geq 1 \rightarrow \|\mathbf{X}\cap\mathbf{Z}\| \geq 1) \tag{30}$$

$$\Theta(\mathbf{X},\mathbf{Y},\mathbf{R}) \stackrel{\text{def}}{=} \mathrm{m}(\mathbf{X},\mathbf{R}) \wedge \forall\mathbf{Z} \,.\, (\|\mathbf{Y}\cap\mathbf{Z}\| \geq 1 \wedge \theta(\mathbf{Z},\mathbf{R})) \rightarrow \|\mathbf{X}\cap\mathbf{Z}\| \geq 1 \tag{31}$$

$$\omega(\mathbf{X},\mathbf{R}) \stackrel{\text{def}}{=} \forall\mathbf{Y}\forall\mathbf{Z} \,.\, \Phi(\mathbf{Y},\mathbf{Z},\mathbf{R}) \rightarrow \big((\|\mathbf{X}\cap\mathbf{Y}\| = 0 \leftrightarrow \|\mathbf{X}\cap\mathbf{Z}\| = 0) \wedge (\|\mathbf{X}\cap\mathbf{Y}\| = 1 \leftrightarrow \|\mathbf{X}\cap\mathbf{Z}\| = 1)\big) \tag{32}$$

$$\Omega(\mathbf{X},\mathbf{Y},\mathbf{R}) \stackrel{\text{def}}{=} \mathrm{m}(\mathbf{X},\mathbf{R}) \wedge \forall\mathbf{Z} \,.\, (\|\mathbf{Y}\cap\mathbf{Z}\| = 1 \wedge \omega(\mathbf{Y},\mathbf{R})) \rightarrow \|\mathbf{X}\cap\mathbf{Z}\| = 1 \tag{33}$$

Figure 10: Initial Configurations and Structural Invariants

$\mathbf{Y}$ and $\mathbf{Z}$) the tuple of second-order variables $\langle X_q \mid q \in \mathcal{Q}_\beta\rangle$ (resp. $\langle Y_q \mid q \in \mathcal{Q}_\beta\rangle$ and $\langle Z_q \mid q \in \mathcal{Q}_\beta\rangle$) 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. $\bigcup_{q\in\mathcal{Q}_\beta} X_q \cap Z_q$), where the definitions of set union and intersection are standard in MSO and omitted to avoid clutter.

Let ν be a model of $\mathit{RewrTree}_\Delta(\mathbf{R})$, meaning that $\mathcal{T}_\nu^{\mathbf{R}}$ is a rewriting tree for ϕ. We recall that, by letting $\rho_1 \stackrel{\text{def}}{=} (\mathcal{A}_\phi() \leftarrow \phi)$ be the first rule in the SID, we capture the fact that $\mathcal{T}_\nu^{\mathbf{R}} \in \mathbb{T}(\phi)$, for each model ν of $\mathit{RewrTree}_\Delta(\mathbf{R})$ (see the second conjunct in line 15 from Figure 6). The formula $\mathit{Init}(\mathbf{Y},\mathbf{R})$ (29) encodes the fact that $\sigma_\nu^{\mathbf{Y}}$ is an initial marking of the Petri net $\mathsf{N}(\Sigma,\alpha_{\mathcal{T}_\nu^{\mathbf{R}}},\beta)$ 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 $\mathsf{C}^q(z)$ occurs in the rule that labels the node.

We assume further that ν is a model of $\mathit{RewrTree}_\Delta(\mathbf{R}) \wedge \mathit{Init}(\mathbf{Y},\mathbf{R})$. 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 $(\mathsf{N}(\Sigma,\alpha_{\mathcal{T}_\nu^{\mathbf{R}}},\beta),\sigma_\nu^{\mathbf{Y}})$, respectively. Similarly, $\omega(\mathbf{X},\mathbf{R})$ (32) and $\Omega(\mathbf{X},\mathbf{R})$ (33) define the mutexes and the mutex invariant of $(\mathsf{N}(\Sigma,\alpha_{\mathcal{T}_\nu^{\mathbf{R}}},\beta),\sigma_\nu^{\mathbf{Y}})$, respectively. We intentionally use the same symbols to denote trap (mutex) invariants and their defining $\mathsf{WS}\kappa S$ 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* $\models^{\nu}_{\mathsf{wsks}} \mathit{RewrTree}_\Delta(\mathbf{R}) \wedge \mathit{Init}(\mathbf{Y},\mathbf{R})$ *then the following hold:*

1. $\Theta(\mathsf{N}(\Sigma,\alpha_{\mathcal{T}_\nu^{\mathbf{R}}},\beta),\sigma_\nu^{\mathbf{Y}}) = \{\sigma_{\nu'}^{\mathbf{X}} \mid \models^{\nu'}_{\mathsf{wsks}} \Theta(\mathbf{X},\mathbf{Y},\mathbf{R}),\ \nu'(\mathbf{Y},\mathbf{R}) = \nu(\mathbf{Y},\mathbf{R})\}$,
2. $\Omega(\mathsf{N}(\Sigma,\alpha_{\mathcal{T}_\nu^{\mathbf{R}}},\beta),\sigma_\nu^{\mathbf{Y}}) = \{\sigma_{\nu'}^{\mathbf{X}} \mid \models^{\nu'}_{\mathsf{wsks}} \Omega(\mathbf{X},\mathbf{Y},\mathbf{R}),\ \nu'(\mathbf{Y},\mathbf{R}) = \nu(\mathbf{Y},\mathbf{R})\}$.

Proof. We give the proof for point (1), the proof for (2) uses a similar argument and is left for the reader.

"$\subseteq$" Let $\mathrm{m} \in \Theta(\mathsf{N}(\Sigma,\alpha_{\mathcal{T}_\nu^{\mathbf{R}}},\beta),\sigma_\nu^{\mathbf{Y}})$ be a marking and ν' be a valuation such that $\sigma_{\nu'}^{\mathbf{X}} = \mathrm{m}$ and ν' agrees with ν over $\mathbf{Y}$ and $\mathbf{R}$. Clearly, such a valuation exists

and, in order to show that $\mathrm{m} \in \{\sigma^{\mathbf{X}}_{\nu'} \mid \models^{\nu'}_{\mathsf{wsks}} \Theta(\mathbf{X},\mathbf{Y},\mathbf{R}),\ \nu'(\mathbf{Y},\mathbf{R}) = \nu(\mathbf{Y},\mathbf{R})\}$, it suffices to prove $\models^{\nu'}_{\mathsf{wsks}} \Theta(\mathbf{X},\mathbf{Y},\mathbf{R})$. Because $\sigma^{\mathbf{X}}_{\nu'} = \mathrm{m}$, we have $\models^{\nu'}_{\mathsf{wsks}} \mathrm{m}(\mathbf{X},\mathbf{R})$ and we are left with proving $\models^{\nu'}_{\mathsf{wsks}} \forall \mathbf{Z}\ .\ \theta(\mathbf{Z},\mathbf{R}) \wedge \|\mathbf{Y} \cap \mathbf{Z}\| \geq 1 \rightarrow \|\mathbf{X} \cap \mathbf{Z}\| \geq 1$. To this end, let μ be any valuation that agrees with ν' over $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{R}$, such that $\models^{\mu}_{\mathsf{wsks}} \theta(\mathbf{Z},\mathbf{R}) \wedge \|\mathbf{Y} \cap \mathbf{Z}\| \geq 1$. It suffices to prove that $\sigma^{\mathbf{Z}}_{\mu}$ is a trap of the marked Petri net $(\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^{\mathbf{R}}_{\nu}}, \beta), \sigma^{\mathbf{Y}}_{\nu})$ to obtain $\models^{\mu}_{\mathsf{wsks}} \|\mathbf{X} \cap \mathbf{Z}\| \geq 1$, using the fact that $\mathrm{m} \in \Theta(\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^{\mathbf{R}}_{\nu}}, \beta), \sigma^{\mathbf{Y}}_{\nu})$ and $\sigma^{\mathbf{X}}_{\mu} = \mathrm{m}$. Note that, since μ agrees with ν' over $\mathbf{X}$, we also have $\sigma^{\mathbf{X}}_{\mu} = \sigma^{\mathbf{X}}_{\nu'} = \mathrm{m}$. Since $\models^{\mu}_{\mathsf{wsks}} \|\mathbf{Y} \cap \mathbf{Z}\| \geq 1$ and μ agrees with ν over $\mathbf{Y}$, we have $\sigma^{\mathbf{Z}}_{\mu} \cap \sigma^{\mathbf{Y}}_{\nu} \neq \emptyset$. Let t be a transition of $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^{\mathbf{R}}_{\nu}}, \beta)$, such that $\sigma^{\mathbf{Z}}_{\mu} \cap {}^{\bullet}t \neq \emptyset$ and let μ' be a valuation that agrees with μ over $\mathbf{Z}$, such that $\sigma^{\mathbf{Y}'}_{\mu'} = {}^{\bullet}t$ and $\sigma^{\mathbf{Z}'}_{\mu} = t^{\bullet}$, where $\mathbf{Y}'$ and $\mathbf{Z}'$ are distinct copies of $\mathbf{Y}$ and $\mathbf{Z}$, respectively. By Lemma 9, we obtain $\models^{\mu'}_{\mathsf{wsks}} \Phi(\mathbf{Y}',\mathbf{Z}',\mathbf{R})$. Since $\models^{\mu}_{\mathsf{wsks}} \theta(\mathbf{Z},\mathbf{R})$ and μ' agrees with μ over $\mathbf{Z}$, we obtain $\models^{\mu'}_{\mathsf{wsks}} \|\mathbf{Z} \cap \mathbf{Z}'\| \geq 1$, thus $\sigma^{\mathbf{Z}}_{\mu} \cap \sigma^{\mathbf{Z}'}_{\mu'} = \sigma^{\mathbf{Z}}_{\mu} \cap t^{\bullet} \neq \emptyset$. We have showed that $\sigma^{\mathbf{Z}}_{\mu}$ is a trap of the marked Petri net $(\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^{\mathbf{R}}_{\nu}}, \beta), \sigma^{\mathbf{Y}}_{\nu})$, which concludes this direction of the proof.

"$\supseteq$" Let ν' be a valuation that agrees with ν over $\mathbf{Y}$ and $\mathbf{R}$, such that $\models^{\nu'}_{\mathsf{wsks}} \Theta(\mathbf{X},\mathbf{Y},\mathbf{R})$. To show that $\sigma^{\mathbf{X}}_{\nu'} \in \Theta(\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^{\mathbf{R}}_{\nu}}, \beta), \sigma^{\mathbf{Y}}_{\nu})$, we must prove that $\sigma^{\mathbf{X}}_{\nu'} \cap \theta \neq \emptyset$, for each trap θ of the marked Petri net $(\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^{\mathbf{R}}_{\nu}}, \beta), \sigma^{\mathbf{Y}}_{\nu})$. Let θ be such a trap and let μ be a valuation that agrees with ν' over $\mathbf{X}$, $\mathbf{Y}$ and $\mathbf{R}$, such that $\sigma^{Z}_{\mu} = \theta$. Since σ^{Z}_{μ} is a trap of the marked Petri net $(\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^{\mathbf{R}}_{\nu}}, \beta), \sigma^{\mathbf{Y}}_{\nu})$, we obtain:

- $\models^{\mu}_{\mathsf{wsks}} \|\mathbf{Y} \cap \mathbf{Z}\| \geq 1$, since $\models^{\nu}_{\mathsf{wsks}} \mathit{Init}(\mathbf{Y},\mathbf{R})$ and μ agrees with ν over $\mathbf{Y}$, and
- $\models^{\mu}_{\mathsf{wsks}} \theta(\mathbf{X},\mathbf{R})$, by Lemma 9, since μ agrees with ν over $\mathbf{X}$ and $\mathbf{R}$.

Since $\models^{\mu}_{\mathsf{wsks}} \Theta(\mathbf{X},\mathbf{Y},\mathbf{R})$, we obtain that $\models^{\mu}_{\mathsf{wsks}} \|\mathbf{X} \cap \mathbf{Z}\| \geq 1$, thus $\sigma^{\mathbf{X}}_{\mu} \cap \sigma^{\mathbf{Z}}_{\mu} = \sigma^{\mathbf{X}}_{\nu'} \cap \theta \neq \emptyset$, 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 $\mathsf{WS}\kappa S$. 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 $\mathsf{N} = (S,T,E)$ if and only if ${}^{\bullet}t \not\subseteq \mathrm{m}$, 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 $\mathit{DeadLock}(\mathbf{X},\mathbf{R})$ 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 $\mathsf{N}(\Sigma, \alpha_{\mathcal{T}^{\mathbf{R}}_{\nu}}, \beta)$, where ν is a model of $\mathit{RewrTree}_{\Delta}(\mathbf{R})$ (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 $\mathbf{X}$.

$$\mathit{DeadLock}(\mathbf{X},\mathbf{R}) \stackrel{\text{def}}{=} \forall\mathbf{Y}\forall\mathbf{Z}\ .\ \Phi(\mathbf{Y},\mathbf{Z},\mathbf{R}) \rightarrow \neg\mathbf{Y} \subseteq \mathbf{X} \tag{34}$$

$$\mathit{ErrSet}_{\langle q_1,\ldots,q_n\rangle}(\mathbf{X}) \stackrel{\text{def}}{=} \bigwedge_{q\in\mathcal{Q}_\beta} \|X_q\| \geq \|\{i \in [1,k] \mid q_i = q\}\| \tag{35}$$

Figure 11: Error Configurations for the Deadlock and Reachability Problems

The set defined by the $\mathit{ErrSet}_{\langle q_1,\ldots,q_n\rangle}(\mathbf{X})$ formula (35) captures the fact that each configuration encoded by $\mathbf{X}$ contains pairwise distinct places $q_1[w_1], \ldots, q_n[w_n]$ which is, essentially, the condition required by a reachability query. Note that, because the tuple of states $\langle q_1, \ldots, q_n\rangle$ is part of the input of the query, the number $\|\{i \in [1,k] \mid q_i = q\}\|$ 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 ϕ, interpreted over a SID Δ, a behavior map β and a tuple of states $\langle q_1, \ldots, q_n\rangle$, the following hold:*

1. *$\mathit{DeadLock}(\mathbf{X},\mathbf{R}) \wedge \Theta(\mathbf{X},\mathbf{Y},\mathbf{R}) \wedge \Omega(\mathbf{X},\mathbf{Y},\mathbf{R}) \wedge \mathit{Init}(\mathbf{Y},\mathbf{R}) \wedge \mathit{RewrTree}_\Delta(\mathbf{R})$ is unsatisfiable only if $\mathit{deadlock}(\phi,\Delta,\beta)$ has a negative answer.*
2. *$\mathit{ErrSet}_{\langle q_1,\ldots q_n\rangle}(\mathbf{X}) \wedge \Theta(\mathbf{X},\mathbf{Y},\mathbf{R}) \wedge \Omega(\mathbf{X},\mathbf{Y},\mathbf{R}) \wedge \mathit{Init}(\mathbf{Y},\mathbf{R}) \wedge \mathit{RewrTree}_\Delta(\mathbf{R})$ is unsatisfiable only if $\mathit{reach}(\phi,\langle q_1,\ldots,q_n\rangle,\Delta,\beta)$ has a negative answer.*

Proof. (1) To prove the contrapositive statement, assume that $\mathit{deadlock}(\phi,\Delta,\beta)$ has a positive answer. By Lemma 10, the query $\mathit{deadlock}^\sharp_i(\phi,\Delta,\beta)$ has a positive answer as well, thus there exists a rewriting tree $\mathcal{T} \in \mathbb{T}(\phi)$ and a marking $\mathrm{m} \in \Theta(\mathsf{N}(\Sigma,\alpha_\mathcal{T},\beta),\mathrm{m}_\mathcal{T}) \cap \Omega(\mathsf{N}(\Sigma,\alpha_\mathcal{T},\beta),\mathrm{m}_\mathcal{T}) \cap \mathit{Dead}(\mathsf{N}(\Sigma,\alpha_\mathcal{T},\beta))$, where $(\alpha_\mathcal{T},\mathrm{m}_\mathcal{T})$ is the canonical model of ϕ corresponding to $\mathcal{T}$. Let ν be a valuation such that $\mathcal{T}^{\mathbf{R}}_\nu = \mathcal{T}$, $\sigma^{\mathbf{Y}}_\nu = \mathrm{m}_\mathcal{T}$ and $\sigma^{\mathbf{X}}_\nu = \mathrm{m}$. 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:

- $\models^\nu_{\mathsf{wsks}} \mathit{Init}(\mathbf{Y},\mathbf{R})$ follows directly from the definitions of $\mathrm{m}_\mathcal{T}$ (Def. 7) and $\mathit{Init}(\mathbf{Y},\mathbf{R})$ (29).
- $\models^\nu_{\mathsf{wsks}} \Theta(\mathbf{X},\mathbf{Y},\mathbf{R})$ follows from point 1 of Lemma 11, because $\sigma^{\mathbf{X}}_\nu = \mathrm{m}$, $\mathcal{T}^{\mathbf{R}}_\nu = \mathcal{T}$ and $\mathrm{m} \in \Theta(\mathsf{N}(\Sigma,\alpha_\mathcal{T},\beta),\mathrm{m}_\mathcal{T})$.
- $\models^\nu_{\mathsf{wsks}} \Omega(\mathbf{X},\mathbf{Y},\mathbf{R})$ follows from point 2 of Lemma 11, because $\sigma^{\mathbf{X}}_\nu = \mathrm{m}$, $\mathcal{T}^{\mathbf{R}}_\nu = \mathcal{T}$ and $\mathrm{m} \in \Omega(\mathsf{N}(\Sigma,\alpha_\mathcal{T},\beta),\mathrm{m}_\mathcal{T})$.
- $\models^\nu_{\mathsf{wsks}} \mathit{DeadLock}(\mathbf{X},\mathbf{R})$ because $\mathit{Dead}(\mathsf{N}(\Sigma,\alpha_\mathcal{T},\beta))$ is the set of markings $\sigma^{\mathbf{X}}_{\nu'}$ of $\mathsf{N}(\Sigma,\alpha_\mathcal{T},\beta)$, such that $\models^{\nu'}_{\mathsf{wsks}} \mathit{DeadLock}(\mathbf{X},\mathbf{R}) \wedge \mathit{RewrTree}_\Delta(\mathbf{R})$. This follows directly from the definition of a deadlock marking and the definition of $\mathit{DeadLock}(\mathbf{X},\mathbf{R})$ (34), using the characterization of the pre- and post-sets of the transitions from $\mathsf{N}(\Sigma,\alpha_\mathcal{T},\beta)$ 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 $\{q_i[u_i] \mid i \in [1,n]\} \subseteq$

m, with $q_1[u_1], \ldots, q_n[u_n]$ pairwise distinct coincides with the set $\{\sigma^{\mathbf{x}}_{\nu'} \mid \models^{\nu'}_{\mathsf{wsks}} ErrSet_{\langle q_1,\ldots,q_n\rangle}(\mathbf{X}) \wedge \mathrm{m}(\mathbf{X},\mathbf{R}) \wedge RewrTree_{\Delta}(\mathbf{R})\}$. □

Since the $\mathsf{WS}\kappa S$ 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 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].

[6] For instance, the tree-shaped architecture with leaves linked in a ring §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 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 $\mathsf{WS}\kappa S$ 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 $\mathsf{WS}\kappa S$ formulæ, for which optimized solvers exist. The whole generation process involves transformations from configuration specifications into rewriting trees and then into $\mathsf{WS}\kappa S$ 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.