Checking Deadlock-Freedom of Parametric Component-Based Systems[⋆]

Marius Bozga, Radu Iosif and Joseph Sifakis

Univ. Grenoble Alpes, CNRS, Grenoble INP[⋆⋆⋆], VERIMAG, 38000 Grenoble France
{Marius.Bozga,Radu.Iosif,Joseph.Sifakis}@univ-grenoble-alpes.fr

Abstract. We propose an automated method for computing inductive invariants used to proving deadlock freedom of parametric component-based systems. The method generalizes the approach for computing structural trap invariants from bounded to parametric systems with general architectures. It symbolically extracts trap invariants from interaction formulae defining the system architecture. The paper presents the theoretical foundations of the method, including new results for the first order monadic logic and proves its soundness. It also reports on a preliminary experimental evaluation on several textbook examples.

Modern computing systems exhibit dynamic and reconfigurable behavior. To tackle the complexity of such systems, engineers extensively use architectures that enforce, by construction, essential properties, such as fault tolerance or mutual exclusion. Architectures can be viewed as parametric operators that take as arguments instances of components of given types and enforce a characteristic property. For instance, client-server architectures enforce atomicity and resilience of transactions, for any numbers of clients and servers. Similarly, token-ring architectures enforce mutual exclusion between any number of components in the ring.

Parametric verification is an extremely relevant and challenging problem in systems engineering. In contrast to the verification of bounded systems, consisting of a known set of components, there exist no general methods and tools succesfully applied to parametric systems. Verification problems for very simple parametric systems, even with finite-state components, are typically intractable [15,9]. Most work in this area puts emphasis on limitations determined mainly by three criteria (1) the topology of the architecture, (2) the coordination primitives, and (3) the properties to be verified.

The main decidability results reduce parametric verification to the verification of a bounded number of instances of finite state components. Several methods try to determine a cut-off size of the system, i.e. the minimal size for which if a property holds, then it holds for any size, e.g. Suzuki [19], Emerson and Namjoshi [14]. Other methods identify systems with well-structured transition relations, for which symbolic enumeration of reachable states is feasible [1] or reduce to known decidable problems, such as reachability in vector addition systems [15]. Typically, these methods apply to systems with

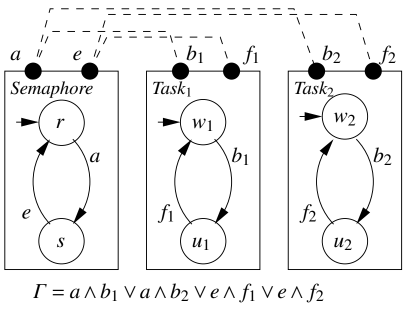
[⋆⋆⋆] Institute of Engineering Univ. Grenoble Alpes

[⋆] The research leading to these results has received funding from the European Union Horizon 2020 research and innovation programme under grant agreement no. 700665 CITADEL (Critical Infrastructure Protection using Adaptive MILS) and no. 730086 ERGO (European Robotic Goal-Oriented Autonomous Controller).

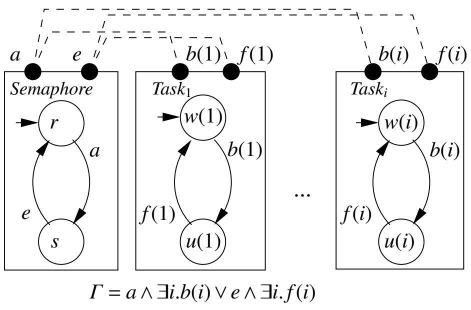
global coordination. When theoretical decidability is not of concern, semi-algorithmic techniques such as *regular model checking* [16,2], SMT-based *bounded model checking* [3,13], *abstraction* [7,10] and *automata learning* [12] can be used to deal with more general classes of The interested reader can find a complete survey on parameterized model checking by Bloem et al. [9].

This paper takes a different angle of attack to the verification problem, seeking generality of the type of parametric systems and focusing on the verification of a particular but essential property: *deadlock-freedom*. The aim is to come up with effective methods for checking deadlock-freedom, by overcoming the complexity blowup stemming from the effective generation of reachability sets. We briefly describe our approach below.

A system is the composition of a finite number of component instances of given types, using interactions that follow the Behaviour-Interaction-Priorities (BIP) paradigm [6]. To simplify the technical part, we assume that components and interactions are finite abstractions of real-life systems. An instance is a finite-state transition system whose edges are labeled by ports. The instances communicate synchronously via a number of simultaneous interactions involving a set of ports each, such that no data is exchanged during interactions. If the number of instances in the system is fixed and known in advance, we say that the system is *bounded*, otherwise it is *parametric*.



(a) Bounded System



(b) Parametric System

Fig. 1: Mutual Exclusion Example

For instance, the bounded system in Figure 1a consist of component types *Semaphore*, with one instance, and *Task*, with two instances. A semaphore goes from the free state r to the taken state s by an acquire action a, and viceversa from s to r by a release action e. A task goes from waiting w to busy u by action b and viceversa, by action f. For the bounded system in Figure 1a, the interactions are $\{a,b_1\},\{a,b_2\},\{e,f_1\}$ and $\{e,f_2\}$, depicted with dashed lines. Since the number of instances is known in advance, we can view an interaction as a minimal satisfying valuation of the boolean formula $\Gamma = (a \wedge b_1) \vee (a \wedge b_2) \vee (e \wedge f_1) \vee (e \wedge f_2)$, where the port symbols are propositional variables. Because every instance has finitely many states, we can write a boolean formula $\Delta = [\neg r \vee \neg(w_1 \vee w_2)] \wedge [\neg s \vee \neg(u_1 \vee u_2)]$, this time over propositional state variables, which defines the configurations in which all interactions are disabled (deadlock). Proving that no deadlock configuration is reachable from the initial configuration $r \wedge w_1 \wedge w_2$, requires finding an over-approximation (invariant) I of the reachable configurations, such that the conjunction $I \wedge \Delta$ is not satisfiable.

The basic idea of our method, supported by the D-Finder deadlock detection tool [8] for bounded component-based systems, is to compute an invariant straight from the interaction formula, without going through costly abstract fixpoint iterations. The invariants we are looking for are in fact solutions of a system of boolean constraints $\Theta(\Gamma)$, of size linear in the size of Γ (written in DNF). In our example, $\Theta(\Gamma) = \bigwedge_{i=1,2}(r \vee w_i) \leftrightarrow (s \vee u_i)$. Finding the (minimal) solutions of this constraint can be done, as currently implemented in D-Finder, by exhaustive model enumeration using a SAT solver. Here we propose a more efficient solution, which consists in writing $\Theta(\Gamma)$ in DNF and remove the negative literals from each minterm. In our case, this gives the invariant $I = (r \vee s) \wedge \bigwedge_{i=1,2}(w_i \vee u_i) \wedge (r \vee u_1 \vee u_2) \wedge (s \vee w_1 \vee w_2)$ and $I \wedge \Delta$ is proved unsatisfiable using a SAT solver.

The main contribution of this paper is the generalization of this invariant generation method to the parametric case. To understand the problem, consider the parametric system from Figure 1, in which a *Semaphore* interacts with *n Tasks*, where $n > 0$ is not known in advance. The interactions are described by a fragment of first order logic, in which the ports are either propositional or monadic predicate symbols, in our case $\Gamma = a \wedge \exists i \,.\, b(i) \vee e \wedge \exists i \,.\, f(i)$. This logic, called *Monadic Interaction Logic* (MIL), is also used to express the constraints $\Theta(\Gamma)$ and compute their solutions. In our case, we obtain $I = (r \vee s) \wedge [\forall i \,.\, w(i) \vee u(i)] \wedge [r \vee \exists i \,.\, u(i)] \wedge [s \vee \exists i \,.\, w(i)]$. As in the bounded case, we can give a parametric description of deadlock configurations $\Delta = [\neg r \vee \neg\exists i \,.\, w(i)] \wedge [\neg s \vee \neg\exists i \,.\, u(i)]$ and prove that $I \wedge \Delta$ is unsatisfiable, using the decidability of MIL, based on an early small model property result due to Löwenheim [18]. In practice, we avoid the model enumeration suggested by this result and check the satisfiability of such queries using a decidable theory of sets with cardinality constraints [17], available in the CVC4 SMT solver [4].

The paper is structured as follows: §1 presents existing results for checking deadlock-freedom of bounded systems using invariants, §2 formalizes the approach for computing invariants using MIL, §3 introduces cardinality constraints for invariant generation, §4 presents the integration of the above results within a verification technique for parametric systems and §5 reports on preliminary experiments carried out with a prototype tool. Finally, §6 presents concluding remarks and future work directions. For reasons of space, all proofs are given in [11].

1 Bounded Component-based Systems

A *component* is a tuple $C = \langle \mathsf{P}, \mathsf{S}, s_0, \Delta \rangle$, where $\mathsf{P} = \{p, q, r, \ldots\}$ is a finite set of *ports*, S is a finite set of *states*, $s_0 \in \mathsf{S}$ is an initial state and $\Delta \subseteq \mathsf{S} \times \mathsf{P} \times \mathsf{S}$ is a set of *transitions* written $s \xrightarrow{p} s'$. To simplify the technical details, we assume there are *no two different transitions with the same port*, i.e. if $s_1 \xrightarrow{p_1} s'_1, s_2 \xrightarrow{p_2} s'_2 \in \Delta$ and $s_1 \neq s_2$ or $s'_1 \neq s'_2$ then $p_1 \neq p_2$. In general, this restriction can be lifted, at the cost of cluttering the presentation.

A *bounded system* $\mathcal{S} = \langle C^1, \ldots, C^n, \Gamma \rangle$ consists of a fixed number (n) of components $C^k = \langle \mathsf{P}^k, \mathsf{S}^k, {s_0}^k, \Delta^k \rangle$ and an *interaction formula* Γ, describing the allowed interactions. Since the number of components is known in advance, we write interaction formulae using boolean logic over the set of propositional variables $\mathsf{BVar} \overset{\text{def}}{=} \bigcup_{k=1}^{n}(\mathsf{P}^k \cup \mathsf{S}^k)$. Here we intentionally use the names of states and ports as propositional variables.

A *boolean interaction formula* is either $a \in \mathsf{BVar}$, $f_1 \wedge f_2$ or $\neg f_1$, where f_i are formulae, for $i = 1,2$, respectively. We define the usual shorthands $f_1 \vee f_2 \stackrel{\mathsf{def}}{=} \neg(\neg f_1 \wedge \neg f_2)$, $f_1 \rightarrow f_2 \stackrel{\mathsf{def}}{=} \neg f_1 \vee f_2$, $f_1 \leftrightarrow f_2 \stackrel{\mathsf{def}}{=} (f_1 \rightarrow f_2) \wedge (f_2 \rightarrow f_1)$. A literal is either a variable or its negation and a minterm is a conjunction of literals. A formula is in disjunctive normal form (DNF) if it is written as $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} \ell_{ij}$, where ℓ_{ij} is a literal. A formula is *positive* if and only if each variable occurs under an even number of negations, or, equivalently, its DNF forms contains no negative literals. We assume interaction formulae of bounded systems to be always positive.

A *boolean valuation* $\beta : \mathsf{BVar} \rightarrow \{\top, \bot\}$ maps each propositional variable to either true ($\top$) or false ($\bot$). We write $\beta \models f$ if and only if $f = \top$, when replacing each boolean variable a with $\beta(a)$ in f. We say that β is a *model* of f in this case and write $f \equiv g$ for $[\![f]\!] = [\![g]\!]$, where $[\![f]\!] \stackrel{\mathsf{def}}{=} \{\beta \mid \beta \models f\}$. Given two valuations β_1 and β_2 we write $\beta_1 \subseteq \beta_2$ if and only if $\beta_1(a) = \top$ implies $\beta_2(a) = \top$, for each variable $a \in \mathsf{BVar}$. We write $f \equiv^{\mu} g$ for $[\![f]\!]^{\mu} = [\![g]\!]^{\mu}$, where $[\![f]\!]^{\mu} \stackrel{\mathsf{def}}{=} \{\beta \in [\![f]\!] \mid \text{ for all } \beta' : \beta' \subseteq \beta \text{ and } \beta' \neq \beta \text{ only if } \beta' \notin [\![f]\!]\}$ is the set of minimal models of f.

1.1 Execution Semantics of Bounded Systems

We use 1-safe marked Petri Nets to define the set of executions of a bounded system. A *Petri Net* (PN) is a tuple $N = \langle S, T, E \rangle$, 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*. The elements of $S \cup T$ are called *nodes*. For a node n, let ${}^{\bullet}n \stackrel{\mathsf{def}}{=} \{m \in S \cup T \mid E(m,n) = 1\}$, $n^{\bullet} \stackrel{\mathsf{def}}{=} \{m \in S \cup T \mid E(n,m) = 1\}$ and lift these definitions to sets of nodes, as usual.

A *marking* for a PN $N = \langle S, T, E \rangle$ is a function $\mathrm{m} : S \rightarrow \mathbb{N}$. A *marked Petri net* is a pair $\mathcal{N} = (N, \mathrm{m}_0)$, where m_0 is the *initial marking* of $N = \langle S, T, E \rangle$. We consider that the reader is familiar with the standard execution semantics of a marked PN. A marking m is *reachable* in $\mathcal{N}$ if and only if there exists a sequence of transitions leading fom m_0 to m. We denote by $\mathcal{R}(\mathcal{N})$ the set of reachable markings of $\mathcal{N}$. A set of markings $\mathcal{M}$ is an *invariant* of $\mathcal{N} = (N, \mathrm{m}_0)$ if and only if $\mathrm{m}_0 \in \mathcal{M}$ and $\mathcal{M}$ is closed under the transitions of N. A marked PN $\mathcal{N}$ is 1-*safe* if $\mathrm{m}(s) \leq 1$, for each $s \in S$ and each $\mathrm{m} \in \mathcal{R}(\mathcal{N})$. In the following, we consider only marked PNs that are 1-safe. In this case, any (necessarily finite) set of reachable markings can be defined by a boolean formula, which identifies markings with the induced boolean valuations. A marking m is a *deadlock* if for no transition is enaled in m and let $\mathcal{D}(\mathcal{N})$ be the set of deadlocks of N. A marked PN $\mathcal{N}$ is *deadlock-free* if and only if $\mathcal{R}(\mathcal{N}) \cap \mathcal{D}(\mathcal{N}) = \emptyset$. A sufficient condition for deadlock freedom is $\mathcal{M} \cap \mathcal{D}(\mathcal{N}) = \emptyset$, for some invariant $\mathcal{M}$ of $\mathcal{N}$.

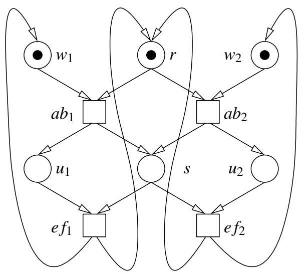


Fig. 2: PN for Mutual Exclusion

In the rest of this section, we fix a bounded system $\mathcal{S} = \langle C^1, \ldots, C^n, \Gamma \rangle$, where $C^k = \langle \mathsf{P}^k, \mathsf{S}^k, {s_0}^k, \Delta^k \rangle$, for all $k \in [1,n]$ and Γ is a positive boolean formula, over propositional variables denoting ports. The set of executions of $\mathcal{S}$ is given by the 1-safe marked PN $\mathcal{N}_{\mathcal{S}} = (N, \mathrm{m}_0)$, where $N = (\bigcup_{i=1}^{n} \mathsf{S}^i, T, E)$, $\mathrm{m}_0(s) = 1$ if and only if $s \in \{{s_0}^i \mid i \in [1,n]\}$ and

T, E are as follows. For each minimal model $\beta \in [[\Gamma]]^\mu$, we have a transition $\mathrm{t}_\beta \in T$ and edges $(s_i, \mathrm{t}_\beta), (\mathrm{t}_\beta, s'_i) \in E$, for all $i \in [1,n]$ such that $s_i \xrightarrow{p_i} s'_i \in \Delta^i$ and $\beta(p_i) = \top$. Moreover, nothing else is in T or E.

For example, the marked PN from Figure 2 describes the set of executions of the bounded system from Figure 1a. Note that each transition of the PN corresponds to a minimal model of the interaction formula $\Gamma = a \wedge b_1 \vee a \wedge b_2 \vee e \wedge f_1 \vee e \wedge f_2$, or equivalently, to the set of (necessarily positive) literals of some minterm in the DNF of Γ.

1.2 Proving Deadlock Freedom of Bounded Systems

A bounded system $\mathcal{S}$ is deadlock-free if and only if its corresponding marked PN $\mathcal{N}_\mathcal{S}$ is deadlock-free. In the following, we prove deadlock-freedom of a bounded system, by defining a class of invariants that are particularly useful for excluding unreachable deadlock markings.

Given a Petri Net $N = (S, T, E)$, a set of places $W \subseteq S$ is called a *trap* if and only if $W^\bullet \subseteq {}^\bullet W$. A trap W of N is a *marked trap* of the marked PN $\mathcal{N} = (N, \mathrm{m}_0)$ if and only if $\mathrm{m}_0(s) = \top$ for some $s \in W$. A *minimal marked trap* is a marked trap such that none of its strict subsets is a marked trap. A marked trap defines an invariant of the PN because some place in the trap will always be marked, no matter which transition is fired. The *trap invariant* of $\mathcal{N}$ is the least set of markings that mark each trap of $\mathcal{N}$. Clearly, the trap invariant of $\mathcal{N}$ subsumes the set of reachable markings of $\mathcal{N}$, because the latter is the least invariant of $\mathcal{N}$ and invariants are closed under intersection[1].

Lemma 1. *Given a bounded system $\mathcal{S}$, the boolean formula:*

$$Trap(\mathcal{N}_\mathcal{S}) \overset{\mathsf{def}}{=} \bigwedge\{\bigvee_{i=1}^k s_i \mid \{s_1, \ldots, s_k\} \textit{ is a marked trap of } \mathcal{N}_\mathcal{S}\}$$

defines an invariant of $\mathcal{N}_\mathcal{S}$.

Next, we describe a method of computing trap invariants that does not explicitly enumerate all the marked traps of a marked PN. First, we consider a *trap constraint* $\Theta(\Gamma)$, derived from the interaction formula Γ, in linear time. By slight abuse of notation, we define, for a given port $p \in \mathsf{P}^i$ of the component C^i, for some $i \in [1,n]$, the pre- and post-state of p in C^i as ${}^\bullet p \overset{\mathsf{def}}{=} s$ and $p^\bullet \overset{\mathsf{def}}{=} s'$, where $s \xrightarrow{p} s'$ is the unique rule[2] involving p in Δ^i, and ${}^\bullet p = p^\bullet \overset{\mathsf{def}}{=} \bot$ if there is no such rule. Assuming that the interaction formula is written in DNF as $\Gamma = \bigvee_{k=1}^N \bigwedge_{\ell=1}^{M_k} p_{k\ell}$, we define the trap constraint:

$$\Theta(\Gamma) \overset{\mathsf{def}}{=} \bigwedge_{k=1}^N \left(\bigvee_{\ell=1}^{M_k} {}^\bullet p_{k\ell}\right) \rightarrow \left(\bigvee_{\ell=1}^{M_k} p_{k\ell}{}^\bullet\right)$$

It is not hard to show[3] that any satisfying valuation of $\Theta(\Gamma)$ defines a trap of $\mathcal{N}_\mathcal{S}$ and, moreover, any such trap is defined in this way. We also consider the formula $Init(\mathcal{S}) \overset{\mathsf{def}}{=} \bigvee_{k=1}^n {s_0}^k$ defining the set of initially marked places of $\mathcal{S}$, and prove the following:

Lemma 2. *Let $\mathcal{S}$ be a bounded system with interaction formula Γ and β be a boolean valuation. Then $\beta \in [[\Theta(\Gamma) \wedge Init(\mathcal{S})]]$ iff $\{s \mid \beta(s) = \top\}$ is a marked trap of $\mathcal{N}_\mathcal{S}$. Moreover, $\beta \in [[\Theta(\Gamma) \wedge Init(\mathcal{S})]]^\mu$ iff $\{s \mid \beta(s) = \top\}$ is a minimal marked trap of $\mathcal{N}_\mathcal{S}$.*

[1] The intersection of two or more invariants is again an invariant.

[2] We have assumed that each port is associated a unique transition rule.

[3] See [**?**] for a proof.

Because $\Theta(\Gamma)$ and $Init(\mathcal{S})$ are boolean formulae, it is, in principle, possible to compute the trap invariant $Trap(\mathcal{N}_{\mathcal{S}})$ by enumerating the (minimal) models of $\Theta(\Gamma) \wedge Init(\mathcal{S})$ and applying the definition from Lemma 1. However, model enumeration is inefficient and, moreover, does not admit generalization for the parametric case, in which the size of the system is unknown. For these reasons, we prefer a computation of the trap invariant, based on two symbolic transformations of boolean formulae, described next.

For a formula f we denote by f^+ the positive formula obtained by deleting all negative literals from the DNF of f. We shall call this operation *positivation*. Second, for a positive boolean formula f, we define the *dual* formula $(f)^\sim$ recursively on the structure of f, as follows: $(f_1 \wedge f_2)^\sim \overset{\mathsf{def}}{=} f_1{}^\sim \vee f_2{}^\sim$, $(f_1 \vee f_2)^\sim \overset{\mathsf{def}}{=} f_1{}^\sim \wedge f_2{}^\sim$ and $a^\sim \overset{\mathsf{def}}{=} a$, for any $a \in \mathsf{BVar}$. Note that $f^\sim$ is equivalent to the negation of the formula obtained from f by substituting each variable a with $\neg a$ in f.

The following theorem gives the main result of this section, the symbolic computation of the trap invariant of a bounded system, directly from its interaction formula.

Theorem 1. *For any bounded system $\mathcal{S}$, with interaction formula Γ, we have:*

$$Trap(\mathcal{N}_{\mathcal{S}}) \equiv \left([\Theta(\Gamma) \wedge Init(\mathcal{S})]^+\right)^\sim$$

Intuitively, any satisfying valuation of $\Theta(\Gamma) \wedge Init(\mathcal{S})$ defines an initially marked trap of $\mathcal{N}_{\mathcal{S}}$ and a minimal such valuation defines a minimal such trap (Lemma 2). Instead of computing the minimal satisfying valuations by model enumeration, we directly cast the above formula in DNF and remove the negative literals. This is essentially because the negative literals do not occur in the propositional definition of a set of places[4]. Then the dualization of this positive formula yields the trap invariants in CNF, as a conjunction over disjunctions of propositional variables corresponding to the places inside a minimal initially marked trap.

Just as any invariants, trap invariants can be used to prove absence of deadlocks in a bounded system. Assuming, as before, that the interaction formula is given in DNF as $\Gamma = \bigvee_{k=1}^{N} \bigwedge_{\ell=1}^{M_k} p_{k\ell}$, we define the set of deadlock markings of $\mathcal{N}_{\mathcal{S}}$ by the formula $\Delta(\Gamma) \overset{\mathsf{def}}{=} \bigwedge_{k=1}^{N} \bigvee_{\ell=1}^{M_k} \neg({}^\bullet p_{k\ell})$. This is the set of configurations in which all interactions are disabled. With this definition, proving deadlock freedom amounts to proving unsatisfiability of a boolean formula.

Corollary 1. *A bounded system $\mathcal{S}$ with interaction formula Γ is deadlock-free if the boolean formula $\left([\Theta(\Gamma) \wedge Init(\mathcal{S})]^+\right)^\sim \wedge \Delta(\Gamma)$ is unsatisfiable.*

2 Parametric Component-based Systems

From now on we shall focus on parametric systems, consisting of a fixed set of component types $C^1, \ldots, C^n$, such that the number of instances of each type is not known in advance. These numbers are given by a function $\mathsf{M} : [1,n] \to \mathbb{N}$, where $\mathsf{M}(k)$ denotes the number of components of type C^k that are active in the system. To simplify the technical

[4] If the DNF is $(p \wedge q) \vee (p \wedge \neg r)$, the dualization would give $(p \vee q) \wedge (p \vee \neg r)$. The first clause corresponds to the trap $\{p, q\}$ (either p or q is marked), but the second does not directly define a trap. However, by first removing the negative literals, we obtain the traps $\{p, q\}$ and $\{r\}$.

presentation of the results, we assume that all instances of a component type are created at once, before the system is started[5]. For the rest of this section, we fix a parametric system $\mathcal{S} = \langle \mathcal{C}^1, \ldots, \mathcal{C}^n, \mathsf{M}, \Gamma \rangle$, where each component type $\mathcal{C}^k = \langle \mathsf{P}^k, \mathsf{S}^k, {s_0}^k, \Delta^k \rangle$ has the same definition as a component in a bounded system and Γ is an interaction formula, written in the fragment of first order logic, defined next.

2.1 Monadic Interaction Logic

For each component type $\mathcal{C}^k$, where $k \in [1,n]$, we assume a set of index variables Var^k and a set of predicate symbols $\mathsf{Pred}^k \stackrel{\text{def}}{=} \mathsf{P}^k \cup \mathsf{S}^k$. Similar to the bounded case, we use state and ports names as monadic (unary) predicate symbols. We also define the sets $\mathsf{Var} \stackrel{\text{def}}{=} \bigcup_{k=1}^n \mathsf{Var}^k$ and $\mathsf{Pred} \stackrel{\text{def}}{=} \bigcup_{k=1}^n \mathsf{Pred}^k$. Moreover, we consider that $\mathsf{Var}^k \cap \mathsf{Var}^\ell = \emptyset$ and $\mathsf{Pred}^k \cap \mathsf{Pred}^\ell = \emptyset$, for all $1 \leq k < \ell \leq n$. For simplicity's sake, we assume that all predicate symbols in Pred are of arity one. For component types $\mathcal{C}^k$, such that $\mathsf{M}(k) = 1$ and predicate symbols $\mathsf{pr} \in \mathsf{Pred}^k$, we shall write pr instead of $\mathsf{pr}(1)$, as in the interaction formula of the system from Figure 1b. The syntax of the *monadic interaction logic* (MIL) is given below:

$$i, j \in \mathsf{Var} \quad \text{index variables}$$
$$\phi := i = j \mid \mathsf{pr}(i) \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid \exists i \,.\, \phi_1$$

where, for each predicate atom $\mathsf{pr}(i)$, if $\mathsf{pr} \in \mathsf{Pred}^k$ and $i \in \mathsf{Var}^\ell$ then $k = \ell$. We use the shorthands $\forall i \,.\, \phi_1 \stackrel{\text{def}}{=} \neg(\exists i \,.\, \neg\phi_1)$ and $\text{distinct}(i_1, \ldots, i_m) \stackrel{\text{def}}{=} \bigwedge_{1 \leq j < \ell \leq m} \neg i_j = i_\ell$[6]. A *sentence* is a formula in which all variables are in the scope of a quantifier. A formula is *positive* if each predicate symbol occurs under an even number of negations. The semantics of MIL is given in terms of structures $\mathcal{I} = (\mathfrak{U}, \nu, \iota)$, where:

- $\mathfrak{U} \stackrel{\text{def}}{=} [1, \max_{k=1}^n \mathsf{M}(k)]$ is the *universe* of instances, over which variables range,
- $\nu : \mathsf{Var} \rightarrow \mathfrak{U}$ is a *valuation* mapping variables to elements of the universe,
- $\iota : \mathsf{Pred} \rightarrow 2^{\mathfrak{U}}$ is an *interpretation* of predicates as subsets of the universe.

For a structure $\mathcal{I} = (\mathfrak{U}, \nu, \iota)$ and a formula ϕ, the satisfaction relation $\mathcal{I} \models \phi$ is defined as:

$$\begin{array}{lll} \mathcal{I} \models \bot \;\Leftrightarrow\; \text{never} & \mathcal{I} \models i = j & \Leftrightarrow\; \nu(i) = \nu(j) \\ \mathcal{I} \models p(i) \;\Leftrightarrow\; \nu(i) \in \iota(p) & \mathcal{I} \models \exists i \,.\, \phi_1 & \Leftrightarrow\; (\mathfrak{U}, \nu[i \leftarrow m], \iota) \models \phi_1 \;\text{ for some } m \in [1, \mathsf{M}(k)] \\ & & \text{provided that } i \in \mathsf{Var}^k \end{array}$$

where $\nu[i \leftarrow m]$ is the valuation that acts as ν, except for i, which is assigned to m. Whenever $\mathcal{I} \models \phi$, we say that $\mathcal{I}$ is a *model* of ϕ. It is known that, if a MIL formula has a model, then it has a model with universe of cardinality at most exponential in the size (number of symbols) of the formula [18]. This result, due to Löwenheim, is among the first decidability results for a fragment of first order logic.

Structures are partially ordered by pointwise inclusion, i.e. for $\mathcal{I}_i = (\mathfrak{U}, \nu_i, \iota_i)$, for $i = 1, 2$, we write $\mathcal{I}_1 \subseteq \mathcal{I}_2$ iff $\iota_1(p) \subseteq \iota_2(p)$, for all $p \in \mathsf{Pred}$ and $\mathcal{I}_1 \subset \mathcal{I}_2$ iff $\mathcal{I}_1 \subseteq \mathcal{I}_2$ and $\mathcal{I}_1 \neq \mathcal{I}_2$. As before, we define the sets $[[\phi]] = \{\mathcal{I} \mid \mathcal{I} \models \phi\}$ and $[[\phi]]^\mu = \{\mathcal{I} \in [[\phi]] \mid \forall \mathcal{I}' \,.\, \mathcal{I}' \subset$

[5] This is not a limitation, since dynamic instance creation can be simulated by considering that all instances are initially in a waiting state, which is left as result of an interaction involving a designated "spawn" port.

[6] Throughout this paper, we consider that $\bigwedge_{i \in I} \phi_i = \top$ if $I = \emptyset$.

$\mathcal{I} \to \mathcal{I}' \notin [[\phi]]\}$ of models and minimal models of a MIL formula, respectively. Given formulae ϕ_1 and ϕ_2, we write $\phi_1 \equiv \phi_2$ for $[[\phi_1]] = [[\phi_2]]$ and $\phi_1 \equiv^\mu \phi_2$ for $[[\phi_1]]^\mu = [[\phi_2]]^\mu$.

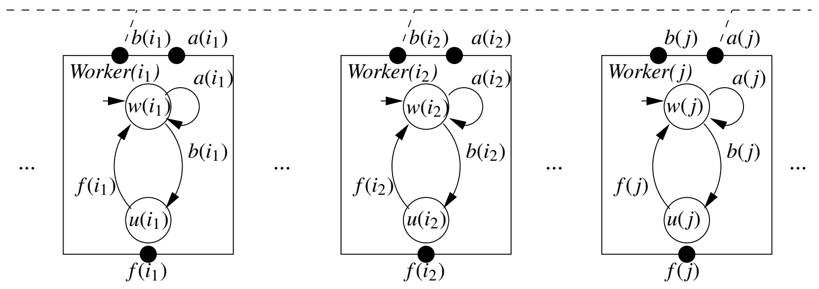
2.2 Execution Semantics of Parametric Systems

We consider the interaction formulae of parametric systems to be finite disjunctions of formulae of the form below:

$$\exists i_1 \ldots \exists i_\ell \wedge \varphi \wedge \bigwedge_{j=1}^{\ell} p_j(i_j) \wedge \bigwedge_{j=\ell+1}^{\ell+m} \forall i_j \,.\, \psi_j \to p_j(i_j) \tag{1}$$

where $\varphi, \psi_{\ell+1}, \ldots, \psi_{\ell+m}$ are conjunctions of equalities and disequalities involving index variables. Intuitively, the formulae (1) state that there are at most ℓ component instances that engage in a multiparty rendez-vous interaction on ports $p_1(i_1), \ldots, p_\ell(i_\ell)$, together with a broadcast to the ports $p_{\ell+1}(i_{\ell+1}), \ldots, p_{\ell+m}(i_{\ell+m})$ of the instances that fulfill the constraints $\psi_{\ell+1}, \ldots, \psi_{\ell+m}$. Observe that, if $m = 0$, the above formula corresponds to a multiparty (generalized) rendez-vous interaction $\exists i_1 \ldots \exists i_\ell \wedge \varphi \wedge \bigwedge_{j=1}^{\ell} p_j(i_j)$. An example of peer-to-peer rendez-vous is the parametric system from Figure 1. Another example of broadcast is given below.

Example 1. Consider the parametric system obtained from an arbitrary number of *Worker* components (Figure 3), where $C^1 = \mathit{Worker}$, $\mathsf{Var}^1 = \{i, i_1, i_2, j\}$ and $\mathsf{Pred}^1 = \{a, b, f, u, w\}$. Any pair of instances can jointly execute the b (*begin*) action provided *all* others are taking the a (*await*) action. Any instance can also execute alone the f (*finish*) action.



$$\Gamma = [\exists i_1 \exists i_2 \,.\, i_1 \neq i_2 \wedge b(i_1) \wedge b(i_2) \wedge \forall j \,.\, j \neq i_1 \wedge j \neq i_2 \to a(j)] \;\vee\; \exists i.f(i)$$

Fig. 3: Parametric System with Broadcast

The execution semantics of a parametric system $\mathcal{S}$ is the marked PN $\mathcal{N}_\mathcal{S} = (N, \mathrm{m}_0)$, where $N = (\bigcup_{k=1}^{n} \mathsf{S}^k \times [1, \mathsf{M}(k)], T, E)$, $\mathrm{m}_0((s_0{}^k, i)) = 1$, for all $k \in [1, n]$ and $i \in [1, \mathsf{M}(k)]$, and the sets of transitions T and edges E are defined next. For each minimal model $\mathcal{I} = (\mathfrak{U}, \nu, \iota) \in [[\Gamma]]^\mu$, we have a transition $\mathrm{t}_\mathcal{I} \in T$ and the edges $((s_i, k), \mathrm{t}_\mathcal{I}), (\mathrm{t}_\mathcal{I}, (s'_i, k)) \in E$ for all $i \in [1, n]$ such that $s_i \xrightarrow{p_i} s'_i \in \Delta^i$ and $k \in \iota(p_i)$. Moreover, nothing else is in T or E.

As a remark, unlike in the case of bounded systems, the size of the marked PN $\mathcal{N}_\mathcal{S}$, that describes the execution semantics of a parametric system $\mathcal{S}$, depends on the maximum number of instances of each component type. The definition of the trap invariant $\mathit{Trap}(\mathcal{N}_\mathcal{S})$ is the same as in the bounded case, except that, in this case, the size of the boolean formula depends on the (unbounded) number of instances in the system. The challenge, addressed in the following, is to define trap invariants using MIL formulae of a fixed size.

2.3 Computing Parametric Trap Invariants

To start with, we define the trap constraint of an interaction formula Γ consisting of a finite disjunction of (1) formulae, as a finite conjunction of formulae of the form below:

$$\forall i_1 \ldots \forall i_\ell \,.\, \left[\varphi \wedge \left(\bigvee_{j=1}^{\ell} {}^\bullet p_j(i_j) \vee \bigvee_{j=\ell+1}^{\ell+m} \exists i_j \,.\, \psi_j \wedge {}^\bullet p_j(i_j)\right)\right] \rightarrow \left[\bigvee_{j=1}^{\ell} p_j{}^\bullet(i_j) \vee \bigvee_{j=\ell+1}^{\ell+m} \exists i_j \,.\, \psi_j \wedge p_j{}^\bullet(i_j)\right]$$

where, for a port $p \in \mathsf{P}^k$ of some component type C^k, ${}^\bullet p(i)$ and $p(i)^\bullet$ denote the unique predicate atoms $s(i)$ and $s'(i)$, such that $s \xrightarrow{p} s' \in \Delta^k$ is the (unique) transition involving p in T^k, or $\bot$ if there is no such rule.

Example 2. For example, the trap constraint for the parametric (rendez-vous) system in Figure 1B is $\forall i.[r \vee w(i)] \rightarrow [s \vee u(i)] \ \wedge \ \forall i.[s \vee u(i)] \rightarrow [r \vee u(i)]$. Analogously, the trap constraint for the parametric (broadcast) system in Figure 3 is:

$$\begin{aligned} &\forall i_1.\forall i_2.\ [i_1 \neq i_2 \wedge (w(i_1) \vee w(i_2) \vee \exists j.(j \neq i_1 \wedge j \neq i_2 \wedge w(j)))] \rightarrow \\ &\qquad [i_1 \neq i_2 \wedge (u(i_1) \vee u(i_2) \vee \exists j.(j \neq i_1 \wedge j \neq i_2 \wedge w(j)))] \\ &\wedge\ \forall i.\ u(i) \rightarrow w(i) \end{aligned}$$

We define a translation of MIL formulae into boolean formulae of unbounded size. Given a function $\mathsf{M} : [1,n] \rightarrow \mathbb{N}$, the *unfolding* of a MIL sentence ϕ is the boolean formula $\mathsf{B}_\mathsf{M}(\phi)$ obtained by replacing each existential [universal] quantifier $\exists i \,.\, \psi(i)$ [$\forall i \,.\, \psi(i)$], for $i \in \mathsf{Var}^k$, by a finite disjunction [conjunction] $\bigvee_{\ell=1}^{\mathsf{M}(k)} \psi[\ell/i]$ [$\bigwedge_{\ell=1}^{\mathsf{M}(k)} \psi[\ell/i]$], where the substitution of the constant $\ell \in \mathsf{M}(k)$ for the variable i is defined recursively as usual, except for $\mathsf{pr}(i)[\ell/i] \overset{\text{def}}{=} (\mathsf{pr}, \ell)$, which is a propositional variable. Further, we relate structures to boolean valuations of unbounded sizes. For a structure $\mathcal{I} = (\mathfrak{U}, \nu, \iota)$ we define the boolean valuation $\beta_\mathcal{I}((\mathsf{pr}, \ell)) = \top$ if and only if $\ell \in \iota(\mathsf{pr})$, for each predicate symbol pr and each integer constant ℓ. Conversely, for each valuation β of the propositional variables (pr, ℓ), there exists a structure $\mathcal{I}_\beta = (\mathfrak{U}, \nu, \iota)$ such that $\iota(\mathsf{pr}) \overset{\text{def}}{=} \{\ell \mid \beta((\mathsf{pr}, \ell)) = \top\}$, for each $\mathsf{pr} \in \mathsf{Pred}$. The following lemma relates the semantics of MIL formulae with that of their boolean unfoldings:

Lemma 3. *Given a* MIL *sentence* ϕ *and a function* $\mathsf{M} : [1,n] \rightarrow \mathbb{N}$, *the following hold:*
1. *for each structure* $\mathcal{I} \in [\![\phi]\!]$, *we have* $\beta_\mathcal{I} \in [\![\mathsf{B}_\mathsf{M}(\phi)]\!]$ *and conversely, for each valuation* $\beta \in [\![\mathsf{B}_\mathsf{M}(\phi)]\!]$, *we have* $\mathcal{I}_\beta \in [\![\phi]\!]$.
2. *for each structure* $\mathcal{I} \in [\![\phi]\!]^\mu$, *we have* $\beta_\mathcal{I} \in [\![\mathsf{B}_\mathsf{M}(\phi)]\!]^\mu$ *and conversely, for each valuation* $\beta \in [\![\mathsf{B}_\mathsf{M}(\phi)]\!]^\mu$, *we have* $\mathcal{I}_\beta \in [\![\phi]\!]^\mu$.

Considering the MIL formula $Init(\mathcal{S}) \overset{\text{def}}{=} \bigvee_{k=1}^{n} \exists i_k \,.\, {s_0}^k(i_k)$, that defines the set of initial configurations of a parametric system $\mathcal{S}$, the following lemma formalizes the intuition behind the definition of parametric trap constraints:

Lemma 4. *Let* $\mathcal{S}$ *be a parametric system with interaction formula* Γ *and* $\mathcal{I}$ *be a structure. Then* $\mathcal{I} \models \Theta(\Gamma) \wedge Init(\mathcal{S})$ *iff* $\{(s,k) \mid k \in \iota(s)\}$ *is a marked trap of* $\mathcal{N}_\mathcal{S}$. *Moreover,* $\mathcal{I} \in [\![\Theta(\Gamma) \wedge Init(\mathcal{S})]\!]^\mu$ *iff* $\{(s,k) \mid k \in \iota(s)\}$ *is a minimal marked trap of* $\mathcal{N}_\mathcal{S}$.

We are currently left with the task of computing a MIL formula which defines the trap invariant $\mathit{Trap}(\mathcal{N}_{\mathcal{S}})$ of a parametric component-based system $\mathcal{S} = \langle \mathcal{C}^1, \ldots, \mathcal{C}^n, \mathsf{M}, \Gamma \rangle$. The difficulty lies in the fact that the size of $\mathcal{N}_{\mathcal{S}}$ and thus, that of the boolean formula $\mathit{Trap}(\mathcal{N}_{\mathcal{S}})$ depends on the number $\mathsf{M}(k)$ of instances of each component type $k \in [1, n]$. As we aim at computing an invariant able to prove safety properties, such as deadlock freedom, independently of how many components are present in the system, we must define the trap invariant using a formula depending exclusively on Γ, i.e. not on M.

Observe first that $\mathit{Trap}(\mathcal{N}_{\mathcal{S}})$ can be equivalently defined using only the minimal marked traps of $\mathcal{N}_{\mathcal{S}}$, which, by Lemma 4, are exactly the sets $\{(s,k) \mid k \in \iota(s)\}$, defined by some structure $(\mathfrak{U}, \nu, \iota) \in [\![\Theta(\Gamma) \wedge \mathit{Init}(\mathcal{S})]\!]^{\mu}$. Assuming that the set of structures $[\![\Theta(\Gamma) \wedge \mathit{Init}(\mathcal{S})]\!]^{\mu}$, or an over-approximation of it, can be defined by a positive MIL formula, the trap invariant is defined using a generalization of boolean dualisation to predicate logic, defined recursively, as follows:

$$(i = j)^{\sim} \stackrel{\mathsf{def}}{=} \neg i = j \quad (\phi_1 \vee \phi_2)^{\sim} \stackrel{\mathsf{def}}{=} \phi_1{}^{\sim} \wedge \phi_2{}^{\sim} \quad (\exists i \,.\, \phi_1)^{\sim} \stackrel{\mathsf{def}}{=} \forall i \,.\, \phi_1{}^{\sim} \quad p(i)^{\sim} \stackrel{\mathsf{def}}{=} p(i)$$

$$(\neg i = j)^{\sim} \stackrel{\mathsf{def}}{=} i = j \quad (\phi_1 \wedge \phi_2)^{\sim} \stackrel{\mathsf{def}}{=} \phi_1{}^{\sim} \vee \phi_2{}^{\sim} \quad (\forall i \,.\, \phi_1)^{\sim} \stackrel{\mathsf{def}}{=} \exists i \,.\, \phi_1{}^{\sim}$$

The crux of the method is the ability of defining, given an arbitrary MIL formula ϕ, a positive MIL formula $\phi^{\oplus}$ that preserve its minimal models, formally $\phi \equiv^{\mu} \phi^{\oplus}$. Because of quantification over unbounded domains, a MIL formula ϕ does not have a disjunctive normal form and thus one cannot define $\phi^{\oplus}$ by simply deleting the negative literals in DNF, as was done for the definition of the positivation operation $(.)^{+}$, in the propositional case. For now we assume that the transformation $(.)^{\oplus}$ of monadic predicate formulae into positive formulae preserving minimal models is defined (a detailed presentation of this step is given next in §3) and close this section with a parametric counterpart of Theorem 1.

Theorem 2. *For any parametric system $\mathcal{S} = \langle \mathcal{C}^1, \ldots, \mathcal{C}^n, \mathsf{M}, \Gamma \rangle$, we have*

$$\mathit{Trap}(\mathcal{N}_{\mathcal{S}}) \equiv \mathsf{B}_{\mathsf{M}}\left(\left((\Theta(\Gamma) \wedge \mathit{Init}(\mathcal{S}))^{\oplus}\right)^{\sim}\right)$$

3 Cardinality Constraints

This section is concerned with the definition of a *positivation* operator $(.)^{\oplus}$ for MIL sentences, whose only requirements are that $\phi^{\oplus}$ is positive and $\phi \equiv^{\mu} \phi^{\oplus}$. For this purpose, we use a logic of quantifier-free *boolean cardinality constraints* [17,4] as an intermediate language, on which the positive formulae are defined. The translation of MIL into cardinality constraints is done by an equivalence-preserving quantifier elimination procedure, described in §3.1. As a byproduct, since the satisfiability of quantifier-free cardinality constraints is NP-complete [17] and integrated with SMT [4], we obtain a practical decision procedure for MIL that does not use model enumeration, as suggested by the small model property [18]. Finally, the definition of a positive MIL formula from a boolean combination of quantifier-free cardinality constraints is given in §3.2.

We start by giving the definition of cardinality constraints. Given the set of monadic predicate symbols Pred, a *boolean term* is generated by the syntax:

$$t := \mathsf{pr} \in \mathsf{Pred} \mid \neg t_1 \mid t_1 \wedge t_2 \mid t_1 \vee t_2$$

When there is no risk of confusion, we borrow the terminology of propositional logic and say that a term is in DNF if it is a disjunction of conjunctions (minterms). We also write $t_1 \rightarrow t_2$ if and only if the implication is valid when t_1 and t_2 are interpreted as boolean formulae, with each predicate symbol viewed as a propositional variable. Two boolean terms t_1 and t_2 are said to be *compatible* if and only if $t_1 \wedge t_2$ is satisfiable, when viewed as a boolean formula.

For a boolean term t and a first-order variable $i \in \mathsf{Var}$, we define the shorthand $t(i)$ recursively, as $(\neg t_1)(i) \stackrel{\mathsf{def}}{=} \neg t_1(i)$, $(t_1 \wedge t_2)(i) \stackrel{\mathsf{def}}{=} t_1(i) \wedge t_2(i)$ and $(t_1 \vee t_2)(i) \stackrel{\mathsf{def}}{=} t_1(i) \vee t_2(i)$. Given a positive integer $n \in \mathbb{N}$ and t a boolean term, we define the following *cardinality constraints*, by MIL formulae:

$$|t| \geq n \stackrel{\mathsf{def}}{=} \exists i_1 \ldots \exists i_n \;.\; \mathrm{distinct}(i_1,\ldots,i_n) \wedge \bigwedge_{j=1}^{n} t(i_j) \qquad |t| \leq n \stackrel{\mathsf{def}}{=} \neg(|t| \geq n+1)$$

We shall further use cardinality constraints with $n = \infty$, by defining $|t| \geq \infty \stackrel{\mathsf{def}}{=} \bot$ and $|t| \leq \infty \stackrel{\mathsf{def}}{=} \top$. The intuitive semantics of cardinality constraints is formally defined in terms of structures $\mathcal{I} = (\mathfrak{U}, \nu, \iota)$ by the semantics of monadic predicate logic, given in the previous. For instance, $|p \wedge q| \geq 1$ means that the intersection of the sets p and q is not empty, whereas $|\neg p| \leq 0$ means that p contains all elements from the universe.

3.1 Quantifier Elimination

Given a sentence ϕ, written in MIL, we build an equivalent boolean combination of cardinality constraints $\mathrm{qe}(\phi)$, using quantifier elimination. We describe the elimination of a single existential quantifier and the generalization to several existential or universal quantifiers is immediate. Assume that $\phi = \exists i_1 \;.\; \bigvee_{k \in K} \psi_k(i_1,\ldots,i_m)$, where K is a finite set of indices and, for each $k \in K$, ψ_k is a quantifier-free conjunction of atomic propositions of the form $i_j = i_\ell$, $\mathsf{pr}(i_j)$ and their negations, for some $j,\ell \in [1,m]$. We write, equivalently, $\phi \equiv \bigvee_{k \in K} \varphi_k \wedge \exists i_1 \;.\; \theta_k(i_1,\ldots,i_m)$, where φ_k does not contain occurrences of i_1 and θ_k is a conjunction of literals of the form $\mathsf{pr}(i_1)$, $\neg\mathsf{pr}(i_1)$, $i_1 = i_j$ and $\neg i_1 = i_j$, for some $j \in [2,m]$. For each $k \in K$, we distinguish the following cases:

1. if $i_1 = i_j$ is a consequence of θ_k, for some $j > 1$, let $\mathrm{qe}(\exists i_1 \;.\; \theta_k) \stackrel{\mathsf{def}}{=} \theta_k[i_j/i_1]$.
2. else, $\theta_k = \bigwedge_{j \in J_k} \neg i_1 = i_j \wedge t_k(i_1)$ for some $J_k \subseteq [2,m]$ and boolean term t_k, and let:

$$\mathrm{qe}(\exists i_1 \;.\; \theta_k) \stackrel{\mathsf{def}}{=} \bigwedge_{J \subseteq J_k} \Big[\mathrm{distinct}(\{i_j\}_{j \in J}) \wedge \bigwedge_{j \in J} t_k(i_j)\Big] \rightarrow |t_k| \geq \|J\| + 1$$
$$\mathrm{qe}(\phi) \stackrel{\mathsf{def}}{=} \bigvee_{k \in K} \varphi_k \wedge \mathrm{qe}(\exists i_1 \;.\; \theta_k)$$

Universal quantification is dealt with using the duality $\mathrm{qe}(\forall i_1 \;.\; \psi) \stackrel{\mathsf{def}}{=} \neg\mathrm{qe}(\exists i_1 \;.\; \neg\psi)$. For a prenex formula $\phi = Q_n i_n \ldots Q_1 i_1 \;.\; \psi$, where $Q_1,\ldots,Q_n \in \{\exists,\forall\}$ and ψ is quantifier-free, we define, recursively $\mathrm{qe}(\phi) \stackrel{\mathsf{def}}{=} \mathrm{qe}(Q_n i_n \;.\; \mathrm{qe}(Q_{n-1} i_{n-1} \ldots Q_1 i_1 \;.\; \psi))$. It is easy to see that, if ϕ is a sentence, $\mathrm{qe}(\phi)$ is a boolean combination of cardinality constraints. The correctness of the construction is a consequence of the following lemma:

Lemma 5. *Given a MIL formula $\phi = Q_n i_n \ldots Q_i i_1 \;.\; \psi$, where $Q_1,\ldots,Q_n \in \{\forall,\exists\}$ and ψ is a quantifier-free conjunction of equality and predicate atoms, we have $\phi \equiv \mathrm{qe}(\phi)$.*

Example 3. (contd. from Example 2) Below we show the results of quantifier elimination applied to the conjunction $\Theta(\Gamma) \wedge \mathit{Init}(\mathcal{S})$ for the system in Figure 1b:

$$\begin{gathered}(\neg r \wedge \neg s \wedge |w \wedge \neg u| \leq 0 \wedge |u \wedge \neg w| \leq 0 \wedge 1 \leq |w|) \vee \\ (\neg r \wedge |w \wedge \neg u| \leq 0 \wedge |\neg w| \leq 0 \wedge 1 \leq |w|) \vee (s \wedge r) \vee (s \wedge |\neg w| \leq 0 \wedge 1 \leq |w|) \vee \\ (\neg s \wedge |\neg u| \leq 0 \wedge |u \wedge \neg w| \leq 0 \wedge 1 \leq |w|) \vee (|\neg u| \leq 0 \wedge |\neg w| \leq 0 \wedge 1 \leq |w|) \;.\end{gathered}$$

Similarly, for the system in Figure 3, we obtain the following cardinality constraints:

$$(3 \leq |w| \wedge |u \wedge \neg w| \leq 0) \vee (2 \leq |w| \wedge |w \wedge \neg u| \leq 1 \wedge |u \wedge \neg w| \leq 0) \vee$$
$$(|\neg u| \leq 1 \wedge |\neg u \wedge \neg w| \leq 0 \wedge |u \wedge \neg w| \leq 0 \wedge 1 \leq |w|) \vee (|w \wedge \neg u| \leq 0 \wedge |u \wedge \neg w| \leq 0 \wedge 1 \leq |w|) \enspace .$$

3.2 Building Positive Formulae that Preserve Minimal Models

Let ϕ be a MIL formula, not necessarily positive. We shall build a positive formula $\phi^{\oplus}$, such that $\phi \equiv^{\mu} \phi^{\oplus}$. By the result of the last section, ϕ is equivalent to a boolean combination of cardinality constraints $\mathrm{qe}(\phi)$, obtained by quantifier elimination. Thus we assume w.l.o.g. that the DNF of ϕ is a disjunction of conjunctions of the form $\bigwedge_{i \in L} |t_i| \geq \ell_i \wedge \bigwedge_{j \in U} |t_j| \leq u_j$, for some sets of indices L, U and some positive integers $\{\ell_i\}_{i \in L}$ and $\{u_j\}_{j \in U}$.

For a boolean combination of cardinality constraints ψ, we denote by $\mathrm{P}(\psi)$ the set of predicate symbols that occur in a boolean term of ψ and by $\mathrm{P}^{+}(\psi)$ ($\mathrm{P}^{-}(\psi)$) the set of predicate symbols that occur under an even (odd) number of negations in ψ. The following proposition allows to restrict the form of ϕ even further, without losing generality:

Proposition 1. *Given* MIL *formulae ϕ_1 and ϕ_2, for any positivation operator $(.)^{\oplus}$, the following hold:*

1. $(\phi_1 \vee \phi_2)^{\oplus} \equiv^{\mu} {\phi_1}^{\oplus} \vee {\phi_2}^{\oplus}$,
2. $(\phi_1 \wedge \phi_2)^{\oplus} \equiv^{\mu} {\phi_1}^{\oplus} \wedge {\phi_2}^{\oplus}$, *provided that* $\mathrm{P}(\phi_1) \cap \mathrm{P}(\phi_2) = \emptyset$.

From now on, we assume that ϕ is a conjunction of cardinality constraints that cannot be split as $\phi = \phi_1 \wedge \phi_2$, such that $\mathrm{P}(\phi_1) \cap \mathrm{P}(\phi_2) = \emptyset$.

Let us consider a cardinality constraint $|t| \geq \ell$ that occurs in ϕ. Given a set $\mathcal{P}$ of predicate symbols, for a set of predicates $S \subseteq \mathcal{P}$, the *complete* boolean minterm corresponding to S with respect to $\mathcal{P}$ is $t_S^{\mathcal{P}} \stackrel{\text{def}}{=} \bigwedge_{p \in S} p \wedge \bigwedge_{p \in \mathcal{P} \setminus S} \neg p$. Moreover, let $\mathcal{S}_t \stackrel{\text{def}}{=} \{S \subseteq \mathrm{P}(\phi) \mid t_S \rightarrow t\}$ be the set of sets S of predicate symbols for which the complete minterm t_S implies t. Finally, each cardinality constraint $|t| \geq \ell$ is replaced by the equivalent disjunction[7], in which each boolean term is complete with respect to $\mathrm{P}(\phi)$:

$$|t| \geq \ell \equiv \bigvee \Big\{ \bigwedge_{S \in \mathcal{S}_t} \left|t_S^{\mathrm{P}(\phi)}\right| \geq \ell_S \mid \text{ for some constants } \{\ell_S \in \mathbb{N}\}_{S \in \mathcal{S}_t} \text{ such that } \sum_{S \in \mathcal{S}_t} \ell_S = \ell \Big\}$$

Note that because any two complete minterms t_S and t_T, for $S \neq T$, are incompatible, then necessarily $|t_S \vee t_T| = |t_S| + |t_T|$. Thus $|t_S \vee t_T| \geq \ell$ if and only if there exist $\ell_1, \ell_2 \in \mathbb{N}$ such that $\ell_1 + \ell_2 = \ell$ and $|t_S| \geq \ell_1$, $|t_T| \geq \ell_2$, respectively.

Notice that, restricting the sets of predicates in $\mathcal{S}_t$ to subsets of $\mathrm{P}(\phi)$, instead of the entire set of predicates, allows to apply Proposition 1 and reduce the number of complete minterm to be considered. That is, whenever possible, we write each minterm $\bigwedge_{i \in L} |t_i| \geq \ell_i \wedge \bigwedge_{j \in U} |t_j| \leq u_j$ in the DNF of ϕ as $\psi_1 \wedge \ldots \wedge \psi_k$, such that $\mathrm{P}(\psi_i) \cap \mathrm{P}(\psi_j) = \emptyset$ for all $1 \leq i < j \leq k$. In practice, this optimisation turns out to be quite effective, as shown by the small execution times of our test cases, reported in §5.

The second step is building, for each conjunction $C = \bigwedge \{\ell_S \leq \left|t_S^{\mathrm{P}(\phi)}\right| \wedge \left|t_S^{\mathrm{P}(\phi)}\right| \leq u_S \mid S \subseteq \mathrm{P}(\phi)\}$[8], as above, a positive formula $C^{\oplus}$, that preserves its set of minimal models

[7] The constraints $|t| \leq u$ are dealt with as $\neg(|t| \geq u + 1)$.

[8] Missing lower bounds ℓ_S are replaced with 0 and missing upper bounds u_S with ∞.

$[\![C]\!]^\mu$. The generalization to arbitrary boolean combinations of cardinality constraints is a direct consequence of Proposition 1. Let $\mathcal{L}^+(\phi)$ (resp. $\mathcal{L}^-(\phi)$) be the set of positive boolean combinations of predicate symbols $p \in \mathrm{P}^+(\phi)$ (resp. $\neg p$, where $p \in \mathrm{P}^-(\phi)$). Further, for a complete minterm $t_S^{\mathcal{P}}$, we write $t_S^{\mathcal{P}+}$ ($t_S^{\mathcal{P}-}$) for the conjunction of the positive (negative) literals in $t_S^{\mathcal{P}}$. Then, we define:

$$C^\oplus \overset{\mathsf{def}}{=} \bigwedge \left\{|\tau| \geq \sum_{t_S^{\mathrm{P}(\phi)+} \rightarrow \tau} \ell_S \mid \tau \in \mathcal{L}^+(\phi)\right\} \wedge \bigwedge \left\{|\tau| \leq \sum_{t_S^{\mathrm{P}(\phi)-} \rightarrow \tau} u_S \mid \tau \in \mathcal{L}^-(\phi)\right\}$$

It is not hard to see that $C^\oplus$ is a positive MIL formula, because:

– for each $\tau \in \mathcal{L}^+(\phi)$, we have $|\tau| \geq k \equiv \exists i_1 \ldots \exists i_k \,.\, \mathrm{distinct}(i_1, \ldots, i_k) \wedge \bigwedge_{j=1}^k \tau(j)$ and
– for each $\tau \in \mathcal{L}^-(\phi)$, we have $|\tau| \leq k \equiv \forall i_1 \ldots \forall i_{k+1} \,.\, \mathrm{distinct}(i_1, \ldots, i_{k+1}) \rightarrow \bigvee_{j=1}^{k+1} \neg\tau(i_j)$.

The following lemma proves that the above definition meets the second requirement of positivation operators, concerning the preservation of minimal models.

Lemma 6. *Given $\mathcal{P}$ a finite set of monadic predicate symbols, $\{\ell_S \in \mathbb{N}\}_{S \subseteq \mathcal{P}}$ and $\{u_S \in \mathbb{N} \cup \{\infty\}\}_{S \subseteq \mathcal{P}}$ sets of constants, for any conjunction $C = \bigwedge\{\ell_S \leq \left|t_S^{\mathcal{P}}\right| \wedge \left|t_S^{\mathcal{P}}\right| \leq u_S \mid S \subseteq \mathcal{P}\}$, we have $C \equiv^\mu C^\oplus$.*

Example 4. (contd. from Example 3)

Consider the first minterm of the DNF of the cardinality constraint obtained by quantifier elimination in Example 3, from the system in Figure 1B. The result of positivation for this minterm is given below:

$$(\neg r \wedge \neg s \wedge |w \wedge \neg u| \leq 0 \wedge |u \wedge \neg w| \leq 0 \wedge 1 \leq |w|)^\oplus = 1 \leq |u \wedge w|$$

Intuitively, the negative literals $\neg r$ and $\neg s$ may safely disapear, because no minimal model will assign r or s to true. Further, the constraints $|w \wedge \neg u| \leq 0$ and $|u \wedge \neg w| \leq 0$ are equivalent to the fact that, in any structure $\mathcal{I} = (\mathfrak{U}, \nu, \iota)$, we must have $\iota(u) = \iota(w)$. Finally, because $|w| \geq 1$, then necessarily $|u \wedge w| \geq 1$.

Similarly, the result of positivation applied to the second conjunct of the DNF cardinality constraint corresponding to the system in Figure 3 is given below:

$$(2 \leq |w| \wedge |w \wedge \neg u| \leq 1 \wedge |u \wedge \neg w| \leq 0)^\oplus = 2 \leq |w| \wedge 1 \leq |u \wedge w|$$

Here, the number of elements in w is at least 2 and, in any structure $\mathcal{I} = (\mathfrak{U}, \nu, \iota)$, we must have $\iota(u) \subseteq \iota(w)$ and at most one element in $\iota(w) \setminus \iota(u)$. Consequently, the intersection of the sets $\iota(u)$ and $\iota(w)$ must contain at least one element, i.e. $|u \wedge w| \geq 1$.

4 Proving Deadlock Freedom of Parametric Systems

We have gathered all the ingredients necessary for checking deadlock freedom of parametric systems, using our method based on trap invariant generation (Figure 4). In particular, we derive a trap constraint $\Theta(\Gamma)$ directly from the interaction formula Γ, both of which are written in MIL. Second, we compute a positive formula that preserves the set of minimal models of $\Theta(\Gamma) \wedge \mathit{Init}(\mathcal{S})$, by first converting the MIL formula into a quantifier-free cardinality constraint, using quantifier elimination, and deriving a positive MIL formula from the latter.

The conjunction between the dual of this positive formula and the formula $\Delta(\Gamma)$ that defines the deadlock states is then checked for satisfiability. Formally, given a parametric system $\mathcal{S}$, with an interaction formula Γ written in the form (1), the MIL formula

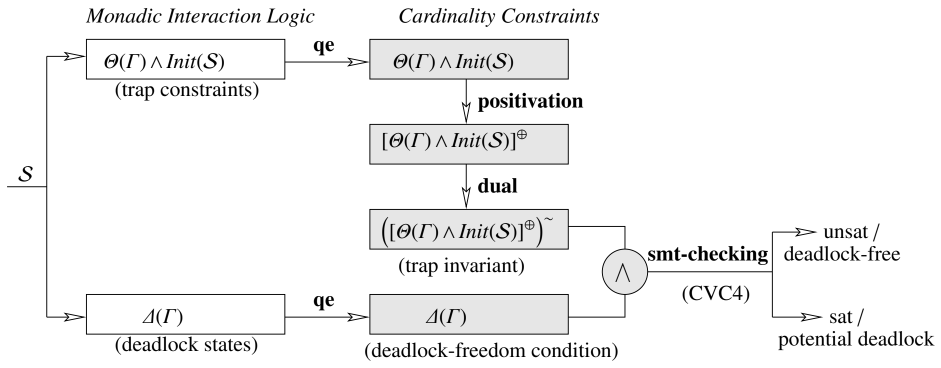

Monadic Interaction Logic
Cardinality Constraints
$\Theta(\Gamma) \wedge Init(\mathcal{S})$
(trap constraints)
qe
$\Theta(\Gamma) \wedge Init(\mathcal{S})$
positivation
$[\Theta(\Gamma) \wedge Init(\mathcal{S})]^{\oplus}$
dual
$\left([\Theta(\Gamma) \wedge Init(\mathcal{S})]^{\oplus}\right)^{\sim}$
(trap invariant)
$\mathcal{S}$
$\wedge$
smt-checking
(CVC4)
unsat / deadlock-free
sat / potential deadlock
$\Delta(\Gamma)$
(deadlock states)
qe
$\Delta(\Gamma)$
(deadlock-freedom condition)

Fig. 4: Verification of Parametric Component-based Systems

characterizing the deadlock states of the system is the following:

$$\Delta(\Gamma) \overset{\mathsf{def}}{=} \forall i_1 \ldots \forall i_\ell \,.\, \varphi \rightarrow \left[\, \bigvee_{j=1}^{\ell} \neg^{\bullet} p_j(i_j) \vee \bigvee_{j=\ell+1}^{\ell+m} \exists i_j \,.\, \psi_j \wedge \neg^{\bullet} p_j(i_j) \right]$$

We state a sufficient verification condition for deadlock freedom in the parametric case:

Corollary 2. *A parametric system $\mathcal{S} = \langle C^1, \ldots, C^n, \mathsf{M}, \Gamma \rangle$ is deadlock-free if*

$$\left((\Theta(\Gamma) \wedge Init(\mathcal{S}))^{\oplus} \right)^{\sim} \wedge \Delta(\Gamma) \rightarrow \bot$$

The satisfiability check is carried out using the conversion to cardinality constraints via quantifier elimination §3.1 and an effective set theory solver for cardinality constraints, implemented in the CVC4 SMT solver [5].

5 Experimental Results

To assess our method for proving deadlock freedom of parametric component-based system, we ran a number of experiments on systems with a small numbers of rather simple component types, but with nontrivial interaction patterns, given by MIL formulae. The `task-sem` i/n examples, $i = 1, 2, 3$, are generalizations of the parametric *Task-Semaphore* example depicted in Figure 1b, in which n *Task*s synchronize using n *Semaphore*s, such that i *Task*s interact with a single *Semaphore* at once, in a multiparty rendez-vous. In a similar vein, the `broadcast` i/n examples, $i = 2, 3$ are generalizations of the system in Figure 3, in which i out of n *Worker*s engage in rendez-vous on the b port, whereas all the other stay idle — here idling is modeled as a broadcast on the a ports. Finally, in the `sync` i/n examples, $i = 1, 2, 3$, we consider systems composed of n *Worker*s (Figure 1b) such that either i out of n instances simultaneously interact on the b ports, or all interact on the f ports. Notice that, for $i = 2, 3$, these systems have a deadlock if and only if $n \neq 0 \mod i$. This is because, if $n = m \mod i$, for some $0 < m < i$, there will be be m instances that cannot synchronize on their b port, in order to move from w to u, in order to engage in the f broadcast.

All experiments were carried out on a Intel(R) Xeon(R) CPU @ 2.00GHz virtual machine with 4GB of RAM. Table 1 shows separately the times needed to generate

example	interaction formula	t-gen	t-smt	result
`task-sem` $1/n$	$\exists i \exists j_1.\, a(i) \wedge b(j_1) \ \bigvee\ \exists i \exists j_1.\, e(i) \wedge f(j_1)$	22 ms	20 ms	unsat
`task-sem` $2/n$	$\exists i \exists j_1 \exists j_2.\, j_1 \neq j_2 \wedge a(i) \wedge b(j_1) \wedge b(j_2) \ \bigvee$ $\exists i \exists j_1 \exists j_2.\, j_1 \neq j_2 \wedge e(i) \wedge f(j_1) \wedge f(j_2)$	34 ms	40 ms	unsat
`task-sem` $3/n$	$\exists i \exists j_1 \exists j_2 \exists j_3.\, \text{distinct}(j_1, j_2, j_3) \wedge a(i) \wedge b(j_1) \wedge b(j_2) \wedge b(j_3) \ \bigvee$ $\exists i \exists j_1 \exists j_2 \exists j_3.\, \text{distinct}(j_1, j_2, j_3) \wedge e(i) \wedge f(j_1) \wedge f(j_2) \wedge f(j_3)$	73 ms	40 ms	unsat
`broadcast` $2/n$	$\exists i_1 \exists i_2. i_1 \neq i_2 \wedge b(i_1) \wedge b(i_2) \wedge$ $\forall j.\, j \neq i_1 \wedge j \neq i_2 \rightarrow a(j) \ \bigvee \exists i. f(i)$	14 ms	20 ms	unsat
`broadcast` $3/n$	$\exists i_1 \exists i_2 \exists i_3. \text{distinct}(i_1, i_2, i_3) \wedge b(i_1) \wedge b(i_2) \wedge b(i_3) \wedge$ $\forall j.\, j \neq i_1 \wedge j \neq i_2 \wedge j \neq i_3 \rightarrow a(j) \ \bigvee\ \exists i. f(i)$	409 ms	20 ms	unsat
`sync` $1/n$	$\exists i. b(i) \ \bigvee\ \forall i. f(i)$	5 ms	20 ms	unsat
`sync` $2/n$	$\exists i_1 \exists i_2.\, i_1 \neq i_2 \wedge b(i_1) \wedge b(i_2) \ \bigvee\ \forall i. f(i)$	7 ms	50 ms	sat
`sync` $3/n$	$\exists i_1 \exists i_2 \exists i_3. \text{distinct}(i_1, i_2, i_3) \wedge b(i_1) \wedge b(i_2) \wedge b(i_3) \ \bigvee\ \forall i. f(i)$	11 ms	40 ms	sat

Table 1: Benchmarks

the proof obligations (trap invariants and deadlock states) from the interaction formulae and the times needed by CVC4 1.7 to show unsatisfiabilty or come up with a model. All systems considered, for which deadlock freedom could not be shown using our method, have a real deadlock scenario that manifests only under certain modulo constraints on the number n of instances. These constraints cannot be captured by MIL formulae, or, equivalently by cardinality constraints, and would require cardinality constraints of the form $|t| = n \mod m$, for some constants $n, m \in \mathbb{N}$.

6 Conclusions

This work is part of a lasting research program on BIP linking two work directions: (1) recent work on modeling architectures using interaction logics, and (2) older work on verification by using invariants. Its rationale is to overcome as much as possible complexity and undecidability issues by proposing methods which are adequate for the verification of essential system properties.

The presented results are applicable to a large class of architectures characterized by the MIL. A key technical result is the translation of MIL formulas into cardinality constraints. This allows on the one hand the computation of the MIL formula characterizing the minimal trap invariant. On the other hand, it provides a decision procedure for MIL, that leverages from recent advances in SMT, implemented in the CVC4 solver [5].

References

1. Abdulla, P.A.: Well (and better) quasi-ordered transition systems. The Bulletin of Symbolic Logic 16(4), 457–515 (2010)
2. Abdulla, P.A., Delzanno, G., Henda, N.B., Rezine, A.: Regular model checking without transducers (on efficient verification of parameterized systems). In: Grumberg, O., Huth, M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 721–736 (2007)

3. Alberti, F., Ghilardi, S., Sharygina, N.: A framework for the verification of parameterized infinite-state systems*. CEUR Workshop Proceedings 1195, 302–308 (01 2014)
4. Bansal, K., Reynolds, A., Barrett, C.W., Tinelli, C.: A new decision procedure for finite sets and cardinality constraints in SMT. In: IJCAR'16 Proceedings. pp. 82–98 (2016)
5. Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: CAV'11 Proceedings. LNCS, vol. 6806, pp. 171–177 (2011)
6. Basu, A., Bensalem, S., Bozga, M., Combaz, J., Jaber, M., Nguyen, T., Sifakis, J.: Rigorous component-based system design using the BIP framework. IEEE Software 28(3), 41–48 (2011)
7. Baukus, K., Bensalem, S., Lakhnech, Y., Stahl, K.: Abstracting ws1s systems to verify parameterized networks. In: Graf, S., Schwartzbach, M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 188–203 (2000)
8. Bensalem, S., Bozga, M., Nguyen, T., Sifakis, J.: D-finder: A tool for compositional deadlock detection and verification. In: CAV'09 Proceedings. LNCS, vol. 5643, pp. 614–619 (2009)
9. Bloem, R., Jacobs, S., Khalimov, A., Konnov, I., Rubin, S., Veith, H., Widder, J.: Decidability of Parameterized Verification. Synthesis Lectures on Distributed Computing Theory, Morgan & Claypool Publishers (2015)
10. Bouajjani, A., Habermehl, P., Vojnar, T.: Abstract regular model checking. In: Alur, R., Peled, D.A. (eds.) Computer Aided Verification. pp. 372–386 (2004)
11. Bozga, M., Iosif, R., Sifakis, J.: Checking Deadlock-Freedom of Parametric Component-Based Systems. Tech. Rep. ArXiv 1805.10073, https://arxiv.org/abs/1805.10073 (2018)
12. Chen, Y., Hong, C., Lin, A.W., Rümmer, P.: Learning to prove safety over parameterised concurrent systems. In: 2017 Formal Methods in Computer Aided Design, FMCAD 2017, Vienna, Austria, October 2-6, 2017. pp. 76–83 (2017)
13. Conchon, S., Goel, A., Krstić, S., Mebsout, A., Zaïdi, F.: Cubicle: A parallel smt-based model checker for parameterized systems. In: Madhusudan, P., Seshia, S.A. (eds.) Computer Aided Verification. pp. 718–724 (2012)
14. Emerson, E.A., Namjoshi, K.S.: Reasoning about rings. In: POPL'95 Proceedings. pp. 85–94 (1995)
15. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM 39(3), 675–735 (1992)
16. Kesten, Y., Maler, O., Marcus, M., Pnueli, A., Shahar, E.: Symbolic model checking with rich assertional languages. Theoretical Computer Science 256(1), 93 – 112 (2001)
17. Kuncak, V., Nguyen, H.H., Rinard, M.C.: Deciding boolean algebra with Presburger arithmetic. J. Autom. Reasoning 36(3), 213–239 (2006)
18. Lowenheim, L.: Über Möglichkeiten im Relativkalkül. Math. Ann 470, 76–447 (1915)
19. Suzuki, I.: Proving properties of a ring of finite-state machines. Inf. Process. Lett. 28(4), 213–214 (1988)