Parameterized Systems in BIP: Design and Model Checking*

Igor Konnov[2], Tomer Kotek[2], Qiang Wang[1], Helmut Veith[2], Simon Bliudze[1], and Joseph Sifakis[1]

1 **École polytechnique fédérale de Lausanne, Switzerland**
`firstname.lastname@epfl.ch`
2 **TU Wien (Vienna University of Technology), Austria**
`lastname@forsyte.at`

Abstract

BIP is a component-based framework for system design built on three pillars: behavior, interaction, and priority. In this paper, we introduce first-order interaction logic (FOIL) that extends BIP without priorities to systems parameterized in the number of components. We show that FOIL captures classical parameterized architectures such as token-passing rings, cliques of identical components communicating with rendezvous or broadcast, and client-server systems.

Although the BIP framework includes efficient verification tools for statically-defined systems, none are available for parameterized systems with an unbounded number of components. On the other hand, the parameterized model checking literature contains a wealth of techniques for systems of classical architectures. However, application of these results requires a deep understanding of parameterized model checking techniques and their underlying mathematical models. To overcome these difficulties, we introduce a framework that automatically identifies parameterized model checking techniques applicable to a BIP design. To our knowledge, this is the first framework that allows one to apply prominent parameterized model checking results in a systematic way.

1998 ACM Subject Classification [Software Engineering] D.2.2: Design Tools and Techniques, D.2.4 Software/Program Verification

Keywords and phrases Rigorous system design, BIP, verification, parameterized model checking

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2016.30

1 Introduction

Design, manufacture and verification of large scale complex hardware/software systems (e.g., cyber-physical systems) remains a grand challenge in system design automation [25]. To address this challenge, the rigorous system design methodology [24] and the behaviour-interaction-priority (BIP) framework [4] have been recently proposed. BIP comes with a formal framework and a toolchain. The BIP framework has well-defined semantics for modeling system behavior and architectures. The BIP toolchain supports verification of high-level system designs and automatic system synthesis of low-level implementations from high-level system designs.

The existing BIP tools focus on design and verification of systems with a fixed number of communicating components [5, 22]. However, many distributed systems are designed with parameterization

* We dedicate this article to the memory of Helmut Veith, who passed away tragically while this manuscript was being prepared. His curiosity and energy ignited our joint effort in this research.
This work was supported by the Austrian National Research Network S11403-N23 (RiSE), the Vienna Science and Technology Fund (WWTF) through the grant APALACHE (ICT15-103), and, partially, by the Swiss National Science Foundation through the National Research Programme "Energy Turnaround" (NRP 70) grant 153997.

in mind. For instance, the number of components in the system is not typically fixed, but varies depending on the system setup. In this case, one talks about parameterized verification, where the number of components is a parameter.

Model checking is a pragmatic approach to verification that has found many applications in industry, e.g., see [19]. Many efforts were invested into extension of model checking to the parameterized case, which led to numerous parameterized model checking techniques (see [9] for a recent survey). Unfortunately, often parameterized model checking techniques come with their own mathematical models, which makes their practical application difficult. To perform parameterized model checking, the user has to thoroughly understand the research literature. Typically, the user needs to first manually inspect the parameterized models and match them with the mathematical formalisms from the relevant parameterized verification techniques. Using the match, the user would then apply the decidability results (if any) for the parameterized models, e.g., by computing a cutoff or translating the parameterized model into the language of a particular tool for the specific architecture. Thus, there is a gap between the mathematical formalisms and algorithms from the parameterized verification research and the practice of parameterized verification, which is usually done by engineers who are not familiar with the details of the research literature. In this paper, we aim at closing this gap by introducing a framework for design and verification of parameterized systems in BIP. With this framework, we make the following contributions:

1. We extend propositional interaction logic to the parameterized case with arithmetics, which we call *first-order interaction logic* (FOIL). We build on the ideas from configuration logic [21] and dynamic BIP [10]. FOIL is powerful enough to express architectures found in distributed systems, including the classical architectures: token-passing rings, rendezvous cliques, broadcast cliques, and rendezvous stars. We also identify a decidable fragment of FOIL which has important applications in practice. This contribution is covered by Section 3.
2. We provide a framework for integration of mathematical models from the parameterized model checking literature in an automated way: given a parameterized BIP design, our framework detects parameterized model checking techniques applicable to this design. This automation is achieved by the use of SMT solvers and standard (non-parameterized) model checkers. This contribution is covered by Sections 4 and 5.
3. We provide a preliminary prototype implementation of the proposed framework. Our prototype tool takes a parameterized BIP design as its input and detects whether one of the following classical results applies to this BIP design: the cut-off results for token-passing rings by Emerson & Namjoshi [16], the VASS-based algorithms by German & Sistla [18], and the undecidability and decidability results for broadcast systems by Abdulla et al. [1] and Esparza et al. [17]. More importantly, our framework is not specifically tailored to the mentioned techniques. This contribution is covered by Sections 5 and 6.

We remark that our framework builds on the notions of BIP, which allows us to express complex notions in terminology understood by engineers. Moreover, our framework allows an expert in parameterized model checking to capture seminal mathematical models found in the verification literature, e.g., [18, 17, 16, 13].

This paper is structured as follows. In Section 2, we briefly recall the BIP modeling framework. In Section 3, we introduce our parameterized extension. In Sections 4 and 5, we present our verification framework and the automatic system architecture identification technique. In Section 6, we present the preliminary experiments. Section 7 closes with related work, conclusions, and future work.

2 BIP without priorities

In this section, we review the notions of BIP [4] with the following restrictions: (i) states of the components do not have specific internal structure; (ii) we do not consider interaction priorities. While we believe that our approach can be extended to priorities, we leave this for future work.

As usual, a labeled transition system is a tuple (S, s_0, A, R) with a set of locations S, an initial location $s_0 \in S$, a non-empty set of actions A, and a transition relation $R \subseteq S \times A \times S$.

▶ **Definition 2.1** (Component type)**.** A component type is a transition system $\mathbb{B} = \langle \mathbb{Q}, \ell^0, \mathbb{P}, \mathbb{E} \rangle$ over the finite sets $\mathbb{Q}$ and $\mathbb{P}$. By convention, the set of actions $\mathbb{P}$ is called the set of ports.

Ports form the interface of a component type. We assume that, for each location, no two outgoing transitions from this location are labeled with the same port. We also assume that the ports of each component type, as well as the locations, are disjoint.

Let $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle$ be a tuple of component types, where each $\mathbb{B}_i$ is $\langle \mathbb{Q}_i, \ell_i^0, \mathbb{P}_i, \mathbb{E}_i \rangle$ for $i \in [0, k)$. We introduce an infinite set of components $\{\mathbb{B}_i[j] \mid j \geq 0\}$ for $i \in [0, k)$. A *component* $\mathbb{B}_i[j] = \langle \mathbb{Q}_i[j], \ell_i^0[j], \mathbb{P}_i[j], \mathbb{E}_i[j] \rangle$ is obtained from the component type $\mathbb{B}_i$ by renaming the set of ports. Thus, as transition systems, $\mathbb{B}_i[j]$ and $\mathbb{B}_i$ are isomorphic. We postulate $\mathbb{P}_i[j] \cap \mathbb{P}_i[j'] = \emptyset$, for $j \neq j'$.

A BIP model is a composition of finitely many components instantiated from the component types $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle$. To denote the number of components of each type, we introduce a size vector $\bar{N} = \langle N_0, \ldots, N_{k-1} \rangle$: there are N_i components of component type $\mathbb{B}_i$, for $i \in [0, k)$.

Coordination of components is specified with interactions. Intuitively, an interaction defines a multi-party synchronization of component transitions. A BIP interaction is a finite set of ports, which defines a possible synchronization among components.

▶ **Definition 2.2** (Interaction)**.** Given a tuple of component types $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle$ and a size vector $\bar{N} = \langle N_0, \ldots, N_{k-1} \rangle$, an interaction $\gamma \subseteq \{p \in \mathbb{P}_i[j] \mid i \in [0, k), j \in [0, N_i)\}$ is a set of ports such that $|\gamma \cap \mathbb{P}_i[j]| \leq 1$ for all $i \in [0, k)$ and $j \in [0, N_i)$, i.e., an interaction is a set of ports such that at most one port of each component takes part in an interaction. If $p \in \gamma$, we say that p is *active* in γ.

▶ **Definition 2.3** (BIP Model)**.** Given a tuple of component types $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle$ and a size vector $\bar{N} = \langle N_0, \ldots, N_{k-1} \rangle$, a BIP model $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle^{\bar{N}, \Gamma}$ is a tuple $\langle \mathcal{B}, \Gamma \rangle$, where $\mathcal{B}$ is the set $\{\mathbb{B}_i[j] \mid i \in [0, k), j \in [0, N_i)\}$ and Γ is a set of interactions defined w.r.t. $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle$ and $\bar{N}$.

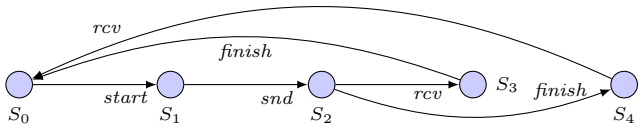
▶ **Definition 2.4** (BIP operational semantics)**.** Given a BIP model $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle^{\bar{N}, \Gamma}$, we define its operational semantics as a transition system $TS(\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle^{\bar{N}, \Gamma}) = \langle S, s_0, \Gamma, R \rangle$, where:

1. The set of *configurations* S is defined as the Cartesian product of the sets of locations of the components $\mathbb{Q}_0^{N_0} \times \cdots \times \mathbb{Q}_{k-1}^{N_{k-1}}$. Given a configuration $s \in S$, we denote by $s(i, j)$ the j^{th} member of the tuple defined by the i^{th} product $\mathbb{Q}_i^{N_i}$ where $j \in [0, N_i)$.
2. The initial configuration $s_0 \in S$ satisfies that $s_0(i, j) = \ell_i^0[j]$ for all $i \in [0, k)$ and $j \in [0, N_i)$.
3. The transition relation R contains a triple (s, γ, s'), if, for each $i \in [0, k)$ and $j \in [0, N_i)$, the j^{th} component of type i
   - either has an active port $p \in \gamma \cap \mathbb{P}_i[j]$ and $\langle s(i, j), p, s'(i, j) \rangle \in \mathbb{E}_i[j]$,
   - or is not participating in the interaction γ, i.e., $\gamma \cap \mathbb{P}_i[j] = \emptyset$ and $s'(i, j) = s(i, j)$.

Intuitively, the local transitions of components fire simultaneously, provided that their ports are included in the interaction; other components do not move.

▶ **Example 2.5** (Milner's scheduler)**.** We follow the formulation by Emerson & Namjoshi [16]. A scheduler is modeled as a token-passing ring. Only the process that owns the token may start running a new task. The component type $\mathbb{B}_0 = \langle \mathbb{Q}_0, \ell_0^0, \mathbb{P}_0, \mathbb{E}_0 \rangle$ is given by the locations $\mathbb{Q}_0 = \{S_0, \ldots, S_4\}$, the

initial location $\ell_0^0 = S_0$, the port types $\mathbb{P}_0 = \{\text{snd}, \text{rcv}, \text{start}, \text{finish}\}$, and the edges $\mathbb{E}_0$ that are shown in the figure below:



A component owns the token when in the location S_0, S_1, or S_3. In S_0, a component initiates its task by interacting on port *start*. The token is then sent to the component's right neighbor on the ring via an interaction on port *snd*. The component then waits until (a) its initiated task has finished, and (b) the component has received the token again. When both (a) and (b) have occurred, the component may initiate a new task. Note that (a) and (b) may occur in either order.

Fix a number $N_0 \in \mathbb{N}$. The following set of interactions represents the ring structure:

$$\Gamma = \{\gamma_{i\to j}, \gamma_{\text{start}(i)}, \gamma_{\text{finish}(i)} \mid 0 \le i < N_0 \text{ and } j \equiv i+1 \bmod n_0\}$$

where $\gamma_{i\to j} = \{(\text{snd}, i), (\text{rcv}, j)\}$ is the interaction passing the token from the i^{th} component to the next component on the ring, while the interactions $\gamma_{\text{start}(i)} = \{(\text{start}, i)\}$ and $\gamma_{\text{finish}(i)} = \{(\text{finish}, i)\}$ allow the i^{th} component to take the internal transitions labeled 'start' and 'finish' respectively. The BIP model of the Milner scheduler of size N_0 is $\langle \mathcal{B}, \Gamma\rangle$, where $\mathcal{B}$ is the set of components $\{\mathbb{B}_0[j] \mid j \in [0, N_0)\}$.

3 Parameterized BIP without priorities

Since the number of possible interactions in a parameterized system is unbounded, and each interaction itself may involve an unbounded number of actions, the set of all possible interactions is infinite. Hence, we need a symbolic representation of such a set. To this end, we propose *first order interaction logic*—a uniform and formal language for system topologies and coordination mechanisms in parameterized systems. Using this logic, we introduce a parameterized extension of BIP, and show that this extension naturally captures standard examples.

3.1 FOIL: First order interaction logic

In this section, we fix a tuple of component types $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1}\rangle$. For each port $p \in \mathbb{P}_i$ of an i^{th} component type, we introduce a unary *port predicate* with the same name p. Furthermore, we introduce a tuple of constants $\bar{n} = \langle n_0, \ldots, n_{k-1}\rangle$, which represents the number of components of each type. We also assume the standard vocabulary of Presburger arithmetic, that is, $\langle 0, 1, \le, +\rangle$.

FOIL syntax Assume an infinite set of index variables $\mathcal{I}$. We say that ψ is a first order interaction logic formula, if it is constructed according to the following grammar:

$$\psi ::= p(i) \mid \neg\psi \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \exists i :: type_j : \phi.\ \psi \mid \forall i :: type_j : \phi.\ \psi\,,$$

where $p \in \mathbb{P}_0 \cup \cdots \cup \mathbb{P}_{k-1}$, $i \in \mathcal{I}$, and ϕ is a formula in Presburger arithmetic over index variables and the vocabulary $\langle 0, 1, \le, +, \bar{n}\rangle$.

Informally, the syntax $\mathbf{Q}\ i :: type_j : \phi.\ \psi$, where $\mathbf{Q} \in \{\exists, \forall\}$, restricts the index variable i to be associated with the component type $\mathbb{B}_j$. Notice, however, that this syntax does not enforce type correctness of ports. For instance, one can write a formula $\exists i :: type_j : p(i)$ with some $p \notin \mathbb{P}_j$. While this formula is syntactically correct, it is not in line with Definition 2.2 of interaction given in Section 2. To this end, we say that a FOIL formula is *natural*, if for each of its subformulae $\mathbf{Q}\ i :: type_j : \phi.\ \psi(i)$, for $\mathbf{Q} \in \{\exists, \forall\}$, and every atomic formula $p(i)$ of ψ, it holds that $p \in \mathbb{P}_j$. From here on, we assume FOIL formulae to be natural. We write $\exists i :: type_j.\ \psi$ as a shorthand for $\exists i :: type_j : true.\ \psi$.

FOIL semantics We give the semantics of a FOIL formula by means of structures. A *first-order interaction logic structure* (FOIL structure) is a pair $\xi = (\mathbb{N}, \alpha_\xi)$: the set of natural numbers $\mathbb{N}$ is the domain of ξ, while α_ξ is the interpretation of all the predicates and of the constants $\bar{n}$. The symbols 0, 1, $\leq$, and + have the natural interpretations over $\mathbb{N}$.

A valuation σ is a function $\sigma : \mathcal{I} \to \mathbb{N}$. We denote by $\sigma[x \mapsto j]$ the valuation obtained from σ by mapping the index variable x to the value j. Assignments are used to give values to free variables in formulae. For a FOIL structure ξ and a valuation σ, the semantics of FOIL is formally given as follows (the semantics of Boolean operators and universal quantifiers is defined in the standard way):

$$\begin{array}{lll} \xi, \sigma \models_{\text{FOIL}} p(i) & \textit{iff} & \alpha_\xi(p) \text{ is true on } \sigma(i) \\ \xi, \sigma \models_{\text{FOIL}} \exists i :: type_j : \phi.\ \psi & \textit{iff} & \text{there is } l \in [0, \alpha_\xi(n_j)) \text{ such that} \\ & & \xi, \sigma[i \mapsto l] \models_{\text{FO}} \phi \text{ and } \xi, \sigma[i \mapsto l] \models_{\text{FOIL}} \psi \end{array}$$

where $\models_{\text{FO}}$ to denotes the standard 'models' relation of first-order logic.

Finally, for a FOIL formula ψ without free variables and a structure ξ, we write $\xi \models_{\text{FOIL}} \psi$, if $\xi, \sigma_0 \models_{\text{FOIL}} \psi$ for the valuation σ_0 that assigns 0 to every index $i \in \mathcal{I}$.[1]

Decidability It is easy to show that checking validity of a FOIL sentence[2] is undecidable, and that FOIL contains an important decidable fragment:

▶ **Theorem 3.1** (Decidability of FOIL)**.** *The following results about FOIL hold:*

(i) Validity of FOIL sentences is undecidable.
(ii) Validity of FOIL sentences in which all additions are of the form $i + 1$ is decidable.

Proof. (i) FOIL contains Presburger arithmetic with unary predicates, which is known to be as strong as Peano arithmetic [20]. Hence, satisfiability and validity of FOIL formulae are undecidable.

(ii) The formula $j = i + 1$ is definable in FOIL by $i \leq j \wedge j \neq i \wedge \psi_{consecutive}(i, j)$, where $\psi_{consecutive}(i, j) = \forall \ell :: type_t.\ (j \leq \ell \wedge \ell \leq i) \to (\ell = i \vee \ell = j)$, where t is the type of i and j. Hence, we can rewrite any FOIL sentence ψ in which all additions are of the form $i + 1$ as an equi-satisfiable first-order logic sentence ψ' without using addition (+). The sentence ψ' belongs to S1S, the monadic second order theory of $(\mathbb{N}, 0, 1, \leq)$, which is decidable, see [27]. ◀

In the following, we restrict addition to the form $i + 1$, and thus stay in the decidable fragment.

3.2 Interactions as FOIL structures

In contrast to Definition 2.2 of a standard interaction, which is represented explicitly as a finite set of ports, we use first order interaction logic formulae to define all the possible interactions in parameterized systems. Our key insight is that each structure of a formula uniquely defines at most one interaction, and the set of all possible interactions is the union of the interactions derived from the structures that satisfy the formula.

Intuitively, if $p(j)$ evaluates to true in a structure ξ, then the j^{th} instance of the respective component type—uniquely identified by the port p—takes part in the interaction identified with ξ. Thus, we can reconstruct a standard BIP interaction from a FOIL structure by taking the set of ports, whose indices are evaluated to true by the unary predicates. Formally, given a FOIL structure $\xi = (\mathbb{N}, \alpha_\xi)$, we define the set $\gamma_\xi = \{(p, j) \mid i \in [0, k),\ p \in \mathbb{P}_i,\ j \in [0, \alpha_\xi(n_j)),\ \alpha_\xi(p)(j) = true\}$. In the following, the notation (p, j) denotes the port p of the j^{th} component of the type $\mathbb{B}_i$ with $p \in \mathbb{P}_i$.

[1] Since ψ has no free variables, our choice of σ_0 is arbitrary: for all σ we have $\xi, \sigma \models_{\text{FOIL}} \psi$ if and only if $\xi, \sigma_0 \models_{\text{FOIL}} \psi$.
[2] A FOIL formula with no free variables is called a *sentence*. A sentence is *valid* if it is satisfied by all structures.

Notice that γ_ξ does not have to be an interaction in the sense of Definition 2.2. Indeed, one can define ξ whose set γ_ξ includes two ports of the same component. We say that ξ *induces an interaction*, if γ_ξ is an interaction in the sense of Definition 2.2.

▶ **Definition 3.2** (Parameterized BIP Model)**.** A parameterized BIP model is a tuple $\langle \mathbb{B}, \bar{n}, \psi, \epsilon \rangle$, where $\mathbb{B} = \langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle$ is a tuple of component types, ψ is a sentence in FOIL over port predicates and a tuple $\bar{n} = \langle n_0, \ldots, n_{k-1} \rangle$ of size parameters, and ϵ is a linear constraint over $\bar{n}$.

The tuple $\bar{n}$ consists of the size parameters for all component types, and the constraint ϵ restricts these parameters. For example, the formula $(n_0 = 1) \wedge (n_1 \geq 10)$ requires every instance of a parameterized BIP model to have only one component of the first type and at least ten components of the second type. The FOIL sentence ψ restricts both the system topology and the communication mechanisms, see Example 3.4.

▶ **Definition 3.3** (PBIP Instance)**.** Given a parameterized BIP model $\langle \mathbb{B}, \bar{n}, \psi, \epsilon \rangle$ and a size vector $\bar{N}$, a *PBIP instance* is a BIP model $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle^{\bar{N}, \Gamma} = \langle \mathcal{B}, \Gamma \rangle$, where $\mathcal{B}$ and Γ are defined as follows:

1. the numbers $\bar{N}$ satisfy the size constraint ϵ,
2. the set of components $\mathcal{B}$ is $\{\mathbb{B}_i[j] \mid i \in [0, k) \text{ and } j \in [0, N_j)\}$, and
3. the set of interactions Γ consists of all interactions γ_ξ induced by a FOIL structure ξ such that the size parameters $\bar{n}$ are interpreted in ξ as $\bar{N}$, and ξ satisfies ψ, i.e. $\alpha_\xi(\bar{n}) = \bar{N}$ and $\xi \models_{\text{FOIL}} \psi$.

In the rest of this section, we give three examples that show expressiveness of parameterized BIP.

▶ **Example 3.4** (Milner's scheduler revisited)**.** The parameterized BIP model of Milner's scheduler is $\langle \langle \mathbb{B}_0 \rangle, \langle n_0 \rangle, \psi, \mathit{true} \rangle$, where $\mathbb{B}_0$ is from Example 2.5 and $\psi = \psi_{\mathit{token}} \vee \psi_{\mathit{internal}}$ defined as follows. The formula ψ_{token} defines the token-passing interactions and the formula ψ_{internal} defines the internal interactions of starting or finishing a task:

$$\begin{array}{lcl} \psi_{\mathit{token}} & = & \exists i, j :: \mathit{type}_0 : j = (i+1) \bmod n_0.\ \mathit{snd}(i) \wedge \mathit{rcv}(j) \wedge \psi_{\mathit{only}}(i,j) \\ \psi_{\mathit{only}}(i,j) & = & \forall \ell :: \mathit{type}_0 : \ell \neq i \wedge \ell \neq j.\ \neg \mathit{snd}(\ell) \wedge \neg \mathit{rcv}(\ell) \wedge \neg \mathit{start}(i) \wedge \neg \mathit{finish}(i) \\ \psi_{\mathit{internal}} & = & \exists i :: \mathit{type}_0.\ \psi_{\mathit{only}}(i,i) \wedge (\mathit{start}(i) \vee \mathit{finish}(i)) \end{array}$$

The formula ψ_{token} does not have free variables and holds for a structure ξ, if the induced interaction γ_ξ is a send-receive interaction along some edge $i \to j$ of the ring, where $j = (i+1) \bmod n_0$. In fact, $j = (i+1) \bmod n_0$ is just a shorthand for the formula: $(i+1 < n_0 \wedge j = i+1) \vee (i+1 = n_0 \wedge j = 0)$. The formula $\psi_{\mathit{only}}(i,j)$ excludes any component other than i and j from participating in the interaction. (If $i = j$ then all components other than i are excluded.) The formula ψ_{internal} enables the transitions labeled with 'start' and 'finish', in which only one component changes its location.

Observe that the semantics of FOIL forces the quantified variables i, j, ℓ to be in the range from 0 to $N_0 - 1$. Hence, we omit explicit range constraints. For instance, ψ_{token} is equivalent to the formula:

$$\exists i, j :: \mathit{type}_0 : 0 \leq i, j < n_0 \wedge (j = (i+1) \bmod n_0).\ \mathit{snd}(i) \wedge \mathit{rcv}(j) \wedge \psi_{\mathit{only}}(i,j)$$

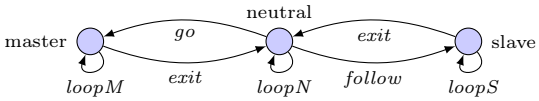
The set of FOIL structures ξ that satisfy ψ induces the same set of interactions Γ as in Example 2.5. While Example 2.5 defines the set Γ explicitly for any fixed value N_0, in the parameterized setting the interactions are defined uniformly by a single FOIL formula ψ, for all values of N_0.

In this example we do not restrict the initial locations so that exactly one process owns the token in the initial configuration. This delicate issue is resolved in Section 5.4.

▶ **Example 3.5** (Broadcast in a star)**.** Let $\langle \langle \mathbb{B}_0, \mathbb{B}_1 \rangle, \langle n_0, n_1 \rangle, \psi, \epsilon \rangle$ be a parameterized BIP model with two component types and the size constraint $\epsilon \equiv (n_0 = 1)$. We also assume that component type $\mathbb{B}_0$ (resp. $\mathbb{B}_1$) has only one port *send* (resp. *receive*), i.e., $\mathbb{P}_0 = \{\mathit{send}\}$ and $\mathbb{P}_1 = \{\mathit{receive}\}$. The

FOIL formula $\psi = \exists i :: type_0.\ send(i)$ specifies broadcast from the component $\mathbb{B}_0[0]$, the center of the star, to the leaves of type $\mathbb{B}_1$. The set of interactions defined by ψ consists of all sets of ports of the form $\{(send, 0)\} \cup \{(receive, d) \mid d \in D)\}$ for all $D \subseteq [0, n_1)$, including the empty set $D = \emptyset$.

▶ **Example 3.6** (Barrier). Consider a barrier synchronization protocol, cf. [9, Example 6.6]. The component type $\mathbb{B}_0$ is as shown below:



The location *neutral* is the initial location. A synchronization episode consists of three stages: (i) First, a single component enters the barrier by moving to *master*. (ii) Then, each of the others components moves to *slave*. (iii) Finally, the master triggers a broadcast and all components leave the barrier by moving to *neutral*. The parameterized BIP model of the barrier synchronization protocol is $\langle\langle \mathbb{B}_0\rangle, \langle n_0\rangle, \psi, true\rangle$, where $\psi = \psi_{go} \vee \psi_{follow} \vee \psi_{exit}$, and the following formulae ψ_{go}, ψ_{follow}, and ψ_{exit} describe the interactions of stages (i), (ii), and (iii) respectively:

$$\begin{aligned}
\psi_{go} &= \exists i :: type_0.\ go(i) \quad \wedge \forall j :: type_0 : i \neq j.\ loopN(j) \\
\psi_{follow} &= \exists i, j :: type_0.\ follow(i) \wedge loopM(j) \wedge \\
&\quad \forall \ell :: type_0 : i \neq \ell.\ loopM(\ell) \vee loopN(\ell) \vee loopS(\ell) \\
\psi_{exit} &= \forall i :: type_0.\ exit(i)
\end{aligned}$$

All three formulae enforce progress by requiring at least one process to change its state.

4 Parameterized model checking

In this section, we review the syntax and semantics of the indexed version of CTL^*, called $ICTL^\star$, which is often used to specify the properties of parameterized systems [9]. Though we use indexed temporal logics to define the standard parameterized model checking problem, these logics are not the focus of this paper. Further, we introduce the parameterized model checking problem for parameterized BIP design, and show its undecidability.

Syntax For a set of index variables $\mathcal{I}$, the $ICTL^\star$ *state* and *path formulae* follow the grammar:

$$\theta ::= true \mid at(q, i) \mid \neg\theta \mid \theta_1 \wedge \theta_2 \mid \exists i :: type_j : \phi.\ \theta \mid \forall i :: type_j : \phi.\ \theta \mid \mathbf{E}\,\varphi \mid \mathbf{A}\,\varphi\,, \quad \text{(state formulae)}$$

$$\varphi ::= \theta \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \mathbf{X}\,\varphi \mid \mathbf{F}\,\varphi \mid \mathbf{G}\,\varphi \mid \varphi_1 \mathbf{U}\,\varphi_2\,. \quad \text{(path formulae)}$$

where $q \in \bigcup_{0 \le j < k} \mathbb{Q}_j$ is a location, $i \in \mathcal{I}$ is an index, and ϕ is a formula in Presburger arithmetic over size variables $\bar{n}$ and index variables from the set $\mathcal{I}$.

Semantics Fix a BIP model $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1}\rangle^{\bar{N},\Gamma}$ and its transition system $M = \langle S, s_0, \Gamma, R\rangle = TS(\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1}\rangle^{\bar{N},\Gamma})$ as per Definition 2.4. To evaluate Presburger formulae, we use the first-order structure $PA = \left\langle \mathbb{N}, 0, 1, \le, +, \bar{N} \right\rangle$. The semantics of $ICTL^\star$ formulae is defined inductively using M and PA. We only briefly discuss semantics to highlight the role of quantifiers in indexed temporal logics. For further discussions, we refer the reader to the textbook [12].

State formulae are interpreted over a configuration s and a valuation of index variables $\sigma : \mathcal{I} \to \mathbb{N}$ (the semantics of Boolean operators and universal quantifiers is defined in the standard way):

$$\begin{aligned}
&M, s, \sigma \models_{\text{ICTL}^\star} at(q, i) && \textit{iff} && q = s(j, \sigma(i)),\ \text{where } q \in \mathbb{Q}_j \\
&M, s, \sigma \models_{\text{ICTL}^\star} \exists i :: type_j : \phi.\ \theta && \textit{iff} && PA, \sigma[i \mapsto l] \models_{\text{FO}} \phi \text{ and } M, s, \sigma[i \mapsto l] \models_{\text{ICTL}^\star} \theta \text{ hold,} \\
& && && \text{for some } l \in [0, N_j) \\
&M, s, \sigma \models_{\text{ICTL}^\star} \mathbf{E}\,\varphi && \textit{iff} && M, \pi, \sigma \models_{\text{ICTL}^\star} \varphi \text{ for some infinite path } \pi \text{ starting from } s
\end{aligned}$$

Path formulae are interpreted over an infinite path π, and the valuation function σ as follows (the semantics for Boolean operators and temporal operators $\mathbf{F}$ and $\mathbf{G}$ is defined in the standard way):

$$
\begin{array}{lll}
M, \pi, \sigma \models_{\text{ICTL}^\star} \theta & \textit{iff} & M, s, \sigma \models_{\text{ICTL}^\star} \theta, \text{ where } s \text{ is the first configuration of the path } \pi \\
M, \pi, \sigma \models_{\text{ICTL}^\star} \mathbf{X}\varphi & \textit{iff} & M, \pi^1, \sigma \models_{\text{ICTL}^\star} \varphi \\
M, \pi, \sigma \models_{\text{ICTL}^\star} \varphi_1 \mathbf{U} \varphi_2 & \textit{iff} & \exists j \geq 0.\ M, \pi^j, \sigma \models_{\text{ICTL}^\star} \varphi_2 \text{ and } \forall i < j.\ M, \pi^i, \sigma \models_{\text{ICTL}^\star} \varphi_1,
\end{array}
$$

where π^i is the suffix of the path π starting with the i^{th} configuration.

Finally, given a formula φ without free variables, we say that M satisfies φ, written as $M \models_{\text{ICTL}^\star} \varphi$, if $M, s_0, \sigma_0 \models_{\text{ICTL}^\star} \varphi$ for the valuation σ_0 that assigns zero to each index from the set $\mathcal{I}$. The choice of σ_0 is arbitrary, as for all σ, it holds that $M, s_0, \sigma \models_{\text{ICTL}^\star} \varphi$ if and only if $M, s_0, \sigma_0 \models_{\text{ICTL}^\star} \varphi$.

Now we are in the position to formulate the parameterized model checking problem for BIP:

▶ **Problem 4.1** (Parameterized model checking)**.** *The verification problem for a parameterized* BIP *model* $\langle \mathbb{B}, \bar{n}, \psi, \epsilon \rangle$ *and an ICTL* state formula* θ *without free variables, is whether every instance* $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle^{\bar{N}, \Gamma}$ *satisfies* θ*.*

Not surprisingly, Problem 4.1 is undecidable in general. For instance, one can use the proof idea [16] to obtain the following theorem. We do not give a detailed proof here: to a large extent, it repeats the encoding of a unidirectional token ring, which we discuss later in Section 5.4.

▶ **Theorem 4.2** (Undecidability)**.** *Given a two-counter machine* M_2*, one can construct an ICTL*-formula* $\mathbf{G}\,\neg\texttt{halt}$ *and a parameterized BIP model* $\mathcal{B} = \langle \mathbb{B}, \bar{n}, \psi, \epsilon \rangle$ *that simulates* M_2 *and has the property:* M_2 *does not halt if and only if* $\langle \mathbb{B}_0, \ldots, \mathbb{B}_{k-1} \rangle^{\bar{N}, \Gamma} \models \mathbf{G}\,\neg\texttt{halt}$ *for all instances of* $\mathcal{B}$*.*

5 Identifying the architecture of a parameterized BIP model

In the non-parameterized case, knowing the architecture is not crucial, as there are model checking algorithms that apply in general to arbitrary finite transition systems. However, the architecture dramatically affects decidability of *parameterized* model checking. Architecture identification plays an important step in our verification framework. In this section, we show how to identify system architectures automatically, and present applications to verification.

Our framework For the sake of exposition, we assume that parameterized BIP models have only one component type. Our identification framework extends easily to the general case.

Given an architecture $\mathcal{A}$, e.g., the token ring architecture, an expert in parameterized model checking creates formula templates in FOIL (*FOIL-templates*) and in temporal logic (*TL-templates*). *FOIL-templates* describe the system topology and communication mechanism for the architecture $\mathcal{A}$. *TL-templates* describe the behaviour of the component type required by the architecture $\mathcal{A}$, e.g., in a token ring, a component which does not have the token cannot send the token. These templates are designed once for all parameterized BIP models compliant with $\mathcal{A}$. In the sequel, *TL-templates* are only used for token rings, thus we omit them from the discussion of other architectures.

Given a parameterized BIP model $\langle \langle \mathbb{B} \rangle, \langle n \rangle, \psi, \epsilon \rangle$—not necessarily compliant with the architecture $\mathcal{A}$—the templates for the architecture $\mathcal{A}$ are instantiated to FOIL formulae $\varphi_1^{FOIL}, \ldots, \varphi_m^{FOIL}$, and temporal logic formulae $\varphi_1^{TL}, \ldots, \varphi_\ell^{TL}$. The FOIL formulae guarantee that the set of interactions expressed by the FOIL formula ψ adheres to $\mathcal{A}$. The temporal logic formulae guarantee that the behaviour of the component type $\mathbb{B}$ adheres to $\mathcal{A}$. The *identification criterion* is as follows: if $\varphi_1^{FOIL} \wedge \cdots \wedge \varphi_m^{FOIL}$ is valid and $\mathbb{B} \models_{\text{TL}} \varphi_1^{TL} \wedge \cdots \wedge \varphi_\ell^{TL}$ holds, then the parameterized model $\langle \langle \mathbb{B} \rangle, \langle n \rangle, \psi, \epsilon \rangle$ is compliant with the architecture $\mathcal{A}$. In practice, we use an SMT solver to check validity of the FOIL formulae and a model checker to check that the component type $\mathbb{B}$ satisfies the temporal formulae.

In the rest of this section we construct FOIL-templates and TL-templates for well-known architectures: cliques of processes communicating via broadcast, cliques of processes communicating via rendezvous, token rings, and server-client systems in which processes are organized in a star and communicate via rendezvous. We show that the provided templates identify the architectures in a sound way.

5.1 The common templates for BIP semantics

As we discussed in Section 3.2, not every FOIL structure induces a BIP interaction. We show that one can write a FOIL-template that restricts FOIL structures to induce BIP interactions. The following template $\eta^{FOIL}_{interaction}(\mathbb{P}_0)$ expresses that there is no component with more than one active port: $\forall j :: type_0.\ \bigwedge_{p,q \in \mathbb{P}_0,\ q \neq p} \neg p(j) \vee \neg q(j)$

As expected, the template $\eta^{FOIL}_{interaction}(\mathbb{P}_0)$ restricts FOIL structures to BIP interactions:

▶ **Proposition 5.1.** *Let $\mathbb{P}_0$ be a set of ports, and η be the instantiation of $\eta^{FOIL}_{interaction}$ with $\mathbb{P}_0$. A FOIL structure ξ satisfies η if and only if ξ induces an interaction.*

To express that a component has at least one active port, we introduce template $active(j) \equiv \bigvee_{p \in \mathbb{P}_0} p(j)$. To simplify notation, parameterization of $active(j)$ by $\mathbb{P}_0$ is omitted.

5.2 Pairwise rendezvous in a clique

In a BIP model, components are said to communicate by binary rendezvous, if all the allowed interactions consist of exactly two ports. The communication is said to be by *pairwise rendezvous*, if there is a binary rendezvous between every two components. Pairwise rendezvous has been widely used as a basic primitive in the parameterized model checking literature, e.g., in [18, 3].

FOIL-templates We construct a template using two formulae $\eta^{FOIL}_{\leq 2}(\mathbb{P}_0)$ and $\eta^{FOIL}_{\geq 2}(\mathbb{P}_0)$:

- The formula $\eta^{FOIL}_{\leq 2}(\mathbb{P}_0)$ expresses that every interaction has at most two ports:
  $\forall i, j, \ell :: type_0.\ active(i) \wedge active(j) \wedge active(\ell) \rightarrow i = j \vee j = \ell \vee i = \ell.$
- The formula $\eta^{FOIL}_{\geq 2}(\mathbb{P}_0)$ expresses that every interaction has at least two ports:
  $\exists i, j :: type_0 : i \neq j.\ active(i) \wedge active(j).$

We show that the combination of $\eta^{FOIL}_{interaction}$, $\eta^{FOIL}_{\geq 2}$, and $\eta^{FOIL}_{\leq 2}$ defines pairwise rendezvous communication in cliques of all sizes:

▶ **Theorem 5.2.** *Given a one-type parameterized* BIP *model $\langle\langle \mathbb{B} \rangle, \langle n \rangle, \psi, \epsilon\rangle$, if $(\psi \wedge \eta^{FOIL}_{interaction}) \leftrightarrow (\eta^{FOIL}_{interaction} \wedge \eta^{FOIL}_{\geq 2} \wedge \eta^{FOIL}_{\leq 2})$ is valid, then for every instance $\mathbb{B}^{N,\Gamma}$, the following holds:*

1. *every interaction is of size 2, that is, $|\gamma| = 2$ for $\gamma \in \Gamma$, and*
2. *for every pair of indices i and j such that $0 \leq i, j < N$ and $i \neq j$ and every pair of ports $p, q \in \mathbb{P}_0$, there is a FOIL structure ξ such that $\xi \models_{\text{FOIL}} \psi \wedge p(i) \wedge q(j)$.*

Proof. Fix an instance $\mathbb{B}^{N,\Gamma}$ of $\langle\langle \mathbb{B} \rangle, \langle n \rangle, \psi, \epsilon\rangle$.

To show Point 1, fix an interaction γ of $\mathbb{B}^{N,\Gamma}$. By Definition 3.3, there is a FOIL structure ξ such that $\xi \models_{\text{FOIL}} \psi$ and $\gamma = \gamma_\xi$. As ξ induces an interaction, by Proposition 5.1, we immediately have that γ_ξ satisfies the instantiation of $\eta^{FOIL}_{interaction}$. Hence, since $(\psi \wedge \eta^{FOIL}_{interaction}) \leftrightarrow (\eta^{FOIL}_{interaction} \wedge \eta^{FOIL}_{\geq 2} \wedge \eta^{FOIL}_{\leq 2})$ is valid we conclude that ξ also satisfies $\eta^{FOIL}_{\geq 2} \wedge \eta^{FOIL}_{\leq 2}$. This immediately gives us the required equality $|\gamma_\xi| = 2$.

To show Point 2, fix a pair of indices i and j such that $0 \leq i, j < N$ and $i \neq j$ and a pair of ports $p, q \in \mathbb{P}_0$. The set $\gamma = \{(p, i), (q, j)\}$ is an interaction. Obviously, one can construct a FOIL structure ξ that induces γ. Since $i \neq j$ and $|\gamma_\xi| = 2$, it holds that $\xi \models_{\text{FOIL}} \eta^{FOIL}_{interaction} \wedge \eta^{FOIL}_{\geq 2} \wedge \eta^{FOIL}_{\leq 2}$. Thus,

since $(\psi \wedge \eta^{FOIL}_{interaction}) \leftrightarrow (\eta^{FOIL}_{interaction} \wedge \eta^{FOIL}_{\geq 2} \wedge \eta^{FOIL}_{\leq 2})$ is valid, it follows that $\xi \models_{\text{FOIL}} \psi$. From this and that ξ induces the interaction γ, we conclude that $\xi \models_{\text{FOIL}} \psi \wedge p(i) \wedge q(j)$. ◀

In Theorem 5.2, the right-hand side of the equivalence does not restrict which pairs of ports may interact, e.g., it does not require the ports to be the same. Thus, if ψ is more restrictive than the right-hand side of the equivalence, validity will not hold. Obviously, one can further restrict the equivalence to reflect additional constraints on the allowed pairs of ports. Moreover, one may restrict which ports are required by the template to communicate via pairwise rendezvous for compositionality, e.g. to allow other ports to participate in other communication primitives and in internal transitions. (One may augment or restrict the templates of all the architectures below similarly.)

Applications Theorem 5.2 gives us a criterion for identifying parameterized BIP models, where all processes may interact with each other using rendezvous communication. To verify such parameterized BIP models, we can immediately invoke the seminal result by German & Sistla [18, Sec. 4]. Their result applies to specifications written in indexed linear temporal logic without the operator $\mathbf{X}$.

More formally, we say that an *ICTL** path formula $\chi(i)$ is a 1-*LTL*$\backslash X$ formula, if χ has only one index variable i and χ does not contain quantifiers $\exists$, $\forall$, $\mathbf{A}$, $\mathbf{E}$, nor temporal operator $\mathbf{X}$. Given a parameterized BIP model $\langle\langle \mathbb{B} \rangle, \langle n \rangle, \psi, \epsilon\rangle$ and a 1-*LTL*$\backslash X$ formula χ, one can check in polynomial time, whether every instance $\mathbb{B}^{N,\Gamma}$ satisfies the formula $\mathbf{E}\ \exists i :: type_0 :\ true.\ \chi(i)$.

5.3 Broadcast in a clique

In BIP, components communicate via broadcast, if there is a "trigger" component whose sending port is active, and the other components either have their receiving port active, or have no active ports. In this section, we denote the sending port with *send* and the receiving port with *receive*. Our results can be easily extended to treat multiple sending and receiving ports. In a broadcast step, all the components with the active ports make their transitions simultaneously. Broadcasts were extensively studied in the parameterized model checking literature [17, 23].

One way to enforce all the processes to receive a broadcast, if they are ready to do so, is to use priorities in BIP: an interaction has priority over any of its subsets. In this paper, we consider BIP without priorities. In this case, one can express broadcast by imposing the following restriction on the structure of the component type $\mathbb{B}$: *every location has a transition labeled with the port receive*. This restriction enforces all interactions to involve all the components, though some of the components may not change their location by firing a self-loop transition. This requirement can be statically checked on the transition system of $\mathbb{B}$, and if the component type does not fulfill the requirement, it is easy to modify the component type's transition system by adding required self-loops.

FOIL-templates First, we define the formula $\eta^{FOIL}_{bcast}(\mathbb{P}_0)$, which guarantees that every interaction includes one sending port by one component and the receiving ports of the other components:

$$\exists i :: type_0.\ send(i) \wedge \forall j :: type_0 : j \neq i.\ receive(j)$$

We show that the combination of $\eta^{FOIL}_{interaction}$ and η^{FOIL}_{bcast} defines broadcast in cliques of all sizes:

▶ **Theorem 5.3.** *Given a one-type parameterized* BIP *model* $\langle\langle \mathbb{B} \rangle, \langle n \rangle, \psi, \epsilon\rangle$*, if* $(\psi \wedge \eta^{FOIL}_{interaction}) \leftrightarrow (\eta^{FOIL}_{interaction} \wedge \eta^{FOIL}_{bcast})$ *is valid, then for every instance* $\mathbb{B}^{N,\Gamma}$*, the following holds:*

1. *every interaction consists of one send port and* $N - 1$ *receive ports.*
2. *for every index c, such that* $0 \leq c < N$*, there is a FOIL structure* ξ *satisfying the following:* $\xi \models_{\text{FOIL}} \psi \wedge send(c) \wedge \forall j :: type_0 : j \neq c.\ receive(j)$.

Proof. The proof follows the same principle as the proof of Theorem 5.2. ◀

Applications Theorem 5.3 gives a criterion for identifying parameterized BIP models in which all components may send and receive broadcast. Its implications are two-fold. First, it is well-known that parameterized model checking of safety properties is decidable [1] (cf. the discussion in [17]), and there are tools for well-structured transition systems applicable to model checking of parameterized BIP. Second, parameterized model checking of liveness properties is undecidable [17]. From the user perspective, this indicates the need to construct abstractions, or to use semi-decision procedures.

Identifying sending and receiving ports Now we illustrate how to automatically detect the sending and receiving ports in a parameterized BIP model. We say that a port $p \in \mathbb{P}_0$ in the component type may be a sending port, if in every interaction exactly one component uses this port. Similarly, we say that a port $q \in \mathbb{P}_0$ in the component type may be a receiving port, if in every interaction all but one component use this port. Intuitively, we have to enumerate all port types and check whether they are acting as sending ports or receiving ports. Formally, to find whether p is a potential sending port and q is a potential receiving port, we check whether the following is valid:

$$\psi \ \wedge \ \eta^{FOIL}_{interaction} \ \wedge \ \exists i :: type_0.\ (p(i) \vee q(i)) \ \rightarrow \ (\exists i :: type_0.\ p(i) \wedge \forall j :: type_0 : j \neq i.\ q(j))$$

5.4 Token rings

Token ring is a classical architecture: (i) all processes are arranged in a ring, (ii) the ring size is parameterized but fixed in each run, and (iii) one component owns the token and can pass the token to its neighbor(s). It is easy to express token-passing with rendezvous, so we re-use the templates from Section 5.2. We assume that there is a pair of ports: the port *send* giving away the token and the port *receive* accepting the token. We do not allow the token to change its type, as the parameterized model checking problem is undecidable in this case [26, 16]. Nevertheless, it is easy to extend our results to multiple token types. Here the token is passed in one direction, that is, every component may only receive the token from one neighbor and may only send the token to its other neighbor.

TL-templates Following the standard assumption [16], we require that every process sends and receives the token infinitely often. We encode this requirement as a local constraint in a form of an LTL formula that is checked against the component type (and not against a BIP instance):

$$\mathbf{G}\left(receive \rightarrow \mathbf{X}\ (\neg receive\ \mathbf{U}\ \ send)\right) \wedge \mathbf{G}\left(send \rightarrow \mathbf{X}\,(\neg send\ \mathbf{U}\ \ receive)\right)$$

The left conjunct forces a component that has the token to eventually send it. The right conjunct prevents a component from sending the token twice before receiving it back.

FOIL-templates We extend the pairwise rendezvous templates with a formula $\eta^{FOIL}_{uniring}(\mathbb{P}_0)$ that restricts the interactions to be performed only among the neighbors in one direction:

$$\exists i, j :: type_0.\ (j = (i+1) \bmod n_0).\ send(i) \wedge receive(j)$$

The modulo notation "$j = (i+1) \bmod n_0$" can be seen as syntactic sugar, as it expands into $(i = n_0 - 1 \rightarrow j = 0) \wedge (i < n_0 - 1 \rightarrow j = i + 1)$.

▶ **Theorem 5.4.** *Given a one-type parameterized* BIP *model* $\langle \langle \mathbb{B} \rangle, \langle n \rangle, \psi, \epsilon \rangle$*, if* $(\psi \wedge \eta^{FOIL}_{interaction}) \leftrightarrow (\eta^{FOIL}_{interaction} \wedge \eta^{FOIL}_{\geq 2} \wedge \eta^{FOIL}_{\leq 2} \wedge \eta^{FOIL}_{uniring})$ *is valid, then every instance* $\mathbb{B}^{N,\Gamma}$ *satisfies:*

1. *every interaction* $\gamma \in \Gamma$ *is of the form* $\{send(c), receive(d)\}$ *for some indices* c *and* d *such that* $0 \leq c, d < N$ *and* $d = (c+1) \bmod N$*, and*

2. *for every index c such that $0 \le c < N$ and the index $d = (c+1) \bmod N$, there is a FOIL structure ξ such that $\xi \models_{\text{FOIL}} \psi \wedge send(c) \wedge receive(d)$.*

Proof. The proof follows the same principle as the proof of Theorem 5.2. ◀

Distributing the token The token ring architecture assumes that initially only one component has the token. Emerson & Namjoshi [16] assumed that the token was distributed using a "daemon", but this primitive is obviously outside of the token ring architecture. Our framework encompasses token distribution. To this end, we restrict the transition system of the component as follows:

- We assume that the location set $\mathbb{Q}_0$ of the component type $\mathbb{B}_0$ is partitioned into two sets: $\mathbb{Q}_0^{tok}$ is the set of locations possessing the token, and $\mathbb{Q}_0^{ntok}$ is the set of locations without the token. The initial location does not possess the token: $\ell^0 \in \mathbb{Q}_0^{ntok}$.
- We assume that there are two auxiliary ports called *master* and *slave* that are only used in a transition from the initial location ℓ^0. There are only two transitions involving ℓ^0: the transition from ℓ^0 to a location in $\mathbb{Q}_0^{tok}$ that broadcasts via the port *master*, and the transition from ℓ^0 to a location in $\mathbb{Q}_0^{ntok}$ that receives the broadcast via the port *slave*. The broadcast interaction can be checked with the constraints similar to those in Section 5.3.

Applications Theorem 5.4 gives us a criterion for identifying parameterized BIP models that express a unidirectional token ring. This criterion has a great impact: one can apply non-parameterized BIP tools to verify parameterized BIP designs expressing token rings. As Emerson & Namjoshi showed in their celebrated paper [16], to verify parameterized token rings, it is sufficient to run model checking on rings of small sizes. The bound on the ring size—called a *cut-off*—depends on the specification and typically requires two or three components.

5.5 Pairwise rendezvous in a star

In a star architecture, one component acts as the center, and the other components communicate only with the center. The components communicate via rendezvous (considered in Section 5.2). This architecture is used in client-server applications. Parameterized model checking for the star architecture was investigated by German & Sistla [18]. We assume that a parameterized BIP model contains two component types: $\mathbb{B}_0$ with only one instance, and $\mathbb{B}_1$ that may have many instances.

FOIL-templates The requirements of rendezvous communication are defined in Section 5.2. We add the restriction η^{FOIL}_{center} that the center is involved in every interaction: $\exists i :: type_0.\ active_0(i)$. By restricting ϵ to have only one instance of type $\mathbb{B}_0$, we arrive at Theorem 5.5, which to a large extent is a consequence of Theorem 5.2.

▶ **Theorem 5.5.** *Given a two-component parameterized* BIP *model $\langle\langle \mathbb{B}_0, \mathbb{B}_1\rangle, \langle n_0, n_1\rangle, \psi, \epsilon\rangle$, if $(\psi \wedge \eta^{FOIL}_{interaction}) \leftrightarrow (\eta^{FOIL}_{interaction} \wedge \eta^{FOIL}_{\ge 2} \wedge \eta^{FOIL}_{\le 2} \wedge \eta^{FOIL}_{center})$ and $\epsilon \leftrightarrow (n_0 = 1)$ are both valid, then every instance $\langle \mathbb{B}_0, \mathbb{B}_1\rangle^{\langle N_0, N_1\rangle, \Gamma}$ admits only the rendezvous interactions with the center, i.e., the only component of type $\mathbb{B}_0$.*

Applications Theorem 5.5 gives us a criterion for identifying parameterized BIP models, where the user processes communicate with the coordinator via rendezvous. To verify such parameterized BIP models, we can immediately invoke several results by German & Sistla [18, Sec. 3]. First, one can analyze such parameterized BIP models for deadlocks, which is of extreme importance to the practical applications of BIP. Second, the results [18] reduce parameterized model checking to reachability in Petri nets, which allows one to use the existing tools for Petri nets.

Benchmark	Architecture model	Outcome	Time (sec.)	Memory (MB)
Milner's scheduler	uni-directional token ring	**positive**	0.068	≤ 10
Milner's scheduler	broadcast in clique	negative	0.016	≤ 10
Semaphore	pairwise rendezvous in star	**positive**	0.096	≤ 10
Semaphore	pairwise rendezvous in clique	negative	0.084	≤ 10
Barrier	broadcast in clique	**positive**	0.028	≤ 10
Barrier	pairwise rendezvous in star	negative	0.008	≤ 10

Table 1 Experimental results of identifying architecture models. The column "Outcome" indicates, whether the benchmark was recognized to have the given architecture (positive), or not (negative). The experiments were performed on a 64-bit Linux machine with 2.8GHz × 4 CPU and 7.8GiB memory.

6 Prototype implementation and experiments

We have implemented a prototype of the framework introduced in Section 5. This prototype uses the following architecture templates: (a) pairwise rendezvous and broadcast in cliques, (b) token rings, (c) and pairwise rendezvous in stars. As described in Section 5 (see *our framework*), given a parameterized BIP model, the tool constructs a set of FOIL formulae and a set of temporal formulae. The parameterized BIP model follows a predefined architecture, if the FOIL formulae are valid and the component types satisfy the temporal formulae. Our implementation uses nuXmv [11] to check temporal formulae and Z3 [14] to check validity of first-order formulae. FOIL formulae are translated to first-order formulae by guarding the range of quantification explicitly, e.g. $\exists i :: type_0.\ \theta$ is substituted with $\exists i.\ 0 \leq i < n_0 \wedge \theta$. To deal with quantifiers, we run a customized solver with tactic 'qe' (i.e. quantifier elimination). The implementation and benchmarks are available at `http://risd.epfl.ch/parambip`.

Table 1 summarizes our experiments with three benchmarks. We conducted each experiment using two kinds of templates: the expected architecture of the benchmark, and an architecture different from the expected one. In all cases, the architectures were identified as expected. Our preliminary results demonstrate both correctness and efficiency of our approach.

7 Related work and conclusions

We have shown that our framework encompasses several prominent parameterized model checking techniques. To our understanding, the other seminal results can be integrated into our framework: the cut-off results for disjunctive and conjunctive guards [15], network decomposition techniques [13, 3], and techniques based on well-structured transition systems [1] and monotonic abstraction [2].

First-order interaction logic extends propositional interaction logic [6, 7], which was also extended by Dy-BIP [10] and configuration logic [21]. Dy-BIP extends propositional interaction logic with quantification to define interaction topology independent of the number of component instances. It uses dedicated *history variables* to break the symmetry and specify that, throughout the system execution, successive interactions happen among the same components. Dy-BIP does not have a mechanism, such as indexing, to statically distinguish instances of the same component type. Hence, many architectures, e.g., token rings, cannot be expressed. Configuration logic uses higher-order formulae to represent sets of topologies. It does not use indexing either, thereby requiring the second-order extension to express simple architectures such as token rings and linear architectures. Finally, no decidability results or decision procedures have been proposed for the configuration logic yet.

In the future, we will study second-order extensions of FOIL to express more complex architectures such as server-client whose coordinator is chosen non-deterministically. In the long term, we

plan to implement a tool that integrates multiple parameterized model checking techniques and uses our framework to guide the verification of parameterized BIP designs. FOIL can also be seen as a specification language for BIP interactions and used for their synthesis similarly to [7]. Finally, it is worth investigating, whether FOIL can be extended to include priorities as in [8].

References

1 P. A. Abdulla, K. Cerans, B. Jonsson, and Y. Tsay. General decidability theorems for infinite-state systems. In *LICS*, 1996.
2 P. A. Abdulla, G. Delzanno, N. B. Henda, and A. Rezine. Monotonic abstraction: on efficient verification of parameterized systems. *Int. J. Found. Comput. Sci.*, 2009.
3 B. Aminof, T. Kotek, S. Rubin, F. Spegni, and H. Veith. Parameterized model checking of rendezvous systems. In *CONCUR*. Springer, 2014.
4 A. Basu, S. Bensalem, M. Bozga, J. Combaz, M. Jaber, T. Nguyen, and J. Sifakis. Rigorous component-based system design using the BIP framework. *Software, IEEE*, 2011.
5 S. Bliudze, A. Cimatti, M. Jaber, S. Mover, M. Roveri, W. Saab, and Q. Wang. Formal verification of infinite-state BIP models. In *ATVA*, 2015.
6 S. Bliudze and J. Sifakis. The algebra of connectors —structuring interaction in BIP. In *EMSOFT*, 2007.
7 S. Bliudze and J. Sifakis. Causal semantics for the algebra of connectors. *FMSD*, 2010.
8 S. Bliudze and J. Sifakis. Synthesizing glue operators from glue constraints for the construction of component-based systems. In *Software Composition*, 2011.
9 R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder. Decidability of parameterized verification. *Synthesis Lectures on Distributed Computing Theory*, 2015.
10 M. Bozga, M. Jaber, N. Maris, and J. Sifakis. Modeling dynamic architectures using Dy-BIP. In *Software Composition*. Springer, 2012.
11 R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta. The nuXmv symbolic model checker. In *CAV*, 2014.
12 E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
13 E. Clarke, M. Talupur, T. Touili, and H. Veith. Verification by network decomposition. In *CONCUR*, 2004.
14 L. De Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, 2008.
15 E. A. Emerson and V. Kahlon. Model checking guarded protocols. In *LICS*, 2003.
16 E. A. Emerson and K. S. Namjoshi. Reasoning about rings. In *POPL*, 1995.
17 J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. *LICS*, 1999.
18 S. M. German and A. P. Sistla. Reasoning about systems with many processes. *J. ACM*, 1992.
19 O. Grumberg and H. Veith. *25 years of model checking: history, achievements, perspectives*. Springer, 2008.
20 J. Y Halpern. Presburger arithmetic with unary predicates is π^1_1 complete. *J. of Symb. Logic*, 1991.
21 A. Mavridou, E. Baranov, S. Bliudze, and J. Sifakis. Configuration logics: Modelling architecture styles. In *FACS*, 2015.
22 Q.Wang and S. Bliudze. Verification of component-based systems via predicate abstraction and simultaneous set reduction. In *TGC*, 2015.
23 S. Schmitz and P. Schnoebelen. The power of well-structured systems. In *CONCUR*, 2013.
24 J. Sifakis. Rigorous system design. *Foundations and Trends in Electr. Design Automation*, 2013.
25 J. Sifakis. System design automation: Challenges and limitations. *Proc. of the IEEE*, 2015.
26 I. Suzuki. Proving properties of a ring of finite-state machines. *Inf. Process. Lett.*, 1988.
27 W. Thomas. *Languages, automata, and logic*. Springer, 1997.