Local Reasoning for Parameterized First Order Protocols

Rylo Ashmore(✉), Arie Gurfinkel(✉), and Richard Trefler(✉)

University of Waterloo, Waterloo, Canada
{rjashmor,arie.gurfinkel,trefler}@uwaterloo.ca

Abstract. First Order Logic (FOL) is a powerful reasoning tool for program verification. Recent work on Ivy shows that FOL is well suited for verification of parameterized distributed systems. However, specifying many natural objects, such as a ring topology, in FOL is unexpectedly inconvenient. We present a framework based on FOL for specifying distributed multi-process protocols in a process-local manner together with an implicit network topology. In the specification framework, we provide an auto-active analysis technique to reason about the protocols locally, in a process-modular way. Our goal is to mirror the way designers often describe and reason about protocols. By hiding the topology behind the FOL structure, we simplify the modelling, but complicate the reasoning. To deal with that, we use an oracle for the topology to develop a sound and relatively complete proof rule that reduces reasoning about the implicit topology back to pure FOL. This completely avoids the need to axiomatize the topology. Using the rule, we establish a property that reduces verification to a fixed number of processes bounded by the size of local neighbourhoods. We show how to use the framework on two examples, including leader election on a ring.

1 Introduction

Auto-active [7] and automated verification engines are now commonly used to analyze the behavior of safety- and system-critical multi-process distributed systems. Applying the analysis techniques early in the design cycle has the added advantage that any errors or bugs found are less costly to fix than if one waits until the system is deployed. Therefore, it is typical to seek a proof of safety for *parametric* designs, where the number of participating program components is not yet determined, but the inter-process communication fits a given pattern, as is common in routing or communication protocols, and other distributed systems. Recently, Ivy [16] has been introduced as a novel auto-active verification technique (in the style of Dafny [7]) for reasoning about parameterized systems. Ivy models protocols in First Order Logic (FOL). The verification conditions are compiled (with user help) to a decidable fragment of FOL, called Effectively Propositional Reasoning (EPR) [17]. Ivy is automatic in the sense that the verification engineer only provides an inductive invariant. Furthermore, unlike Dafny,

J. M. Badger and K. Y. Rozier (Eds.): NFM 2019, LNCS 11460, pp. 36–53, 2019.
https://doi.org/10.1007/978-3-030-20652-9_3

$$\forall x, y, z \cdot btw(x, y, z) \Rightarrow btw(y, z, x)$$
$$\forall w, x, y, z \cdot btw(w, x, y) \wedge btw(w, y, z) \Rightarrow btw(w, x, z)$$
$$\forall w, x, y \cdot btw(w, x, y) \Rightarrow \neg btw(w, y, x)$$
$$\forall w, x, y \cdot distinct(w, x, y) \Rightarrow (btw(w, x, y) \vee btw(w, y, x))$$
$$\forall a, b \cdot (next(a, b) \iff \forall x \cdot x \neq a \wedge x \neq b \Rightarrow btw(a, b, x))$$

Fig. 1. A description of a unidirectional ring in FOL as presented by Ivy [16].

it guarantees that the verification is never stuck inside the decision procedure (verification conditions are decidable).

In representing a protocol in Ivy, an engineer must formally specify the entire protocol, including the topology. For instance, in verifying the leader election on a ring, Ivy requires an explicit axiomatization of the ring topology, as shown in Fig. 1. The predicate $btw(x, y, z)$ means that a process y is between processes x and z in the ring; similarly, $next(a, b)$ means that b is an immediate neighbour of a on the ring. All (finite) rings satisfy the axioms in Fig. 1. The converse is not true in general. For instance, take the rationals $\mathbb{Q}$ and let $btw(x, y, z)$ be defined as $x < y < z \ \vee \ y < z < x \ \vee \ z < x < y$. All axioms of btw are satisfied, but the only consistent interpretation of *next* is an empty set. This satisfies all the axioms, but does not define a ring. For the axioms in Fig. 1, all *finite* models of btw and *next* describe rings. This is not an issue for Ivy, since infinite models do not need to be considered for EPR. Such reasoning is non-trivial and is a burden on the verification engineer. As another example, we were not able to come up with an axiomatization of rings of alternating red and black nodes (shown in Fig. 2a) within EPR. In general, a complete axiomatization of the topology might be hard to construct.

In this paper, we propose to address this problem by specifying the topology independently of process behaviour. We present a framework which separates the two and provides a clean way to express the topology. We then specify our transitions locally, as this is a natural and common way to define protocols. Once these preliminaries are done, we provide a process-local proof rule to verify properties of the system. To generate the proof rule, we offload topological knowledge to an oracle that can answer questions about the topology. Finally, we prove various properties of the proof rule.

In summary, the paper makes the following contributions. First, in Sect. 3, we show how to model protocols locally in FOL. This is an alternative to the global modelling used in Ivy. Second, in Sect. 4, we show a proof rule with verification conditions (VC) in FOL, which are often in EPR. When the VC is in EPR, this gives an engineer a mechanical check of inductiveness. This allows reasoning about topology without axiomatizing it. Third, in Sect. 5, we show that our proof rule (a) satisfies a small model property, and (b) is relatively complete. The first guarantees the verification can be done on small process domains; the second ensures that our proof rule is relatively expressive.

We illustrate our approach on two examples. First, as a running example, motivated by [13], is a protocol on rings of alternating red and black nodes. These rings have only rotational symmetry, however, they have substantial local symmetry [8,12,13] consisting of two equivalence classes, one of red nodes, and one of black nodes. Second, in Sect. 6, we consider a modified version of the leader election protocol from Ivy [16]. This is of particular interest, since the local symmetry of [8,12,13] has not been applied to leader election. We thus extend [8,12,13] by both allowing more symmetries and infinite-state systems.

2 Preliminaries

FOL Syntax and Semantics. We assume some familiarity with the standard concepts of many sorted First Order Logic (FOL). A signature Σ consists of sorted predicates, functions, and constants. Terms are variables, constants, or (recursively) k-ary functions applied to k other terms of the correct sort. For every k-ary predicate P and k terms $t_1, \ldots, t_k$ of the appropriate sort for P, the formula $P(t_1, \ldots, t_k)$ is a well-formed formula (wff). Wffs are then boolean combinations of formulae and universally or existentially quantified formulae. Namely, if ψ and φ are wffs, then so are $(\psi \wedge \varphi)$, $(\psi \vee \varphi)$,$(\neg\psi)$, $(\psi \Rightarrow \varphi)$, $(\psi \iff \varphi)$, $(\forall x \cdot \psi)$, and $(\exists x \cdot \psi)$. A variable x in a formula ψ is bound if it appears under the scope of a quantifier. A variable not bound is free. A wff with no free variables is called a sentence. For convenience, we often drop unnecessary parenthesis, and use $\top$ to denote true and $\bot$ to denote false.

An FOL interpretation $\mathcal{I}$ over a domain D assigns every k-ary predicate P a sort-appropriate semantic interpretation $\mathcal{I}(P) : D^k \to \{T, F\}$; to every k-ary function f a sort-appropriate interpretation $\mathcal{I}(f) : D^k \to D$, and to every constant c an element $\mathcal{I}(c) \in D$. Given an interpretation $\mathcal{I}$ and a sentence ψ, then either ψ is true in $\mathcal{I}$ (denoted, $\mathcal{I} \models \psi$), or ψ is false in $\mathcal{I}$ (denoted $\mathcal{I} \not\models \psi$). The definition of the models relation is defined on the structure of the formula as usual, for example, $\mathcal{I} \models (\varphi \wedge \psi)$ iff $\mathcal{I} \models \varphi$ and $\mathcal{I} \models \psi$. We write $\models \varphi$ if for every interpretation $\mathcal{I}$, $\mathcal{I} \models \varphi$.

We write $\mathcal{I}(\Sigma')$ to denote a restriction of an interpretation $\mathcal{I}$ to a signature $\Sigma' \subseteq \Sigma$. Given disjoint signatures Σ, Σ' and corresponding interpretations $\mathcal{I}, \mathcal{I}'$ over a fixed domain D, we define $\mathcal{I} \oplus \mathcal{I}'$ to be an interpretation of $\Sigma \cup \Sigma'$ over domain D defined such that $(\mathcal{I} \oplus \mathcal{I}')(t) = \mathcal{I}(t)$ if $t \in \Sigma$, and $(\mathcal{I} \oplus \mathcal{I}')(t) = \mathcal{I}'(t)$ if $t \in \Sigma'$. Given interpretation $\mathcal{I}$ and sub-domain $D' \subseteq D$ where D' contains all constants, we let $\mathcal{I}(D')$ be the interpretation restricted to domain D'.

FOL Modulo Structures. We use an extension of FOL to describe structures, namely graphs. In this case, the signature Σ is extended with some pre-defined functions and predicates, and the interpretations are restricted to particular intended interpretations of these additions to the signature. We identify a structure class $\mathcal{C}$ with its signature $\Sigma^{\mathcal{C}}$ and an intended interpretation. We write $FOL^{\mathcal{C}}$ for First Order Logic over the structure class $\mathcal{C}$. Common examples are FOL over strings, FOL over trees, and other finite structures.

A structure $\mathcal{S} = (D, \mathcal{I})$ is an intended interpretation $\mathcal{I}$ for structural predicates/functions $\Sigma^{\mathcal{C}}$ over an intended domain D. A set of structures is denoted $\mathcal{C}$. The syntax of $FOL^{\mathcal{C}}$ is given by the syntax for FOL with signature $\Sigma \uplus \Sigma^{\mathcal{C}}$ (where Σ is an arbitrary disjoint signature). For semantics, any FOL interpretation $\mathcal{I}$ of signature Σ leads to an $FOL^{\mathcal{C}}$ interpretation $\mathcal{I} \oplus \mathcal{I}^{\mathcal{C}}$ of the signature $\Sigma \uplus \Sigma^{\mathcal{C}}$. We write $\models_{\mathcal{C}} \varphi$ iff every $FOL^{\mathcal{C}}$ interpretation $\mathcal{I}$ satisfies $\mathcal{I} \models \varphi$. We introduce a process sort *Proc* and require the intended domain D to be exactly the set of *Proc*-sorted elements, so that we put our intended structure on the processes.

First Order Transition Systems. We use First Order Transitions Systems from Ivy [15,16]. While the original definition was restricted to the EPR fragment of FOL, we do not require this. A transition system is a tuple $Tr = (S, S_0, R)$, where S is a set of states, $S_0 \subseteq S$ is a set of initial states, and $R \subseteq S \times S$ is a transition relation. A trace π is a (finite or infinite) sequence of states $\pi = s_0 \cdots s_i \cdots$ such that $s_0 \in S_0$ and for every $0 \leq i < |\pi|$, $(s_i, s_{i+1}) \in R$, where $|\pi|$ denotes the length of π, or ∞ if π is infinite. A transition system may be augmented with a set $B \subseteq S$ of "bad" states. The system is safe iff all traces contain no bad states. A set of states I is inductive iff $S_0 \subseteq I$ and if $s \in I$ and $(s, s') \in R$, then $s' \in I$. Showing the existence of an inductive set I that is disjoint from bad set B suffices to show a transition system is safe.

A First-Order Transition System Specification (FOTSS) is a tuple $(\Sigma, \varphi_0, \tau)$ where Σ is an FOL signature, φ_0 is a sentence over Σ and τ is a sentence over $\Sigma \uplus \Sigma'$, where $\uplus$ denotes disjoint union and $\Sigma' = \{t' \mid t \in \Sigma\}$. The semantics of a FOTSS are given by First Order Transition Systems (FOTS). Let D be a fixed domain. A FOTSS $(\Sigma, \varphi_0, \tau)$ defines a FOTS over D as follows: $S = \{\mathcal{I} \mid \mathcal{I} \text{ is an FOL interpretation over } D\}$, $S_0 = \{\mathcal{I} \in S \mid \mathcal{I} \models \varphi_0\}$, and $R = \{(\mathcal{I}_1, \mathcal{I}_2) \in S \times S \mid \mathcal{I}_1 \oplus \mathcal{I}_2' \models \tau\}$, where $\mathcal{I}'$ interprets Σ'. We may augment a FOTSS with a FOL sentence *Bad*, giving bad states in the FOTS by $\mathcal{I} \in B$ iff $\mathcal{I} \vDash Bad$. A FOTSS is safe if all of its corresponding FOTS Tr are safe, and is unsafe otherwise. That is, a FOTSS is unsafe if there exists at least one FOTS corresponding to it that has at least one execution that reaches a bad state. A common way to show a FOTSS is safe is to give a formula *Inv* such that $\models \varphi_0 \Rightarrow Inv$ and $\models Inv \wedge \tau \Rightarrow Inv'$. Then for any FOTS over domain D, the set $I \subseteq S$ given by $I = \{\mathcal{I} \in S \mid \mathcal{I} \models Inv\}$ is an inductive set, and $\models Inv \Rightarrow \neg Bad$ then suffices to show that the state sets I, B in the FOTS are disjoint. Finding an invariant *Inv* satisfying the above proves the system safe.

Example 1. Consider the following FOTSS:

$$\Sigma \triangleq \{Even, +, 1, var\} \qquad \varphi_0 \triangleq Even(var)$$
$$\tau \triangleq (var' = (var + 1) + 1) \wedge Unch(Even, +, 1) \qquad Bad \triangleq \neg Even(var)$$

where $Unch(Even, +, 1)$ means that *Even*, $+$, and 1 have identical interpretations in the pre- and post-states of τ.

Our intention is to model a program that starts with an even number in a variable *var* and increments *var* by 2 at every transition. It is an error if *var* ever

becomes odd. A natural invariant to conjecture is $Inv \triangleq Even(var)$. However, since the signature is uninterpreted, the FOTSS does not model our intention.

For example, let $D = \{0, 1, 2\}$, $\mathcal{I}_0(Even) = \{1, 2\}$, $\mathcal{I}_0(1) = 1$, $\mathcal{I}_0(+)(a, b) = a + b \mod 3$, and $\mathcal{I}_0(var) = 1$. Thus, $\mathcal{I}_0 \models \varphi_0$. Let $\mathcal{I}_1$ be the same as $\mathcal{I}_0$, except $\mathcal{I}_1(var) = 0$. Then, $\mathcal{I}_0 \oplus \mathcal{I}'_1 \models \tau$ and $\mathcal{I}_1 \models Bad$. Thus, this FOTSS is unsafe.

One way to explicate our intention in Example 1 is to axiomatize the uninterpreted functions and relations in FOL as part of φ_0 and τ. Another alternative is to restrict their interpretation to a particular structure. This is the approach we take in this paper. We define a First-Order (relative to $\mathcal{C}$) Transition System Specification (FOCTSS).

We need to be able to talk about the structural objects in $\Sigma^{\mathcal{C}}$, and so we require that every FOCTSS $(\Sigma, \varphi_0, \tau)$ be an FOTSS with $\Sigma^{\mathcal{C}} \subseteq \Sigma$. Once we have these structural objects, any structure $(D, \mathcal{I}^{\mathcal{C}}) \in \mathcal{C}$ gives a FOCTS with states $\mathcal{I}$ where $\mathcal{I}(\Sigma^{\mathcal{C}}) = \mathcal{I}^{\mathcal{C}}$, initial states $\mathcal{I}$ where $\mathcal{I} \models \varphi_0$, transitions $(\mathcal{I}_1, \mathcal{I}_2)$ where $\mathcal{I}_1 \oplus \mathcal{I}'_2 \models \tau$, and bad states $\mathcal{I}$ for which $\mathcal{I} \models Bad$.

3 First-Order Protocols

We introduce the notion of a *First-Order Protocol (FOP)* to simplify and restrict specifications in a FOTS. We choose restrictions to make our protocols asynchronous compositions of processes over static network topologies. Each process description is relative to its process neighbourhood. For example, a process operating on a ring has access to its immediate left and right neighbours, and transitions are restricted to these processes. This simplifies the modelling.

We begin with formalizing the concept of a network topology. As a running example, consider a Red-Black-Ring (RBR) topology, whose instance with 4 processes is shown in Fig. 2a. Processes are connected in a ring of alternating Red and Black processes. Each process is connected to two neighbours using two links, labelled *left* and *right*, respectively. From the example it is clear how to extend this topology to rings of arbitrary (even) size.

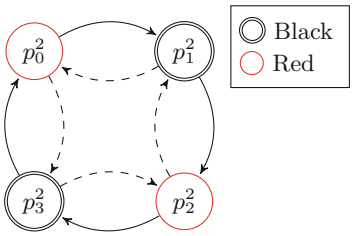
To formalize this, we assume that there is a unique sort *Proc* for processes. Define $\Sigma^{\mathcal{C}} = \Sigma_E^{\mathcal{C}} \uplus \Sigma_T^{\mathcal{C}}$ to be a *topological signature*, where $\Sigma_E^{\mathcal{C}}$ is a set of unary *Proc*-sorted functions and $\Sigma_T^{\mathcal{C}}$ is a set of distinct k-ary *Proc*-sorted predicates. Functions in $\Sigma_E^{\mathcal{C}}$ correspond to communication edges, such as *left* and *right* in our example. Predicates in $\Sigma_T^{\mathcal{C}}$ correspond to classes of processes, such as *Red* and *Black* in our example. For simplicity, we assume that all classes have the same arity k. We often omit k from the signature when it is contextually clear. We are now ready to define the concept of a network topology:

Definition 1. *A* network topology $\mathcal{C}$ *over a topological signature* $\Sigma^{\mathcal{C}}$ *is a collection of directed graphs* $G = (V, E)$ *augmented with an edge labelling* $dir : E \to \Sigma_E^{\mathcal{C}}$ *and* k*-node labelling* $kind : V^k \to \Sigma_T^{\mathcal{C}}$*. Given a node* p *in a graph* $G = (V, E)$ *from a network topology* $\mathcal{C}$*, the neighbourhood of* p *is defined as* $nbd(p) = \{p\} \cup \{q \mid (p, q) \in E\}$*, and a neighbourhood of a tuple* $\boldsymbol{p} = (p_1, \ldots, p_k)$ *is defined as* $nbd(\boldsymbol{p}) = \bigcup_{i=1}^{k} nbd(p_i)$*. A network topology is* deterministic *if for*

every distinct pair $q, r \in nbd(p) \setminus \{p\}$, $dir(p, q) \neq dir(p, r)$. *That is, each neighbour of* p *corresponds to a distinct name in* Σ_E.

Given a deterministic network topology $\Sigma_T^{\mathcal{C}} \cup \Sigma_E^{\mathcal{C}}$, the intended interpretation of a predicate $P \in \Sigma_T^{\mathcal{C}}$ is the set of all nodes in the network topology labelled by P, and the intended interpretation of a function $f \in \Sigma_E^{\mathcal{C}}$ is such that $f(p) = q$ if an edge (p, q) is labelled by f and $f(p) = p$, otherwise.

Each graph G in a network topology $\mathcal{C}$ provides a possible intended interpretation for the sort of processes *Proc*, and the edge and node labelling provide the intended interpretation for predicates and functions in $\Sigma^{\mathcal{C}}$.



(a) Red-Black-Ring of 4 process. Dashed arrows are *right*, and solid are *left*.

$$
\begin{aligned}
Init: &\quad var := null \\
Tr: &\quad black \Rightarrow right.var := r \\
&\quad red \Rightarrow right.var := b \\
Bad: &\quad red \wedge var = b
\end{aligned}
$$

(b) A simple protocol over Red-Black-Ring topology.

Fig. 2. An example of a topology and a protocol. (Color figure online)

Example 2. For our running example, consider the protocol informally shown in Fig. 2b described by a set of guarded commands. The protocol is intended to be executed on the RBR topology shown in Fig. 2a. Initially, all processes start with their state variable *var* set to a special constant *null*. Then, at each step, a non-deterministically chosen process, sends a color to its right. Every black process sends a red color r, and every red process sends a black color b. It is bad if a Red process ever gets a black color.

To formalize the topology, for each $n > 1$, let $G_n = (V_n, E_n)$, where $V_n = \{p_i^n \mid 0 \leq i < 2n\}$, and $E_n = \{(p_i^n, p_j^n) \mid |i - j| \bmod 2n = 1\}$. The edge labelling is given by $dir(p_i^n, p_j^n) = right$ if $j = (i+1) \mod n$ and *left* if $j = (i-1) \mod n$. Processes have colour $kind(p_i^n) = Red$ if i is even, and *Black* if i is odd. Finally, we define $\mathcal{RBR} = \{G_n \mid n \geq 2\}$ as the class of Red-Black Rings (RBR). □

Note that any set of graphs $\mathcal{G}$ with an upper bound on the out-degree of any vertex can be given a finite labelling according to the above definition.

First-Order Protocols. Once we have specified the topology, we want to establish how processes transition. We define the syntax and semantics of a protocol.

A protocol signature Σ is a disjoint union of a topological signature $\Sigma^{\mathcal{C}}$, a state signature Σ_S, and a background signature Σ_B. Recall that all functions and relations in $\Sigma^{\mathcal{C}}$ are of sort *Proc*. All elements of Σ_S have arity of at least 1 with the first and only the first argument of sort *Proc*. Elements of Σ_B do not

allow arguments of sort *Proc* at all. Intuitively, elements of $\Sigma^{\mathcal{C}}$ describe how processes are connected, elements of Σ_S describe the current state of some process, and elements of Σ_B provide background theories, such as laws of arithmetic or uninterpreted functions.

For an interpretation $\mathcal{I}$, and a set of processes $P \subseteq \mathcal{I}(Proc)$, we write $\mathcal{I}(\Sigma_S)(P)$ for the interpretation $\mathcal{I}(\Sigma_S)$ restricted to processes in P. Intuitively, we look only at the states of P and ignore the states of all other processes.

Definition 2. *A* First-Order Protocol *(FO-protocol) is a tuple* $P = (\Sigma, Init(\boldsymbol{p}), Mod(p), TrLoc(p), \mathcal{C})$*, where* Σ *is a protocol signature;* $Init(\boldsymbol{p})$ *is a formula with* k *free variables* $\boldsymbol{p}$ *of sort Proc;* $Mod(p)$ *is a set of terms* $\{t(p) \mid t \in dir(E)\} \cup \{p\}$*;* $TrLoc(p)$ *is a formula over the signature* $\Sigma \cup \Sigma'$ *with free process variable* p*, and* $\mathcal{C}$ *is a network topology. Furthermore,* $Init(\boldsymbol{p})$ *is of the form* $\bigwedge_{P \in \Sigma_T^{\mathcal{C}}} (P(\boldsymbol{p}) \Rightarrow Init_P(\boldsymbol{p}))$*, and each* $Init_P$ *is a formula over* $\Sigma \setminus \Sigma_{\mathcal{C}}$ *(an initial state described without reference to topology for each relevant topological class); and terms of sort Proc occurring in* $TrLoc(p)$ *are a subset of* $Mod(p)$*.*

Note that the *semantic* local neighbourhood $nbd(p)$ and the set of *syntactic* terms in $Mod(p)$ have been connected. Namely, for every edge $(p, q) \in E$, there is a term $t(p) \in Mod(p)$ to refer to q, and for every term $t(p) \in Mod(p)$, we will refer to some process in the neighbourhood of p.

$$
\begin{aligned}
Const &= \{null_{/0}, r_{/0}, b_{/0}\} \qquad Func = \{left_{/1}, right_{/1}, var_{/1}\} \\
Pred &= \{Red_{/1}, Black_{/1}, =_{/2}\} \qquad \Sigma = (Const, Func, Pred) \\
Init(p) &= (Red(p) \Rightarrow var(p) = null) \wedge (Black(p) \Rightarrow var(p) = null) \\
Mod(p) &= \{p, right(p), left(p)\} \\
t_r(p) &= var'(right(p)) = b \wedge var'(p) = var(p) \wedge var'(left(p)) = var(left(p)) \\
t_b(p) &= var'(right(p)) = r \wedge var'(p) = var(p) \wedge var'(left(p)) = var(left(p)) \\
TrLoc(p) &= (Red(p) \Rightarrow t_r(p)) \wedge (Black(p) \Rightarrow t_b(p))
\end{aligned}
$$

Fig. 3. A FO-protocol description of the system from Fig. 2.

$$
\varphi_0 \triangleq \forall \boldsymbol{p} \cdot Init(\boldsymbol{p}) \quad \tau \triangleq \exists p \cdot TrLoc(p) \wedge Frame(p)
$$

$$
Frame(p) \triangleq UnMod \wedge (\forall y \cdot y \notin Mod(p) \Rightarrow Unch(y)))
$$

$$
Unch(y) \triangleq \left(\bigwedge_{P \in Pred_S} \forall \boldsymbol{v} \cdot P(y, \boldsymbol{v}) \iff P'(y, \boldsymbol{v}) \right) \wedge \left(\bigwedge_{f \in Func_S} \forall \boldsymbol{v} \cdot f(y, \boldsymbol{v}) = f'(y, \boldsymbol{v}) \right)
$$

$$
UnMod \triangleq \left(\bigwedge_{P \in Pred_B} \forall \boldsymbol{v} \cdot P(\boldsymbol{v}) \iff P'(\boldsymbol{v}) \right) \wedge \left(\bigwedge_{f \in Func_B} \forall \boldsymbol{v} \cdot f(\boldsymbol{v}) = f'(\boldsymbol{v}) \right)
$$

Fig. 4. An FOTS of the protocol in Fig. 3.

A formal description of our running example is given in Fig. 3 as a FO-protocol. We define the signature including $\Sigma^{\mathcal{C}} = \{left, right, Red, Black\}$, the initial states $Init(p)$ in the restricted form, and modification set $Mod(p)$, where we allow processes to only write to their local neighbourhood. Next we specify two kinds of transitions, a red t_r and a black t_b transition. Each writes to their right neighbour the colour they expect that process to be. Each process p does not change the *var* states of $p, left(p) \in Mod(p)$. Finally, we specify our local transitions $TrLoc(p)$ by allowing each of the sub-transitions. Note that all process-sorted terms in $TrLoc(p)$ are in $Mod(p) = \{left(p), p, right(p)\}$, and we are allowed to call on topological predicates in $TrLoc$, finishing our specification.

The semantics of a protocol P are given be a FOCTSS as shown in Fig. 4. The protocol signature Σ is the same in the FOCTSS as in the FOP. Initially, φ_0 requires that all k-tuples of a given topology satisfy a topology-specific initial state. Second, to take a transition τ, some process takes a local transition $TrLoc(p)$ modifying states of processes that can be described using the terms in $Mod(p)$. $Frame(p), Unch(y)$ guarantee that the transition does not affect local state of processes that are outside of $Mod(p)$. Finally, $UnMod$ makes all functions and predicates in the background signature retain their interpretation during the transition. Overall, this describes a general multiprocess asynchronous protocol.

This definition of a FO-protocol places some added structure on the notion of FOTSS. It restricts how transition systems can be specified, which might seem like a drawback. On the contrary, the added structure provides two benefits. First, it removes the need for axiomatizing the network topology, since the topology is given semantically by $\mathcal{C}$. Second, the system guarantees that we model asynchronous composition of processes with local transitions – a common framework for specifying and reasoning about protocols.

To show safety of such a system, we will be concerned with invariants which only discuss a few processes, say $Inv(\boldsymbol{p})$ where $\boldsymbol{p} = p_1, \ldots, p_k$. Then our FO-invariants will be of the form $\forall \boldsymbol{p} \cdot Inv(\boldsymbol{p})$, and substituting φ_0 into our background, we find a natural check for when a given formula is inductive:

$$InvOk(Inv) \triangleq ((\forall \boldsymbol{p} \cdot Init(\boldsymbol{p})) \Rightarrow (\forall \boldsymbol{p} \cdot Inv(\boldsymbol{p}))) \wedge ((\forall \boldsymbol{p} \cdot Inv(\boldsymbol{p})) \wedge \tau \Rightarrow (\forall \boldsymbol{p} \cdot Inv'(\boldsymbol{p})))$$

Indeed, by unpacking definitions, one sees that $\models_{\mathcal{C}} InvOk$ means that every state on any trace of a FOCTS satisfies $\forall \boldsymbol{p} \cdot Inv(\boldsymbol{p})$, and thus it suffices to check that $\models_{\mathcal{C}} \forall \boldsymbol{p} \cdot Inv(\boldsymbol{p}) \Rightarrow \neg Bad$ to prove safety. We, however, will focus on the task of verifying a candidate formula as inductive or not.

To decide if a candidate is inductive or not requires reasoning in $FOL^{\mathcal{C}}$. However, reasoning about FOL extended with an arbitrary topology is difficult (or undecidable in general). We would like to reduce the verification problem to pure FOL. One solution is to axiomatize the topology in FOL – this is the approach taken by Ivy [16]. Another approach is to use properties of the topology to reduce reasoning about FO-protocols to FOL. This is similar to the use of topology to reduce reasoning about parameterized finite-state systems to reasoning about finite combinations of finite-state systems in [12]. In the next section, we show how this approach can be extended to FO-protocols.

4 Verifying FO-Protocols Using First Order Logic

In this section, we present a technique for reducing verification of FO-protocols over a given topology $\mathcal{C}$ to a decision problem in pure FOL. We assume that we are given a (modular) inductive invariant $\forall \boldsymbol{q} \cdot \mathit{Inv}(\boldsymbol{q})$ of the form $\left(\forall \boldsymbol{q} \cdot \bigwedge_{\mathit{Top} \in \Sigma_T^{\mathcal{C}}} \mathit{Top}(\boldsymbol{q}) \Rightarrow \mathit{Inv}_{\mathit{Top}}(\boldsymbol{q})\right)$. That is, Inv has a local inductive invariant $\mathit{Inv}_{\mathit{Top}(\boldsymbol{q})}$ for each topological class Top.

Given a First-Order Protocol and candidate invariant, we want to know if $\models_{\mathcal{C}} \mathit{InvOk}$. But deciding this is hard, and so we show that deciding validity of InvOk can be done in pure FOL using modular verification conditions in the style of Owicki-Gries [14] and Paramaterized Compositional Model Checking [12].

The input to our procedure is a formula $\mathit{Inv}_{\mathit{Top}}$ over signature $\Sigma_B \uplus \Sigma_S$ for each topological class $\mathit{Top} \in \Sigma_T^{\mathcal{C}}$. The VC is a conjunction of sentences ensuring that for each tuple of processes $\boldsymbol{q}$ in a topological class Top, $\mathit{Inv}_{\mathit{Top}}(\boldsymbol{q})$ is true initially, is stable under a transition of one process in $\boldsymbol{q}$, and is stable under interference by any other process p whose execution might affect some $q_i \in \boldsymbol{q}$. If the VC is FOL-valid, an inductive invariant has been found. If not, there will be a local violation to inductiveness, which may correspond to a global violation.

Formally, $\mathit{VC}(\mathit{Inv})$ is a conjunction of statements of the following two forms:

$$\forall \boldsymbol{q} \cdot (\mathit{CrossInit}_{\mathit{Top}}(\boldsymbol{q}) \Rightarrow \mathit{Inv}_{\mathit{Top}}(\boldsymbol{q})) \tag{1}$$

$$\forall p, \boldsymbol{q} \cdot ((\mathit{CrossInv}_{\mathit{Top}}(\mathit{Mod}(p), \boldsymbol{q}) \wedge \tau) \Rightarrow \mathit{Inv}'_{\mathit{Top}}(\boldsymbol{q})) \tag{2}$$

Statements of form (1) require that every local neighbourhood of $\boldsymbol{q}$ that satisfies all appropriate initial states also satisfies $\boldsymbol{q}$'s invariant. Statements of form (2) capture both transitions where $p = q_i$ for some i, or process p acts and modifies $q_i \in \mathit{nbd}(p)$, since p is quantified universally. All that remains is to formally construct the statements $\mathit{CrossInit}$, $\mathit{CrossInv}$. In order to do so, we construct a local characteristic formula $\chi(A, \boldsymbol{q})$ of a process $\boldsymbol{q}$ and neighbourhood A. Intuitively, we aim for $\chi(A, \boldsymbol{q})$ to encode the available local neighbourhoods of processes in A and $\boldsymbol{q}$ in $\mathcal{C}$.

Let $\chi_{\mathit{Top}}(A, \boldsymbol{q})$ be the strongest formula that satisfies $\models_{\mathcal{C}} \forall \boldsymbol{q} \cdot \mathit{Top}(\boldsymbol{q}) \Rightarrow \chi_{\mathit{Top}}(A, \boldsymbol{q})$, subject to the following syntactic restrictions. A formula is a candidate for $\chi_{\mathit{Top}}(A, \boldsymbol{q})$ when it is (1) over signature $\Sigma_T^{\mathcal{C}} \cup \Sigma_E^{\mathcal{C}} \cup \{=\}$, (2) contains only terms $A \cup \{q_i \mid q_i \in \boldsymbol{q}\}$, and (3) is in CNF and all literals from $\Sigma_T^{\mathcal{C}}$ appear in positive form. The syntactic restrictions are to capture when elements of $A, \boldsymbol{q}$ satisfy various topological notions given by signature $\Sigma_E^{\mathcal{C}} \cup \{=\}$. We also never force some processes to be outside of some topological class. Intuitively, χ is a formula that captures all topological knowledge derivable from the topology given that we know that $\mathit{Top}(\boldsymbol{q})$ holds. For instance, in $\mathcal{RBR}$, we have $\chi_{\mathit{Red}}(\emptyset, q) = \mathit{Red}(q)$, while expanding this for $A = \{\mathit{left}(p), p, \mathit{right}(p)\}$ results in the following formula. We drop some trivial statements. For instance, $\mathit{left}, \mathit{right}$ are inverse functions.

$$\chi_{Red}(\{left(p), p, right(p)\}, q) = Red(q) \wedge distinct(left(p), p, right(p)) \wedge$$
$$((Red(left(p)) \wedge Black(p) \wedge Red(right(p)) \wedge p \neq q) \vee$$
$$(Black(left(p)) \wedge Red(p) \wedge Black(right(p)) \wedge distinct(left(p), right(p), q)))$$

These characteristics are illustrated in Fig. 5. When we just look at $\chi_{Red}(\emptyset, q)$, we find q is red. However, if we expand our local reasoning to the characteristic $\chi_{Red}(Mod(p), q)$, we find that there are two options given by $\mathcal{RBR}$. One option is p is red, and $q = p$ is optional (dotted lines), while $q \neq left(p), right(p)$. Alternatively, p is black, and $q \neq p$, but q could be $left(p), right(p)$, or neither.

Once we have $\chi_{Top}(A, \boldsymbol{q})$, we can define our statements $CrossInit_{Top}$, $CrossInv_{Top}$. First, $CrossInit_{Top}(\boldsymbol{q})$ is obtained from $\chi_{Top}(\emptyset, \boldsymbol{q})$ by replacing every instance of $Top_i(\boldsymbol{q})$ with $Init_{Top_i}(\boldsymbol{q})$. We build our interference constraints in a similar way. We construct $CrossInv_{Top}(\boldsymbol{q})$ by modifying $\chi_{Top}(Mod(p), \boldsymbol{q})$. Namely, we obtain $CrossInv_{Top}(Mod(p), \boldsymbol{q})$ from $\chi_{Top}(Mod(p), \boldsymbol{q})$ by replacing every instance of $Top_i(\boldsymbol{q})$ with $Top_i(\boldsymbol{q}) \wedge Inv_{Top_i}(\boldsymbol{q})$.

Example 3. The VC generated by the $\mathcal{RBR}$ topology may be partitioned into VC_{Red} and VC_{Black}, each consisting of the statements whose conclusions are Inv_{Red}, Inv'_{Red} and Inv_{black}, Inv'_{black}, respectively. VC_{Red} is shown in Fig. 6. The conditions for VC_{Black} are symmetric. One can check that

$$Inv_{red}(p) \triangleq var(p) \neq b \qquad\qquad Inv_{black}(p) \triangleq \top$$

is an inductive invariant for the protocol in Fig. 2. □

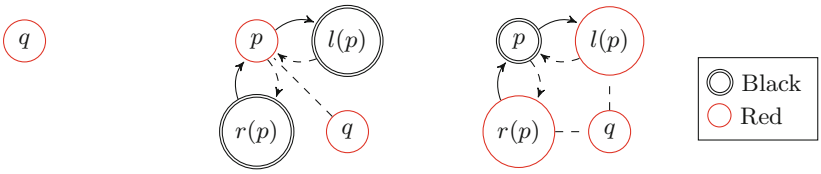


Fig. 5. Characteristics $\chi_{Red}(\emptyset, q)$ and $\chi_{Red}(Mod(p), q)$ for the $\mathcal{RBR}$ topology. (Color figure online)

$$\forall p \cdot Init_{red}(p) \Rightarrow Inv_{red}(p) \tag{3}$$

$$\forall p, q \cdot (Red(q) \wedge Inv_{red}(q) \wedge Red(left(p)) \wedge Inv_{red}(left(p)) \wedge$$
$$Black(p) \wedge Inv_{black}(p) \wedge Red(right(p)) \wedge Inv_{red}(right(p)) \wedge$$
$$p \neq q \wedge distinct(left(p), p, right(p))) \Rightarrow Inv'_{red}(q) \tag{4}$$

$$\forall p, q \cdot (Red(q) \wedge Inv_{red}(q) \wedge Black(left(p)) \wedge Inv_{black}(left(p)) \wedge$$
$$Red(p) \wedge Inv_{red}(p) \wedge Black(right(p)) \wedge Inv_{black}(right(p)) \wedge$$
$$distinct(left(p), right(p), q) \wedge distinct(left(p), p, right(p))) \Rightarrow Inv'_{red}(q) \tag{5}$$

$$VC_{P,1}(Inv_{red}, Inv_{black}) \triangleq (3) \wedge (4) \wedge (5) \tag{6}$$

Fig. 6. The verification conditions VC_{Red} for the red process invariant.

In practice, the role of the χ-computing oracle can be filled by a verification engineer. A description of local neighbourhoods starts by allowing all possible neighbourhoods. Then, a verifier may dismiss local configurations that cannot occur on the topology as they occur.

5 Soundness and Completeness

In this section, we present soundness and relative completeness of our verification procedure from Sect. 4.

Soundness. To show soundness, we present a model-theoretic argument to show that whenever the verification condition from Sect. 4 is valid in FOL, then the condition *InvOk* is valid in FOL extended with the given topology $\mathcal{C}$.

Theorem 1. *Given a FO-protocol P and a local invariant per topological class $Inv_{Top_1}(\boldsymbol{p}), \ldots, Inv_{Top_n}(\boldsymbol{p})$, if $\models VC(Inv)$, then $\models_{\mathcal{C}} InvOk(Inv)$.*

Proof. Assume $\models VC(Inv)$. We show that $InvOk(Inv)$ is valid in $FOL^{\mathcal{C}}$ by showing that any pair of $FOL^{\mathcal{C}}$ interpretations $\mathcal{I}$ and $\mathcal{I}'$ satisfy $VC(Inv)$ as FOL interpretations, and this is strong enough to guarantee $\mathcal{I} \oplus \mathcal{I}' \models InvOk(Inv)$.

Let $\mathcal{I}, \mathcal{I}'$ be $FOL^{\mathcal{C}}$ interpretations over some $G = (V, E) \in \mathcal{C}$. Then $\mathcal{I} \oplus \mathcal{I}' \models VC(Inv)$ because $VC(Inv)$ is valid and $\mathcal{I} \oplus \mathcal{I}'$ is an FOL interpretation.

We first show that $\mathcal{I} \models (\forall \boldsymbol{p} \cdot Init(\boldsymbol{p}) \Rightarrow \forall \boldsymbol{p} \cdot Inv(\boldsymbol{p}))$. Suppose that $\mathcal{I} \models \forall \boldsymbol{p} \cdot Init(\boldsymbol{p})$. Let $\boldsymbol{p}$ be an arbitrary tuple in G. If $\mathcal{I} \models \neg Top_i(\boldsymbol{p})$ for every $Top_i \in \Sigma_T$, then $Inv(\boldsymbol{p})$ follows vacuously. Otherwise, suppose $\mathcal{I} \models Top_i(\boldsymbol{p})$. Then by definition of χ, we obtain $\mathcal{I} \models \chi_{Top_i}(\emptyset, \boldsymbol{p})$ since $\mathcal{I} \models Top_i(\boldsymbol{p}) \Rightarrow \chi_{Top_i}(\emptyset, \boldsymbol{p})$. Since $\mathcal{I} \models \forall \boldsymbol{p} \cdot Init(\boldsymbol{p})$, this gives us that $\mathcal{I} \models CrossInit(\boldsymbol{p})$ (for any $Top_j(\boldsymbol{p}')$ in $\chi_{Top_i}(\emptyset, \boldsymbol{p})$, find that $Init(\boldsymbol{p}')$, and thus $Top_j(\boldsymbol{p}')$ implies $Init_{Top_j}(\boldsymbol{p}')$, giving $CrossInit$). Since $\mathcal{I} \models CrossInit_{Top_i}(\boldsymbol{p})$ and $\mathcal{I} \models VC$, we get $\mathcal{I} \models CrossInit_{Top_i}(\boldsymbol{p}) \Rightarrow Inv_{Top_i}(\boldsymbol{p})$, finally giving us $\mathcal{I} \models Inv_{Top_i}(\boldsymbol{p})$, as desired.

Second, we show that $\mathcal{I} \oplus \mathcal{I}' \models (\forall \boldsymbol{p} \cdot Inv(\boldsymbol{p})) \wedge \tau \Rightarrow (\forall \boldsymbol{p} \cdot Inv(\boldsymbol{p}))$. Suppose that $\mathcal{I} \models \forall \boldsymbol{p} \cdot Inv(\boldsymbol{p})$ and $\mathcal{I} \oplus \mathcal{I}' \models TrLoc(p) \wedge Frame(p)$ for some $p \in V$. We show that $\mathcal{I}' \models \forall \boldsymbol{q} \cdot Inv'(\boldsymbol{q})$. Let $\boldsymbol{q} \in V^k$ be an arbitrary process tuple. If $\mathcal{I}' \not\models Top_i(\boldsymbol{q})$ for all $1 \leq i \leq n$, then $\mathcal{I}' \models Inv'(\boldsymbol{q})$ vacuously. Suppose $\mathcal{I}' \models Top_i(\boldsymbol{q})$ for some $Top_i \in \Sigma_T$. Then $\mathcal{I} \models Top_i(\boldsymbol{q}) \Rightarrow \chi_{Top_i}(Mod(p), \boldsymbol{q})$, and so $\mathcal{I} \models \chi_{Top_i}(Mod(p), \boldsymbol{q})$. Again by instantiating $\forall \boldsymbol{p} \cdot Inv(\boldsymbol{p})$ on terms in $Mod(p), \boldsymbol{q}$, we may obtain that $\mathcal{I} \models CrossInv(Mod(p), \boldsymbol{q})$. Combined, we have $\mathcal{I} \oplus \mathcal{I}' \models CrossInv(Mod(p), \boldsymbol{q}) \wedge \tau$. Applying VC finally gives $Inv_{Top_i}(\boldsymbol{q})$. Thus both conjuncts of $InvOk(Inv)$ are satisfied, giving our result. □

Intuitively, the correctness of Theorem 1 follows from the fact that any interpretation under $FOL^{\mathcal{C}}$ is also an interpretation under FOL, and all preconditions generated for VC are true under $FOL^{\mathcal{C}}$ interpretation.

Small Model Property. Checking validity of universally quantified statements in FOL is in the fragment EPR, and thus we obtain a result saying that we only need to consider models of a given size. This means that a FOL solver needs to only reason about finitely many elements of sort *Proc*. It further means that topologies such as $\mathcal{RBR}$ may be difficult to compile to EPR in Ivy, but our methodology guarantees our verifications will be in EPR.

Theorem 2. *If* $\models VC(Inv)$ *for all process domains of size at most* $|Mod(p)|+k$, *then* $\models_{\mathcal{C}} InvOk(Inv)$.

Proof. By contrapositive, suppose $\not\models_{\mathcal{C}} InvOk(Inv)$. Then, by Theorem 1, $\not\models VC(Inv)$. Let $\mathcal{I} \oplus \mathcal{I}'$ be a falsifying interpretation. It contains an assignment to $Mod(p)$ and $\boldsymbol{q}$, or to $\boldsymbol{p}$ that makes at least one statement in $VC(Inv)$ false. Then $\mathcal{I} \oplus \mathcal{I}'(Mod(p) \cup \boldsymbol{q})$ or $\mathcal{I}(\boldsymbol{p})$ is also a counter-model to $VC(Inv)$, but with at most $|Mod(p)| + k$ elements of sort *Proc*.

Relative Completeness. We show that our method is relatively complete for local invariants that satisfy the *completability* condition. Let $\varphi(\boldsymbol{p})$ be a formula of the form $\bigwedge_{i=1}^{n}(Top_i(\boldsymbol{p}) \Rightarrow \varphi_{Top_i}(\boldsymbol{p}))$ with $\varphi_{Top_i}(\boldsymbol{p})$ over the signature $\Sigma_S \cup \Sigma_B$. Intuitively, $\varphi(\boldsymbol{p})$ is *completable* if every interpretation $\mathcal{I}$ that satisfies $\forall \boldsymbol{p} \cdot \varphi(\boldsymbol{p})$ and is consistent with some $\mathcal{C}$-interpretation $\mathcal{I}_G$ can be extended to a full $\mathcal{C}$-interpretation (not necessarily $\mathcal{I}_G$) that satisfies $\forall \boldsymbol{p} \cdot \varphi(\boldsymbol{p})$. Formally, φ is *completable* relative to topology $\mathcal{C}$ iff for every interpretation $\mathcal{I}$ with domain $U \subseteq V$ for $G = (V, E) \in \mathcal{C}$ with an intended interpretation $\mathcal{I}_G$ such that $(\mathcal{I} \uplus \mathcal{I}_G)(U) \models \forall \boldsymbol{p} \cdot \varphi(\boldsymbol{p})$, there exists an interpretation $\mathcal{J}$ with domain V s.t. $(\mathcal{J} \uplus \mathcal{I}_G) \models \forall \boldsymbol{p} \cdot \varphi$ and $\mathcal{I}(U) = \mathcal{J}(U)$. In addition to relative completeness, we need a lemma for when a FOL interpretation can be lifted to a $\mathcal{C}$ interpretation.

Lemma 1. *If FOL interpretation* $\mathcal{I}$ *of signature* $\Sigma^{\mathcal{C}}$ *satisfies* $\mathcal{I} \models \chi_{Top}(A, \boldsymbol{q})$, *then there exists a* $\mathcal{C}$ *interpretation* $\mathcal{J}$ *of the same signature with* $\mathcal{J} \models \chi_{Top}(A, \boldsymbol{q})$ *and* $\mathcal{I} \models t_i = t_j$ *iff* $\mathcal{J} \models t_i = t_j$ *for terms* $t_i, t_j \in A \cup \boldsymbol{q}$.

Proof. Let $\mathcal{I} \models \chi_{Top}(A, \boldsymbol{q})$. Let $\varphi(A, \boldsymbol{q})$ be the conjunction of all atomic formulae over the signature $\{=\}$ and statements $\neg Top_j(\boldsymbol{q}')$ that is true of elements of $A, \boldsymbol{q}$ in interpretation $\mathcal{I}$. If no $\mathcal{C}$ interpretation $\mathcal{J} \models Top(\boldsymbol{q}) \wedge \varphi(A, \boldsymbol{q})$, then we can add the clause $\neg\varphi(A, \boldsymbol{q})$ to $\chi_{Top}(A, \boldsymbol{q})$, thus strengthening it (this is stronger since $\mathcal{I} \models Top(\boldsymbol{q}), \not\models \neg\varphi(A, \boldsymbol{q})$, and is true of every interpretation modelling $Top(\boldsymbol{q})$). However, this violates the assumptions that χ_{Top} is as strong as possible. Thus, some $\mathcal{J} \models Top(\boldsymbol{q}) \wedge \varphi(A, \boldsymbol{q})$. Note that $\mathcal{J}$ already satisfies $t_i = t_j$ iff $\mathcal{I}$ satisfies $t_i = t_j$ since every statement of $=, \neq$ is included in $\varphi(A, \boldsymbol{q})$. Finally, since $\mathcal{J}$ is a $\mathcal{C}$ interpretation and $\mathcal{J} \models Top(\boldsymbol{q})$, then $\mathcal{J} \models \chi_{Top}(A, \boldsymbol{q})$ by definition. □

Theorem 3. *Given an FO-protocol* P, *if* $\models_{\mathcal{C}} InvOk(Inv)$ *and both* $Inv(\boldsymbol{p})$ *and* $Init(\boldsymbol{p})$ *are completable relative to* $\mathcal{C}$, *then* $\models VC(Inv)$.

Proof. By contra-positive, we show that given a completable local invariant $Inv(\boldsymbol{p})$, if $VC(Inv)$ is falsifiable in FOL, then $InvOk(Inv)$ is falsifiable in $FOL^{\mathcal{C}}$.

Suppose $VC(Inv)$ is not valid, and let $\mathcal{I} \oplus \mathcal{I}'$ by such that $\mathcal{I} \oplus \mathcal{I}' \not\models VC(Inv)$. We consider two cases – a violation initially or inductively.

Case 1: Initialization: For some processes $\boldsymbol{p} = (p_1, \ldots, p_k)$ and $1 \leq i \leq |\Sigma_T^{\mathcal{C}}|$, $\mathcal{I} \models \mathit{CrossInit}_{\mathit{Top}_i}(\boldsymbol{p})$ and $\mathcal{I} \not\models \mathit{Inv}_{\mathit{Top}_i}(\boldsymbol{p})$. Modify $\mathcal{I}(\Sigma_T)$ for every $\boldsymbol{q}$ so that $\mathit{Top}_j(\boldsymbol{q})$ is interpreted to be true iff $\mathit{Init}_{\mathit{Top}_j}(\boldsymbol{q})$ is true. Noting that all initial conditions are outside of the signature $\Sigma_T^{\mathcal{C}}$, we observe that this is done without loss of generality. Since $\mathcal{I} \models \mathit{CrossInit}_{\mathit{Top}_i}(\boldsymbol{p})$, we conclude now that $\mathcal{I} \models \chi_{\mathit{Top}_i}(\emptyset, \boldsymbol{p})$. Applying Lemma 1 to $\mathcal{I}(\Sigma_{\mathcal{C}})$, we get a $\mathcal{C}$ interpretation $\mathcal{J} \models \chi_{\mathit{Top}_i}(\emptyset, \boldsymbol{p}^{\mathcal{C}})$. Since this model has the same equalities of terms $\boldsymbol{p}^{\mathcal{C}}$ in $\mathcal{J}$ as $\boldsymbol{p}$ in $\mathcal{I}$, we may copy the states $\mathcal{I}(\Sigma_S)(p_i)$ to $\mathcal{J}(\Sigma_S)(p_i^{\mathcal{C}})$. Set $\mathcal{J}(\Sigma_B) = \mathcal{I}(\Sigma_B)$. Since *Init* is completable by assumption, we complete $\mathcal{J}(\Sigma_S \cup \Sigma_B)(\boldsymbol{p})$ to $\mathcal{J}(\Sigma_S \cup \Sigma_B)$, completing our construction of $\mathcal{J}$ interpreting $\Sigma^{\mathcal{C}} \cup \Sigma_S \cup \Sigma_B$. Note that $\mathcal{J} \models \forall \boldsymbol{p} \cdot \mathit{Init}(\boldsymbol{p})$, but $\mathcal{J} \models \mathit{Top}_i(\boldsymbol{p}^{\mathcal{C}}) \wedge \neg \mathit{Inv}_{\mathit{Top}_i}(\boldsymbol{p}^{\mathcal{C}})$, thus showing that $\mathit{InvOk}(\mathit{Inv})$ is falsifiable in $\mathit{FOL}^{\mathcal{C}}$.

Case 2: Inductiveness: For some $p, \boldsymbol{q}$, and $1 \leq i \leq |\Sigma_T^{\mathcal{C}}|$, we have $\mathcal{I} \models \mathit{CrossInv}_{\mathit{Top}_i}(\mathit{Mod}(p), \boldsymbol{q})$, $(\mathcal{I} \oplus \mathcal{I}') \models \mathit{TrLoc}(p) \wedge \mathit{Frame}(p)$, and $\mathcal{I}' \not\models \mathit{Inv}_{\mathit{Top}_i}(\boldsymbol{q})$. By construction, $\models \mathit{CrossInv}(\mathit{Mod}(p), \boldsymbol{q}) \Rightarrow \chi_{\mathit{Top}_i}(\mathit{Mod}(p), \boldsymbol{q})$. Applying Lemma 1 to $\mathcal{I}(\Sigma_{\mathcal{C}}) \models \chi_{\mathit{Top}_i}(\mathit{Mod}(p), \boldsymbol{q})$, we get a $\mathcal{C}$ interpretation of $\Sigma_T^{\mathcal{C}}$, $\mathcal{J} \models \chi_{\mathit{Top}_i}(\mathit{Mod}(p^{\mathcal{C}}), \boldsymbol{q}^{\mathcal{C}})$. We extend this to a full model $\mathcal{J} \oplus \mathcal{J}'$ of signature $\Sigma^{\mathcal{C}} \cup \Sigma_S \cup \Sigma_B$, and its primed copy. We set $\mathcal{J}'(\Sigma_{\mathcal{C}}) = \mathcal{J}(\Sigma_{\mathcal{C}})$. Then, since $\mathcal{J}$ and $\mathcal{I}$, and $\mathcal{J}'$ and $\mathcal{I}'$ share equalities across terms in $\mathit{Mod}(p) \cup \boldsymbol{q}$ and $\mathit{Mod}(p^{\mathcal{C}}) \cup \boldsymbol{q}^{\mathcal{C}}$, we can lift states from terms $t \in \mathit{Mod}(p) \cup \boldsymbol{q}$ by $\mathcal{J}(\Sigma_S \cup \Sigma_B)(t^{\mathcal{C}}) \triangleq \mathcal{I}(\Sigma_S \cup \Sigma_B)(t)$ and $\mathcal{J}'(\Sigma_S)(t^{\mathcal{C}}) \triangleq \mathcal{I}'(\Sigma_S)(t)$. Since *Inv* is completable, we complete this interpretation with $\mathcal{J}(\Sigma_S \cup \Sigma_B)$ and clone the completion to $\mathcal{J}'(\Sigma_S \cup \Sigma_B)(V \setminus (\mathit{Mod}(p) \cup \boldsymbol{q}))$. Overall, this completes the interpretation $\mathcal{J} \oplus \mathcal{J}'$.

Note that $\mathcal{J} \models \forall \boldsymbol{p} \cdot \mathit{Inv}(\boldsymbol{p})$ by construction. Similarly, $\mathcal{J} \oplus \mathcal{J}' \models \tau$ since $\mathcal{I} \oplus \mathcal{I}' \models \tau(p)$ and $\mathit{Mod}(p)$ terms are lifted directly from $\mathcal{I}$ and $\mathcal{I}'$ to $\mathcal{J}$ and $\mathcal{J}'$. Finally, $\mathcal{J}' \models \neg \mathit{Inv}'_{\mathit{Top}_i}(\boldsymbol{q})$ since $\mathcal{J}'(\Sigma_S)$ is lifted directly from $\mathcal{I}'(\Sigma_S \cup \Sigma_B)$, which is the language of invariants. Thus, we have shown that $\mathit{InvOk}(\mathit{Inv})$ is falsifiable in $\mathit{FOL}^{\mathcal{C}}$ in this case as well. □

How restrictive is the requirement of *completability*? Intuitively, suppose a protocol is very restrictive about how processes interact. Then the system is likely sufficiently intricate that trying to reason locally may be difficult independent of our methodology. For instance, the invariant we later find for leader election is not completable. However, if equivalence classes are small, then most reasonable formulae satisfy the completability condition.

Theorem 4. *If $\mathit{Inv}_{\mathit{Top}_i}(p)$ is satisfiable over any domain for each $1 \leq i \leq n$ and topological predicates are of arity $k = 1$, then $\mathit{Inv}(p)$ is completable.*

Proof. Let $\mathit{Inv}_i(p)$ be satisfiable for each $1 \leq i \leq n$. Then let $\mathcal{I}(V')$ be an interpretation of $\Sigma_B \uplus \Sigma_S$ over domain $V' \subseteq V$ for $G = (V, E) \in \mathcal{C}$. For each $p \in V \setminus V'$, suppose $\mathcal{I}_G \models \mathit{Top}_i(p)$ for some $1 \leq i \leq n$. Then choose $\mathcal{J}(p) \models \mathit{Inv}_{\mathit{Top}_i}(p)$ since $\mathit{Inv}_{\mathit{Top}_i}(p)$ is satisfiable. Otherwise, if $\mathcal{I}_G \not\models \mathit{Top}_i(p)$ for all

$1 \leq i \leq n$, then $\mathcal{J}(p)$ is chosen arbitrarily. In either case, $\mathcal{J} \models Inv(p)$. Finally, define $\mathcal{J}(p) = \mathcal{I}(p)$ for $p \in V'$. Then $\mathcal{J}$ completes the partial interpretation $\mathcal{I}$.

Theorem 4 can be generalized to the case where the topological kinds Σ_T are non-overlapping, and individually completable, where by individually completable, we mean that if $Top(\boldsymbol{p})$ and process states of $\boldsymbol{p}' \subset \boldsymbol{p}$ are given, then there is a way to satisfy $Inv(\boldsymbol{p})$ without changing the states of $\boldsymbol{p}'$.

6 Example: Leader Election Protocol

In this section, we illustrate our approach by applying it to the well-known leader election protocol [3]. This is essentially the same protocol used to illustrate Ivy in [16]. The goal of the protocol is to choose a leader on a ring. Each process sends messages to its neighbour on one side and receives messages from a neighbour on the other side. Initially, all processes start with distinct identifiers, *id*, that are totally ordered. Processes pass *id*s around the ring and declare themselves the leader if they ever receive their own *id*.

We implement this behaviour by providing each process a comparison variable *comp*. Processes then pass the maximum between *id* and *comp* to the next process. A process whose *id* and *comp* have the same value is the leader. The desired safety property is that there is never more than one leader in the protocol.

In [16], the protocol is modelled by a global transition system. The system maintains a bag of messages for each process. At each step, a currently waiting message is selected and processed according to the program of the protocol (or a fresh message is generated). The network topology is axiomatized, as shown in Sect. 1. Here, we present a local model of the protocol and verify it locally.

Network Topology. The leader election protocol operates on a ring of size at least 3. For $n \geq 3$, let $G_n = (V_n, E_n)$, where $V_n = \{p_i^n \mid 0 \leq i < n\}$ and $E_n = \{(p_i^n, p_j^n) \mid 0 \leq i < n, j = i + 1 \bmod n\}$. Let $\Sigma_E = \{next\}$ and $\Sigma_T = \{btw\}$, where btw is a ternary relation such that $btw(p_i^n, p_j^n, p_k^n)$ iff $i < j < k$, $j < k < i$, or $k < i < j$. Finally, the network topology is $\mathcal{BTW} = \{G_n \mid n \geq 3\}$. Note that while $\mathcal{BTW}$ can be axiomatized in FOL, we do not require such an axiomatization. The definition is purely semantic, no theorem prover sees it.

$$
\begin{array}{l}
Const \triangleq \{0_{/0}\} \quad Func \triangleq \{next_{/1}, id_{/1}, comp_{/1}\} \quad Pred \triangleq \{\leq_{/2}, =_{/2}, btw_{/3}\} \quad \mathcal{C} \triangleq \mathcal{BTW} \\
\Sigma \triangleq (Const, Func, Pred) \quad LO_0(\leq) \triangleq LO(\leq) \wedge \forall x \cdot 0 \leq x \quad Mod(p) \triangleq \{p, next(p)\} \\
Init(p) \triangleq (LO_0(\leq) \wedge btw(x,y,z) \Rightarrow (distinct(id(x), id(y), id(z)) \wedge 0 < id(x) \wedge comp(x) = 0)) \\
\tau_1(p) \triangleq \big(id(p) \leq comp(p) \Rightarrow \big(comp'(next(p)) = comp(p)\big)\big) \\
\tau_2(p) \triangleq \big(comp(p) \leq id(p) \Rightarrow \big(comp'(next(p)) = id(p)\big)\big) \\
TrLoc(p) \triangleq \big(id(p) = id'(p) \wedge comp(p) = comp'(p) \wedge id'(next(p)) = id(next(p)) \wedge \tau_1(p) \wedge \tau_2(p)\big)
\end{array}
$$

Fig. 7. A model of the Leader Election protocol as a FO-protocol.

$$(btw(x,y,z) \wedge id(y) = comp(x)) \Rightarrow (id(z) \leq id(y))$$
$$(btw(x,y,z) \wedge id(x) = comp(x)) \Rightarrow (id(y) \leq id(x) \wedge id(z) \leq id(x))$$
$$(btw(x,y,z) \wedge id(x) = comp(x) \wedge id(y) = comp(y)) \Rightarrow x = y$$

Fig. 8. Local inductive invariant $Inv_{lead}(x,y,z)$ for Leader Election from Fig. 7.

A formal specification of the leader election as an FO-protocol is shown in Fig. 7, where $LO(\leq)$ is an axiomatization of total order from [16], and $x < y$ stands for $x \leq y \ \wedge \ x \neq y$. The model follows closely the informal description of the protocol given above. The safety property is $\neg Bad$, where $Bad = btw(x,y,z) \wedge id(x) = comp(x) \wedge id(y) = comp(y)$. That is, a bad state is reached when two processes that participate in the *btw* relation are both leaders.

A local invariant Inv_{lead} based on the invariant from [16] is shown in Fig. 8. The invariant first says if an *id* passes from y to x through z, then it must witness $id(y) \geq id(z)$ to do so. Second, the invariant says that if a process is a leader, then it has a maximum id. Finally, the invariant asserts our safety property.

This invariant was found interactively with Ivy by seeking local violations to the invariant. Our protocol's *btw* is uninterpreted, while Ivy's *btw* is explicitly axiomatized. The inductive check assumes that the processes $p, next(p), \boldsymbol{q}$ all satisfy a finite instantiation of the ring axioms (this could be done by the developer as needed if an axiomatization is unknown, and this is guaranteed to terminate as there are finitely many relevant terms), and $btw(\boldsymbol{q})$. Once the invariants are provided, the check of inductiveness is mechanical[1]. Overall, this presents a natural way to model protocols for engineers that reason locally.

An Uncompletable Invariant. The invariant for the leader election is not completable. To see this, we present a partial interpretation $\mathcal{I}$ over $\{p_0^3, p_2^3\} \subseteq V_3$ from G_3 with no extension. We choose $\mathcal{I}(\leq)$ to be $\leq$ over $\mathbb{N}$, as intended. Then we choose $\mathcal{I}(id)$ to map $p_0^3 \mapsto 1$ and $p_2^3 \mapsto 2$. We also choose $\mathcal{I}(comp)$ to map $p_0^3 \mapsto 0$ and $p_2^3 \mapsto 1$. Since no tuple satisfies *btw*, this vacuously satisfies all invariants thus far. Let $\mathcal{J}$ be a $\mathcal{BTW}$ interpretation agreeing on p_0^3, p_2^3. Consider $id(p_1^3)$. We know $id(p_1^3) \neq 0, 1, 2$ since we require distinct ids across the new *btw* relation. But we also have $id(p_0^3) = comp(p_2^3)$ and thus to satisfy *Inv* we must have $id(p_0^3) \geq id(p_1^3)$. Thus we seek an $n \in \mathbb{N}$ such that $1 \geq n$, but $n \neq 0, 1$, which cannot exist. Thus *Inv* is uncompletable.

7 Related Work

Finite-state parameterized verification is undecidable [2]. We have shown how analysis techniques for parametric distributed systems composed of components running on locally symmetric topologies, introduced in [8–10,12,13], can be generalized and applied within a First Order Logic based theorem proving engine.

[1] Ivy verifications for both examples, globally and locally, can be found at github.com/ashmorer/fopExamples.

We based our description of leader election on Ivy's [16]. However, the analysis carried out in Ivy [16] is global, while the analysis given in this paper is local, where the local structures reason about triples of processes in the ring.

There has been extensive work on proving properties of parametric, distributed protocols. In particular the work in [1] offers an alternative approach to parametric program analysis based on "views". In that work, cut off points are calculated during program analysis. As another example, in [8,12,13] the "cut-offs" are based on the program topology and the local structural symmetries amongst the nodes of the process interconnection networks.

The notion of a "cutoff" proof of safety for a parametric family of programs was first introduced by [5]. For example, in [5], if a ring of 3 processes satisfies a parametric property then the property must hold for all rings with at least three nodes. The technique used here is somewhat different; rather than needing to check a ring of 3 processes, we check all pseudo-rings of a given size.

Local symmetry reduction for multi-process networks and parametric families of networks generalizes work on "global" symmetry reduction introduced by [6] and [4]. Local symmetry is, in general, an abstraction technique that can offer exponentially more reduction than global symmetry. In particular, ring structures are globally rotationally symmetric, but for isomorphic processes may be fully-locally symmetric [12,13].

Recent work [18] has focused on *modular* reasoning in the proof or analysis of distributed systems. In the current work, the modularity in the proof is driven by a natural modularity in the program structures. In particular, for programs of several processes proofs are structured by modules that are local to a neighborhood of one or more processes [8,12,13].

8 Conclusion

We have presented a framework for specifying protocols in a process-local manner with topology factored out. We show that verification is reducible to FOL with an oracle to answer local questions about the topology. This reduction results in a decidable VC when the background theories are decidable. This cleanly separates the reasoning about the topology from that of the states of the processes.

Many open questions remain. We plan to investigate our methodology on other protocols and topologies, implement oracles for common topologies, and explore complexity of the generated characteristic formulae. Finally, we restricted ourselves to static topologies of bounded degree. Handling dynamic or unbounded topologies, for example in the AODV protocol [11], is left open.

Acknowledgements. The authors' research was supported, in part, by Individual Discovery Grants from the Natural Sciences and Engineering Research Council of Canada.

References

1. Abdulla, P., Haziza, F., Holík, L.: Parameterized verification through view abstraction. Int. J. Softw. Tools Technol. Transf. **18**(5), 495–516 (2016)
2. Apt, K.R., Kozen, D.C.: Limits for automatic verification of finite-state concurrent systems. Inf. Process. Lett. **22**(6), 307–309 (1986)
3. Chang, E., Roberts, R.: An improved algorithm for decentralized extrema-finding in circular configurations of processes. Commun. ACM **22**(5), 281–283 (1979)
4. Clarke, E.M., Enders, R., Filkorn, T., Jha, S.: Exploiting symmetry in temporal logic model checking. Form. Methods Syst. Des. **9**(1–2), 77–104 (1996)
5. Emerson, E.A., Namjoshi, K.S.: Reasoning about rings. In: Proceedings of the 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 1995, pp. 85–94. ACM, New York (1995)
6. Emerson, E.A., Sistla, A.P.: Symmetry and model checking. Form. Methods Syst. Des. **9**(1–2), 105–131 (1996)
7. Leino, K.R.M.: Dafny: an automatic program verifier for functional correctness. In: Clarke, E.M., Voronkov, A. (eds.) LPAR 2010. LNCS (LNAI), vol. 6355, pp. 348–370. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17511-4_20
8. Namjoshi, K.S., Trefler, R.J.: Local symmetry and compositional verification. In: Kuncak, V., Rybalchenko, A. (eds.) VMCAI 2012. LNCS, vol. 7148, pp. 348–362. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27940-9_23
9. Namjoshi, K.S., Trefler, R.J.: Uncovering symmetries in irregular process networks. In: Giacobazzi, R., Berdine, J., Mastroeni, I. (eds.) VMCAI 2013. LNCS, vol. 7737, pp. 496–514. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35873-9_29
10. Namjoshi, K.S., Trefler, R.J.: Analysis of dynamic process networks. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 164–178. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46681-0_11
11. Namjoshi, K.S., Trefler, R.J.: Loop freedom in AODVv2. In: Graf, S., Viswanathan, M. (eds.) FORTE 2015. LNCS, vol. 9039, pp. 98–112. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19195-9_7
12. Namjoshi, K.S., Trefler, R.J.: Parameterized compositional model checking. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 589–606. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_39
13. Namjoshi, K.S., Trefler, R.J.: Symmetry reduction for the local Mu-Calculus. In: Beyer, D., Huisman, M. (eds.) TACAS 2018. LNCS, vol. 10806, pp. 379–395. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-89963-3_22
14. Owicki, S.S., Gries, D.: Verifying properties of parallel programs: an axiomatic approach. Commun. ACM **19**(5), 279–285 (1976)
15. Padon, O., Hoenicke, J., Losa, G., Podelski, A., Sagiv, M., Shoham, S.: Reducing liveness to safety in first-order logic. PACMPL **2**(POPL), 26:1–26:33 (2018)
16. Padon, O., McMillan, K.L., Panda, A., Sagiv, M., Shoham, S.: Ivy: safety verification by interactive generalization. In: Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2016, Santa Barbara, CA, USA, 13–17 June 2016, pp. 614–630 (2016)

17. Piskac, R., de Moura, L.M., Bjørner, N.: Deciding effectively propositional logic using DPLL and substitution sets. J. Autom. Reason. **44**(4), 401–424 (2010)
18. Taube, M., et al.: Modularity for decidability of deductive verification with applications to distributed systems. In: Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2018, pp. 662–677. ACM, New York (2018)