Sound Verification Procedures for Temporal Properties of Infinite-State Systems

Quentin Peyras[1], Jean-Paul Bodeveix[2], Julien Brunel[1], and David Chemouil[1][0000-0003-4136-783X]

[1] ONERA DTIS, Université de Toulouse, France, first.last@onera.fr
[2] IRIT CNRS UPS, Université de Toulouse, France, first.last@irit.fr

Abstract. First-Order Linear Temporal Logic (FOLTL) is particularly convenient to specify distributed systems, in particular because of the unbounded aspect of their state space. We have recently exhibited novel decidable fragments of FOLTL which pave the way for tractable verification. However, these fragments are not expressive enough for realistic specifications. In this paper, we propose three transformations to translate a typical FOLTL specification into two of its decidable fragments. All three transformations are proved sound (the associated propositions are proved in Coq) and have a high degree of automation. To put these techniques into practice, we propose a specification language relying on FOLTL, as well as a prototype which performs the verification, relying on existing model checkers. This approach allows us to successfully verify safety and liveness properties for various specifications of distributed systems from the literature.

1 Introduction

Verifying properties of distributed protocols is a demanding endeavor. Several approaches have been proposed, ranging from verification frameworks, like IronFleet [12] or Verdi [27] to tool-supported languages like TLA$^+$ [17], Event-B [1] or Ivy [21, 20]. However, when systems of *arbitrary size* are considered, verifying properties usually requires some remarkable effort: inductive invariants must be sought and exhibited (possibly with tool support), and some manual proof effort may still be necessary. Worse, when *liveness* properties are checked, this effort becomes very substantial and tool support is still quite limited.

A natural setting for specification, in particular for safety and liveness properties of infinite-state systems, is (mono- and many-sorted) first-order linear temporal logic (FOLTL). However, it is highly undecidable [13, 14]. In recent work [23, 24], some of the present authors devised the "Geneva" fragments of FOLTL, which were shown to be decidable. More precisely, these fragments enjoy a "bounded domain property" (BDP), a form of computable finite model property over the first-order domains. Decidability is obtained by expanding first-order quantifiers over the domains (using the computed bounds) and then relying on (decidable) propositional-LTL satisfiability checking.

The Geneva fragments are rather expressive but still have limitations that thwart their use for the specification of systems. In particular, most forms of fairness assumptions, as well as frame conditions (which specify what does not change when a transition happens in a system), do not fit in the fragments. Furthermore, topological properties of systems (such as ring topologies) are hard or even impossible to specify.

In this article, we mitigate this deficiency by exhibiting three transformations that allow to map an *undecidable*, expressive fragment of $\text{FOLTL}^*_=$ (FOLTL with equality and reflexive-transitive closure, to characterize topological properties) into decidable fragments (akin to the Geneva ones), thus allowing the automatic verification of safety and liveness properties of infinite-state systems. Then we apply these techniques to the verification of properties of various protocols.

Notice that none of the proposed transformations is complete. It is actually impossible to devise complete transformations, even assuming a procedure that would be fed additional user input. This is because FOLTL is not even semi-decidable.[3]

In more detail, we make the following contributions (cf. Fig. 1):

- we define an undecidable, expressive specification language, called Cervino, the semantics of which is expressed in terms of $\text{FOLTL}^*_=$;
- we exhibit two fragments of many-sorted FOLTL that enjoy the BDP;
- we devise three abstraction transformations that map (the semantics of) Cervino into one of the said two fragments:
  - the first of these transformations (called TEA) is fully automatic while the other two (TTC and TFC) must be passed additional data (in the shape of peculiar formulas);
  - these three transformations, as well as other minor ones, are implemented as *tactics* in a prototype tool [22];
  - the associated theorems and lemmas are also formalized and proved correct, using Coq [22];
- we demonstrate our approach on several case studies that are often used as benchmarks in the literature.

This article is organized as follows: in Sect. 2, we illustrate our approach using an example (a leader election protocol). Sect. 3 introduces definitions as well as the two fragments used in the rest of the paper. In Sect. 4, we present basic techniques, which are used in some of our transformations. Then, in Sect. 5, we formalize the automatic TEA transformation. Sect. 6 and 7 present, respectively, the TFC and TTC transformations. In Sect. 8, we evaluate our approach on various protocols. Finally, we compare our results with related work in Sect. 9.

2 The Cervino Language

In this section, we present the Cervino modeling language informally. Its semantics, given in terms of many-sorted $\text{FOLTL}^*_=$ (FOLTL with equality and

[3] Indeed, having such a transformation would give a procedure for semi-decidability by testing all possible inputs on this transformation.

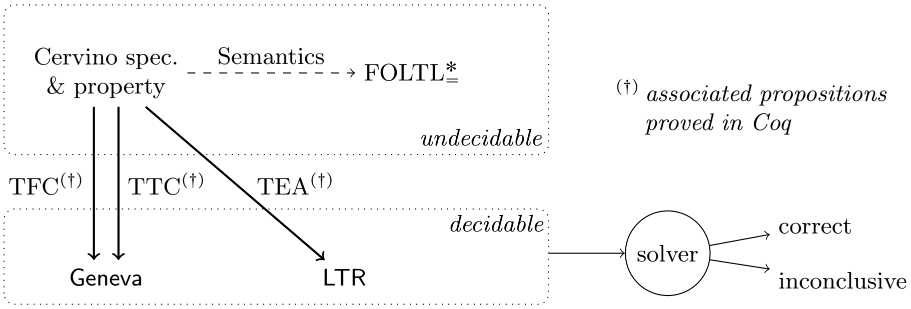

Fig. 1. Summary of the contributions of this article

reflexive-transitive closure), is formally introduced in Sect. 3.3. This language is suitable for specifying infinite-state systems. It is undecidable but we enforce some syntactic constraints anyway, in order to ease the further application of transformations mapping into decidable fragments of logic.

Cervino is illustrated in Fig. 2 using the example of a leader election protocol [6] in a ring of unbounded size. Nodes sit in a directed ring and each node has a unique ID. There is a total order on IDs. The goal of the protocol is to elect a leader (in practice, the one with the greatest ID). A node can send to its successor in the ring the IDs it knows about, the receiver keeping those that are greater than its own ID. A node is elected if it receives its own ID.

2.1 Sorts, relations and axioms

A Cervino specification may define sorts, (first-order) sorted relations and sorted constants. An interpretation structure for such a specification is a set of *infinite* traces of states. Classically, a state maps a sort to a non-empty set, a constant to an element of such a set and a relation to a set of tuples, all respecting the obvious sorting and arity constraints. The interpretation of sets and constants is *rigid* while that of relations is *flexible*.

In the example, nodes and their IDs are conflated into a single sort Node; and: an *elected* relation represents the set of elected nodes; a *succ* relation represents if two nodes are successive in the ring topology; a *toSend* relation represents the mailbox for each node; an *lte* relation defines a total ordering on nodes; an *lmax* constant represents the highest maximal identifier among nodes.

States can be constrained by `axioms`, *i.e.* sets of formulas. The latter belong to FOLTL$^*_=$, that is they can mix first-order logic (with equality) with the "always" (**G**), "eventually" (**F**) and "next" (written as a prime symbol and only applied to atoms), as well as a reflexive-transitive closure connective (written *). However, we enforce a syntactic constraint on axioms: after converting them to *negation normal form* (NNF), *an existential quantifier cannot appear in the scope of a universal quantifier or of a* **G** *connective* (no $\forall \ldots \exists \ldots$, no $\mathbf{G} \ldots \exists \ldots$).

```
sort Node // nodes are conflated with their identifiers
relation succ in Node ∗ Node // successor in the ring
  using btw // btw[succ] is enabled, succ is a function
relation lte in Node ∗ Node // "less than or equal" on nodes
relation toSend in Node ∗ Node // toSend(x,id): id is in x's mailbox
relation elected in Node // set of elected nodes
constant lmax in Node // node with maximal identifier
axiom connected { G (∀ x, y: Node · succ∗(x, y) ) }
axiom order { G { ∀ id: Node · lte(id, lmax)
                   ... } } // + classic total ordering axioms
axiom is_elected { G (∀ x: Node · elected'(x) ⇔ (elected(x) ∨ toSend'(x, x))) }
axiom init { // in the initial state...
  ∃ y: Node · succ(lmax, y) // the largest has a successor
  ∀ x, id: Node · !toSend(x, id) // empty mailboxes
  ∀ x: Node · !elected(x) } // no one is elected
event send [src: Node]
  modifies toSend at { (dst,id) · (toSend(src,id) ∨ id = src) ∧ succ(src,dst) },
    elected {
  ∀ dst: Node, id: Node · (succ(src,dst) ∧ (toSend(src,id) ∨ id = src)) ⇒
    (toSend'(dst, id) ⇔
      (toSend(dst, id) ∨ (lte(dst, id) ∧ (id = src ∨ toSend(src, id))))) }
check Safety { G (∀ x: Node · elected(x) ⇒ x = lmax ) }
  using TFC ... // + hidden parameters (see Sect. 6)
check Liveness { F (∃ y: Node · elected(y)) }
  assuming {
    ∀ src: Node · G F {
      ∀ dst: Node, id: Node · (succ(src,dst) ∧ (toSend(src,id) ∨ id = src)) ⇒
        (toSend'(dst, id) ⇔
          (toSend(dst, id) ∨ (lte(dst, id) ∧ (id = src ∨ toSend(src, id))))) } }
  using TTC ... // + hidden parameters (see Sect. 7)
```

Fig. 2. Specification of the leader election protocol (prettified syntax)

A binary relation r can by “tagged” (written **using btw**) to force r to be a function[4] and enable a special ternary relation **btw**[r]. Then, **btw**[r](x,y,z) means that there is an acyclic path between x and z passing through y. The semantics of **btw**[r] is given through axioms (see def. 14) and is related to r* through the following equivalence: r*(x,y) ⇔ **btw**[r](x,y,y).

2.2 Events

Events specify how the system may evolve from one state to another. Events (more precisely: event schemas) are declared with a name and a list of arguments that are the only variables that can appear free in the body of the event. The declaration of an event also features a `modifies` section describing which

[4] ∀ x,y,z: s · r(x,y) ∧ r(x,z) ⇒ y = z.

tuples of which relations may be modified by the event. Other relations or parts of relations are necessarily left unchanged. The body of an event is specified in *primed FO* (FO augmented with primed relation symbols representing the value of these relations in the next state) with the additional constraint that *no existential quantifier may appear positively in the body.*

The semantics for events is standard and comparable to the one used in TLA$^+$ or Electrum: in every state, at least one event is fired. In other words, there is a valuation for arguments of at least one event such that the body of the said event evaluates to true. More formally (and ignoring sorting constraints for the sake of readability), given event bodies $\phi_1, \ldots, \phi_n$ and arguments $y_1, \ldots, y_{m_i}$ appearing as free variables in ϕ_i, the semantics of event is given by the formula: $\mathbf{G}(\bigvee_{i=1}^{n} \exists y_1, \ldots y_{m_i} \cdot \phi_i)$. We insist that this formula is only implicit: it cannot be input by the specifier as it is the purpose of transformations to massage it. Finally, if needed, fairness constraints must be added by the specifier .

In the example, the *send* event represents the fact that a node updates its successor's mailbox by adding all IDs that are larger than the successor's ID. This way, the largest ID is passed along the ring. Notice we use *universal* quantification: we could have defined *dst* and *id* as parameters of *send*, but the implicit existential quantification, although theoretically acceptable, can be costly performance-wise (as *succ* is a function, this is significant for the *id* argument only). We also specify that the event `modifies` the *toSend* relation for specific pairs of a node and an identifier, only if these satisfy a condition saying that the ID is in the sender's mailbox (or corresponds to the sender's ID) and if the node is the sender's successor (the body of the event says what happens in that case).

2.3 Commands

A `check` declares a command to verify whether a property holds. To do so, a command uses a certain tactic (`TEA`, `TFC`, `TTC`), as well as additional parameters in the case of `TFC` and `TTC` (these are presented in Sect. 5 and 6, respectively). The purpose of this article is precisely to present these transformations. We notice that a command may also be associated with additional, specific axioms in an `assuming` section (in the example, this section contains a fairness property, necessary to prove the liveness property).

3 Background on FOLTL

3.1 Syntax and Semantics of FOLTL

The basic vocabulary of MSFOLTL (that we simply call FOLTL in the following) is defined out of a signature $\Sigma = (\mathcal{S}, \mathit{Const}, \mathcal{R})$ where $\mathcal{S}$ is a set of sorts, *Const* is the set of (sorted) constant symbols and $\mathcal{R} = (\mathcal{R}_{\vec{s}})_{\vec{s} \in \mathcal{S}^\star}$ is a family of sets of *relation symbols*, with $\mathcal{R}_{\vec{s}}$ the set of relation symbols over tuples of sort $\vec{s}$.

Definition 1 (Formulas). *Given a signature $\Sigma = (\mathcal{S}, \mathit{Const}, \mathcal{R})$ and a set of variables $\mathcal{V}$, FOLTL$_=$* formulas *over Σ and $\mathcal{V}$ are defined inductively by the following grammar:*

$$\psi ::= \ r(t_1, \ldots, t_n) \mid t_1 = t_2 \mid \neg\psi \mid \psi \vee \psi \mid \mathbf{X}\,\psi \mid \mathbf{F}\,\psi \mid \forall x : s \cdot \psi \mid \exists x : s \cdot \psi$$

where $x \in \mathcal{V}_s$, $r \in \mathcal{R}_{s_1,\ldots s_n}$ and $t_i \in \mathcal{V}_{s_i} \cup \mathit{Const}_{s_i}$ for each i, with $\mathcal{V}_s$ (resp. Const_s) the set of variables (resp. constants) of sort s.

$\mathbf{X}$ and $\mathbf{F}$ stand for the "next" and "eventually" connectives. Usually FOLTL includes the $\mathbf{U}$ connectives, however it is not required in this paper. We also define "always" as $\mathbf{G}\,\psi = \neg\,\mathbf{F}(\neg\psi)$. Similarly, classical propositional connectives $\wedge$, $\Rightarrow$ and $\Leftrightarrow$ are defined in the natural way. Additionally:

- We write $\psi[x]$ for a formula ψ having x as a free variable.
- We write $\mathrm{FV}(\phi)$ for the set of *free variables* of a formula, defined in the obvious way. A formula ϕ is said to be *closed* if $\mathrm{FV}(\phi) = \varnothing$.
- Classically, a formula is in *negation normal form* (NNF) if negations only appear in front of relation symbols.
- If $\mathbf{C}$ is a subset of $\{\mathbf{X}, \mathbf{F}, \mathbf{G}\}$ then we denote by FOLTL$_=(C)$ (resp. FOLTL(C)) the set of FOLTL$_=$ formulas (resp. FOLTL formulas without equality) in NNF containing only temporal operators from $\mathbf{C}$.
- A formula l is called literal if $l = r(t_1, \ldots, t_n)$ or $l = \neg r(t_1, \ldots, t_n)$ where $x \in \mathcal{V}$, $r \in \mathcal{R}_n$ and $t_i \in \mathit{Const} \cup \mathcal{V}$ for each i.

We now introduce the semantics of FOLTL$_=$. In the interpretation structures defined below, the interpretation of relations *varies* over time while that of function symbols *does not.*

Definition 2 (Interpretation Structure). *Given a signature $\Sigma = (\mathcal{S}, \mathit{Const}, \mathcal{R})$, an* (interpretation) structure $\mathcal{M}$ (over Σ) *is a triple $((D_s)_{s\in\mathcal{S}}, \sigma, \rho)$ where:*

- *$D = (D_s)_{s\in\mathcal{S}}$ is a family of pairwise-disjoint nonempty sets and each D_s is the domain of the sort s.*
- *σ maps each constant $c \in \mathit{Const}_s$ to an domain element $\sigma(c) \in D_s$.*
- *ρ maps any pair $(i, r) \in \mathbb{N} \times \mathcal{R}_{s_1\ldots s_n}$ of instant and relation to the set $\rho(i, r) \subseteq D_{s_1} \times \ldots \times D_{s_n}$ of tuples satisfying r at instant i.*

Definition 3 (Assignment). *An* assignment $\mathcal{C}$ *in domains $(D_s)_{s\in\mathcal{S}}$ for variables in $\mathcal{V}$ is a map $\mathcal{V} \to D$. We write $\mathcal{C}[x \mapsto d]$ the assignment defined as $\mathcal{C}[x \mapsto d](x) = d$ and $\mathcal{C}[x \mapsto d](y) = \mathcal{C}(y)$ if $y \neq x$. The* extension *of $\mathcal{C}$ to* terms, *also written $\mathcal{C}$, is defined in the obvious way.*

Definition 4 (Satisfaction). *Given a structure $\mathcal{M} = (D, \sigma, \rho)$ and an assignment $\mathcal{C}$, the* satisfaction *relation $\models$ is defined by induction on formulas, for any $i \in \mathbb{N}$, as follows:*

- *$\mathcal{M}, i, \mathcal{C} \models t_1 = t_2$ iff $\mathcal{C}(t_1) = \mathcal{C}(t_2)$;*

- $\mathcal{M}, i, \mathcal{C} \models r(t_1, \ldots, t_n)$ *iff* $(\mathcal{C}(t_1), \ldots, \mathcal{C}(t_n)) \in \rho_i(r)$;
- $\mathcal{M}, i, \mathcal{C} \models \neg\phi$ *iff* $\mathcal{M}, i, \mathcal{C} \not\models \phi$;
- $\mathcal{M}, i, \mathcal{C} \models \phi_1 \vee \phi_2$ *iff* $\mathcal{M}, i, \mathcal{C} \models \phi_1$ *or* $\mathcal{M}, i, \mathcal{C} \models \phi_2$;
- $\mathcal{M}, i, \mathcal{C} \models \mathbf{X}\,\phi$ *iff* $\mathcal{M}, i+1, \mathcal{C} \models \phi$;
- $\mathcal{M}, i, \mathcal{C} \models \mathbf{F}\,\phi$ *iff there exists* $k \in \mathbb{N}$ *s.t.* $\mathcal{M}, i+k, \mathcal{C} \models \phi$;
- $\mathcal{M}, i, \mathcal{C} \models \exists y : s \cdot \phi$ *iff there exists* $d \in D_s$ *s.t.* $\mathcal{M}, i, \mathcal{C}[y \mapsto d] \models \phi$;
- $\mathcal{M}, i, \mathcal{C} \models \forall x : s \cdot \phi$ *iff for every* $d \in D_s$*, we have* $\mathcal{M}, i, \mathcal{C}[x \mapsto d] \models \phi$.

Given a closed formula ϕ*, we write* $\mathcal{M}, k \models \phi$ *if* $\mathcal{M}, k, [] \models \phi$*, where* $[]$ *is the empty assignment. Then* $\mathcal{M}od(\phi)$ *denotes the set of structures* $\mathcal{M}$ *such that* $\mathcal{M}, 0 \models \phi$.

Definition 5 (Reflexive-Transitive Closure[5]). *We write* $FOLTL^*_=$ *for the enrichment of* $FOLTL_=$ *with a reflexive-transitive closure connective. Then for any sort* $s \in \mathcal{S}$ *and any binary relation symbol* $r \in \mathcal{R}_{s,s}$*, the language of* $FOLTL^*_=$ *is augmented with a fresh binary relation symbol :* $r^* \in \mathcal{R}_{s,s}$*, and we have:*
$\mathcal{M}, i, \mathcal{C} \models r^*(t_1, t_2)$ *iff* $\mathcal{M}, i, \mathcal{C} \models t_1 = t_2$ *or there exists* $n \in \mathbb{N}$ *s.t.* $\mathcal{M}, i, \mathcal{C} \models \exists x_0, \ldots, x_n \cdot t_1 = x_0 \wedge t_2 = x_n \wedge (\bigwedge_{0 \leqslant i \leqslant n-1} r(x_i, x_{i+1}))$.

Let ϕ, ϕ' be two FOLTL$^*_=$ formulas. If for any structure $\mathcal{M}$ and any assignment $\mathcal{C}$, we have $\mathcal{M}, 0, \mathcal{C} \models \phi$ iff $\mathcal{M}, 0, \mathcal{C} \models \phi'$ then we say that ϕ and ϕ' are logically equivalent, written $\phi \equiv \phi'$.

3.2 Bounded Domain Property

In this section we introduce the Bounded Domain Property (BDP) and present two fragments of FOLTL that enjoy the BDP. These fragments play an important role in the verification procedures presented in this article.

Definition 6 (Bounded Domain Property). *A fragment Frag of FOLTL enjoys the* bounded domain property *(BDP) if given* $\phi \in$ *Frag,* ϕ *is not satisfiable, or there is a domain-finite structure* $\mathcal{M}$ *s.t.* $\mathcal{M}, 0 \models \phi$ *whose the domain size is computable from* ϕ*. Additionally, BDP implies decidability.*

We now present the two fragments that are used in this paper. Both fragments are included in a larger fragment for which the BDP is established in [24].

Definition 7 (LTR fragment). *A formula* ϕ *of* $FOLTL_=$ *is said to belong to the (multisorted) Linear-Temporal Reasoning (*LTR*) fragment if* ϕ *is in NNF and existential quantifiers only appear in the head of* ϕ.

Theorem 1 ([16][24]).
Any formula $\phi \in$ LTR *(even with equality) enjoys the BDP. The bound of verification for each sort is the sum of the numbers of existential quantifiers and constant symbols over this sort.*

[5] It is possible to fully axiomatize the transitive closure in pure FOLTL, however since it does not fit into the scope of this paper such an axiomatization is not presented here and we simply extends FOLTL with the classical definition of transitive closure.

Definition 8. *An FOLTL formula ψ is in FOLTL$(\exists\uparrow, \forall\downarrow)$ if $\psi = \exists y_1 : s_1 \dots y_n : s_n \cdot \theta[y_1, \dots, y_n]$, where θ has the following syntax: $\theta ::= \ell \mid \alpha \mid \theta \vee \theta \mid \theta \wedge \theta \mid \mathbf{X}\,\theta \mid \mathbf{G}\,\theta \mid \mathbf{F}\,\theta$, where α is an FO formula in NNF without any existential quantifier and ℓ is a literal.*

Definition 9. *FOLTL$(\mathbf{X}, \mathbf{F}, \forall\downarrow)$ is defined by the following grammar: $\phi ::= \ell \mid \alpha \mid \phi \vee \phi \mid \phi \wedge \phi \mid \mathbf{X}\,\phi \mid \mathbf{F}\,\phi \mid \exists y : s \cdot \phi$, with α an FO formula in NNF without any existential quantifier, ℓ a literal and $y \in \mathcal{V}$.*

Definition 10 (Geneva fragment). *The Geneva fragment of FOLTL consists of formulas $\psi \wedge \mathbf{G}(\phi)$ s.t. ϕ is a closed formula of FOLTL$(\mathbf{X}, \mathbf{F}, \forall\downarrow)$ and ψ is a closed formula of FOLTL$(\exists\uparrow, \forall\downarrow)$.*

Definition 11. *Given a formula $\phi \in$ FOLTL$(\mathbf{X}, \mathbf{F})$ in NNF, we define its* stride *K_ϕ as the maximal number of nested $\mathbf{X}$ connectives. Formally :*

$K_\ell = K_{\mathbf{F}\,\phi} = 0$ *(if ℓ is a literal)* $\qquad K_{\mathbf{X}\,\phi} = K_\phi + 1$

$K_{\forall x \cdot \phi} = K_{\exists x \cdot \phi} = K_\phi \qquad K_{\phi_1 \wedge \phi_2} = K_{\phi_1 \vee \phi_2} = \max(K_{\phi_1}, K_{\phi_2})$

Theorem 2 ([24]). *The Geneva fragment enjoys the FDP. If $\psi \wedge \mathbf{G}(\phi)$ is a satisfiable formula in this fragment, for each sort s the (exact) bound on the domain size is : $|Const_s| + (K_\phi + 1) \times |\mathcal{V}_s|$.*

3.3 Semantics of Cervino

In this section, we define the semantics of a Cervino machine as an FOLTL$^*_=$ formula. Notice first that, in Cervino, the next instant is referred using the prime symbol, applied to relations only: this translates to an FOLTL sub-formula using the $\mathbf{X}$ connective, after application of the semantics.

Now, a frame condition is defined as a formula that specifies that a certain relation will not change (between the instant before and after the event occuring) for tuples satisfying some constraints.

Definition 12 (Frame condition). *We define a frame condition as a formula expressing that, under some hypotheses, a certain relation does not change along a transition. Given the the tuple $(r, \vec{x}, \psi)$ where $r \in \mathcal{R}_{\vec{s}}$, $\vec{x} \in \mathcal{V}^{|\vec{s}|}$, ψ is a Boolean formula, where variables in $\vec{x}$ may appear free, we define the frame condition $\mathsf{unchanged}[r, \vec{x}, \psi]$ as the formula $\forall \vec{x} : \vec{s} \cdot \psi \Rightarrow (r(\vec{x}) \Leftrightarrow \mathbf{X}\, r(\vec{x}))$.*

Definition 13 (Semantics of an event). *Let ev be an event of a Cervino machine declared as follows:* ***event*** *ev[$\vec{y}$: $\vec{s}$] modif {τ}, with modif =* ***modifies*** *q_1* ***at*** *$\{(\vec{x}_1) \cdot \psi_1\}, \dots, q_j$* ***at*** *$\{(\vec{x}_j) \cdot \psi_j\}$, where the free variables in each ψ_k are included in $\vec{x_k}, \vec{y}$. Its semantics is defined as $[\![ev]\!] = \exists \vec{y} : \vec{s} \cdot (\tau \wedge [\![modif]\!])$, where*

$$[\![modif]\!] = \Big(\bigwedge_{r \in \mathcal{R} \setminus \{q_1, \dots, q_j\}} \mathsf{unchanged}[r, \vec{x}, \top] \Big) \wedge \Big(\bigwedge_{1 \leqslant k \leqslant j} \mathsf{unchanged}[q_k, \vec{x_k}, \neg\psi_k] \Big)$$

where each list $\vec{x}$ of variables have sorts corresponding to the profile of r.

For any binary relation r that enables **btw**[r], the ternary relation **btw**[r] stating that there exists an acyclic path between two elements passing through a third element is axiomatized in FO following [18].

Definition 14 (Semantics of between). *Given a binary relation symbol r, the semantics of* ***btw****[r] is given by adding axioms of transitivity, antisymmetry, partial totality, partial reflexivity, cycle maximality, transitivity of reachability, path consistency, taken from [18] in addition to the following axiom:*

$$\forall x, y : s \cdot \Big[r(x, y) \Leftrightarrow \Big(\boldsymbol{btw}[r](x, y, y) \wedge (\forall z : s \cdot \boldsymbol{btw}[r](x, z, z) \Rightarrow \boldsymbol{btw}[r](x, y, z)) \Big) \Big] \tag{S}$$

The property (TC) relating ***btw****[r] and r^* can be deduced from the axioms provided that the domain of s is finite.*

$$\forall x, y : s \cdot \big[\boldsymbol{btw}[r](x, y, y) \Leftrightarrow r^*(x, y) \big] \tag{TC}$$

Then, calling BTW *the conjunction of all between axioms,* $[\![\boldsymbol{btw}[r]]\!] = \mathbf{G}\,\textsc{btw}$.

Definition 15 (Semantics of Cervino). *Let Mch be a Cervino machine with axioms $\psi_1, \ldots, \psi_n$, events $ev_1, \ldots, ev_m$ and such that the relations enabling* ***btw*** *are $r_1, \ldots, r_l$. Then its semantics is given by the following FOLTL* formula:*

$$[\![Mch]\!] = \phi_0 \wedge (\mathbf{G}\,\phi_{tr}) \wedge \phi_{\boldsymbol{btw}}$$

where $\phi_0 = \bigwedge_{i=1}^{n} \psi_i$, $\phi_{tr} = \bigvee_{i=1}^{m} [\![ev_i]\!]$ *and* $\phi_{\boldsymbol{btw}} = \bigwedge_{1 \leqslant i \leqslant l} [\![\boldsymbol{btw}[r_i]]\!]$

The semantics of a Cervino machine is then an FOLTL$^*_=$ formula describing the set of its traces. But, since we aim at verifying systems, we are not only interested in the set of traces but also in the set of counterexamples of a property. This set is also described by an FOLTL$^*_=$ formula which is the conjunction of the semantics of the machine and the negation of the property we aim to check.

Definition 16 (Counterexamples). *If Mch is a Cervino machine and ϕ is an FOLTL$^*_=$ formula. Then we define* $[\![Mch]\!]_\phi = [\![Mch]\!] \wedge [\![\neg\phi]\!]$

4 Basic Transformations

In this section, we present basic transformations used to build the more complex TFC and TTC tactics (respectively presented in Sect. 6 and 7). These transformations are used to map (the semantics of) a system specification into a more general Geneva formula.

4.1 Transforming Equality

Equality is replaced[6] by a dynamic congruence relation $\equiv_s$, for every sort s. The signature is therefore extended with these fresh $\equiv_s$ relations.

[6] In practice, we ensure that the semantics of the `modifies` section, which uses equality, is also affected by this transformation.

Definition 17 (Equality transformation). *Given a fresh binary relation $\equiv_s$ for every sort s of a formula, the transformation of equality is defined recursively:*

- $Abs_=(t_1 = t_2) = t_1 \equiv_s t_2$ *if the sort of* t_1 *(and necessarily of* t_2*) is* s
- $Abs_=(\ell) = \ell$
- *(the rest is just a recursive walk on formulas)*

Furthermore, the following set $\boldsymbol{Eq}_\equiv$ *of axioms is added to the whole specification:*

- *for any sort* s*:* $\mathbf{G}\, \forall x : s \cdot x \equiv_s x$
- *for any sort* s*:* $\mathbf{G}\, \forall x : s, y : s \cdot x \equiv_s y \Rightarrow y \equiv_s x$
- *for any sort* s*:* $\mathbf{G}\, \forall x : s, y : s, z : s \cdot x \equiv_s y \wedge y \equiv_s z \Rightarrow x \equiv_s z$
- *for any relation* r *and adequate sorts* $\vec{s}$ *conforming to the profile of* r*:*
  $\mathbf{G}\, \forall \vec{x} : \vec{s}, \vec{y} : \vec{s} \cdot (x_1 \equiv_{s_1} y_1 \wedge \ldots \wedge x_n \equiv_{s_n} y_n) \Rightarrow (r(\vec{x}) \Leftrightarrow r(\vec{y}))$

Lemma 1. *Given an FOLTL*$_=$ *formula* ϕ*, if* ϕ *is satisfiable then* $Abs_=(\phi)$ *is satisfiable (and does contain* $=$ *anymore).*

Proof. Proof validated in Coq. It is easy to see that equality is a particular case of the equivalence relation introduced by this transformation. □

4.2 Restricted Skolemization

The following transformation corresponds to a form of Skolemization meant to create only new constants symbols. Its main purpose is to introduce constants that can then be used by instantiation (Sect. 4.3). Existentially-quantified variables can be substituted by fresh constants, except when under a **G** connective.

Definition 18 (Skolemization). *Skolemization is defined by the following operation (all fresh constant symbols are added to the signature):*

- $Abs_\exists(t_1 = t_2) = t_1 = t_2$ *and* $Abs_\exists(\ell) = \ell$
- $Abs_\exists(\mathbf{G}\, \phi) = \mathbf{G}\, \phi$
- $Abs_\exists(\forall x : s \cdot \phi) = \forall x : s \cdot \phi$
- $Abs_\exists(\exists y : s \cdot \phi) = Abs_\exists(\phi[y \mapsto c])$ *where* c *is a fresh constant symbol*
- *(the rest is just a recursive walk on formulas)*

Lemma 2. *Given an FOLTL*$_=$ *formula* ϕ*, then* $Abs_\exists(\phi)$ *and* ϕ *are equisatisfiable.*

Proof. Proof validated in Coq. Corresponds to a usual Skolemization procedure. □

4.3 Instantiation

One of the main limitations of the Geneva fragment is the prohibition of temporal operators under universal quantifiers. The solution we propose to this problem is to *finitely* instantiate such universal quantifiers. The following transformation formalizes this idea: all universal quantifiers over temporal formulas are replaced by a conjunction over the set of constants and existentially-bound variables.

Definition 19 (Forall instantiation). *Given a set $\mathcal{I}$ of constant and variable symbols, we define the transformation of universal quantifiers as follows:*

- $Abs_{\forall,\mathcal{I}}(t_1 = t_2) = t_1 = t_2$ *and* $Abs_{\forall,\mathcal{I}}(\ell) = \ell$
- $Abs_{\forall,\mathcal{I}}(\exists y : s \cdot \phi) = \exists y : s \cdot Abs_{\forall,\mathcal{I}\cup\{y\}}(\phi)$
- *if* $\phi \in FO$ *(ϕ does not contain temporal connectives) then* $Abs_{\forall,\mathcal{I}}(\forall x : s \cdot \phi) = \forall x : s \cdot \phi$*, otherwise* $Abs_{\forall,\mathcal{I}}(\forall x : s \cdot \phi) = \bigwedge_{c\in\mathcal{I}_s} Abs_{\forall,\mathcal{I}}(\phi[x \mapsto c])$ *(where $\mathcal{I}_s$ is the set of terms in $\mathcal{I}$ of sort s)*
- *(the rest is just a recursive walk on formulas)*

Remark 1. There is no need to transform a universal quantifier if all temporal operators in its scope permute with it, for instance: $\forall x \cdot \mathbf{G}\, P$ is equivalent to $\mathbf{G}(\forall x \cdot P)$ and $\forall x \cdot (\mathbf{X}\, P) \Rightarrow (\mathbf{X}\, Q)$ is equivalent to $\mathbf{X}(\forall x \cdot P \Rightarrow Q)$.

Lemma 3. *Given an FOLTL$_=$ formula ϕ, if ϕ is satisfiable and $\mathcal{I} \subseteq Const$ then $Abs_{\forall,\mathcal{I}}(\phi)$ is satisfiable.*

Proof. Proof validated in Coq. This operation consists in instantiating universal operators, thus preserving satisfiability. □

4.4 Addressing Transitive Closure and the Between Relation

Since we target fragments of FOLTL (without transitive closure), we define the transformation $Abs_*()$, which leaves a formula unchanged except it *uninterprets* the operator *, *i.e.*, $Abs_*(\phi)$ returns ϕ where every occurrence of r^* is considered as a new relation symbol, unrelated with r.

Besides, the between relation axioms does not fit into Geneva or LTR, so we define their abstract semantics as follows.

Definition 20 (Transformation of between axioms). *Given a binary relation symbol r, we define $(\!|\boldsymbol{btw}[r]|\!) = \mathbf{G}\,\mathrm{BTW}$ where BTW is the conjunction of the axioms from Def.14, except that*

- *the axiom S is replaced by the axiom (AS) (in order to prevent existential quantifier in the scope of a universal one)*
- *and the property (TC) relating r^* and **btw**[r] is now considered as an axiom (since r^* has no semantics in the targeted FOLTL fragments)*

$$\forall x, y : s \cdot \Big[r(x,y) \Rightarrow \Big(\boldsymbol{btw}[r](x,y,y) \wedge (\forall z : s \cdot \boldsymbol{btw}[r](x,z,z) \Rightarrow \boldsymbol{btw}[r](x,y,z)) \Big) \Big] \tag{AS}$$

$$\forall x, y : s \cdot \big[\boldsymbol{btw}[r](x,y,y) \Leftrightarrow r^*(x,y) \big] \tag{TC}$$

4.5 Geneva Transformation

The basic transformations introduced above are mainly used together, in a specific order.

Definition 21 (Geneva Transformation). *We define:*

$$Abs_{Gen}(\phi) = Abs_{*}(Abs_{\forall,Const}(Abs_{\exists}(\boldsymbol{Eq}_{\equiv} \wedge Abs_{=}(\phi))))$$

Theorem 3. *Given $\psi \in FOLTL_{\vec{y_1}}(\forall)$ and $\phi \in FOLTL_{\vec{y_1}\cup\vec{y_2}}(\mathbf{X}, \mathbf{F}, \forall)$ then $Abs_{Gen}(\exists\vec{y_1} : \vec{s_1} \cdot (\psi \wedge \mathbf{G}(\exists\vec{y_2} : \vec{s_2} \cdot \phi)))$ belongs to the* Geneva *fragment.*

Proof. Recall the conditions to belong to Geneva: (1) no **G** operator in the scope of an existential quantifier that is itself under an **G** connective; (2) no existential quantifier in the scope of a universal quantifier; (3) no equality; (4) no temporal quantifier in the scope of universal quantifiers; and (5) no transitive closure. Given ψ, ϕ satisfying the given hypotheses, let us write $\alpha = \exists\vec{y_1} : \vec{s_1} \cdot (\psi \wedge \mathbf{G}(\exists\vec{y_2} : \vec{s_2} \cdot \phi))$. Then, in α, existential quantifiers appear either at the head of the formula or under an **G** operator over the ϕ formula. Since ϕ contains no other temporal connectives than **X** and **F**, condition (1) is met. Condition (2) is met as all existential quantifiers appear before universal quantifiers. $Abs_{=}(.)$ ensures that equality is not used in the final formula, thus ensuring condition (3). $Abs_{\forall,Const}(.)$ instantiates all universal quantifiers that contain temporal connectives in their scope (we assume that if such operator could have been swapped with an universal quantifier, it has been done beforehand), which ensures condition (4). Finally $Abs_{*}(.)$ erases the reflexive transitive closure, ensuring condition (5). Since it is obvious that none of the transformations can introduce formulas breaking any of the conditions, we conclude that $Abs_{Gen}(\alpha)$ belongs to Geneva. □

5 TEA: Transforming existential quantifiers

We now present the fully-automatic TEA transformation. It starts with the observation that the formula specifying events (see def. 13) is of the shape $\mathbf{G}\,\exists\vec{x} \cdot \bigvee_i ev_i(\vec{x})$, that is, in every state, at least an event is fired. The gist of the TEA transformation is then twofold: (1) we replace these existential quantifiers by *universal* ones; (2) for every such existential quantifier, we add a fresh relation $\mathbb{E}$, which holds only for the constant semantically associated to this quantifier.

The whole resulting abstract specification lies in the LTR fragment, which enjoys the BDP (Theorem 1). The formula specifying events is however more general than the original one, because it allows more transitions to happen. The abstract system may thus violate a property holding on the original specification. But it is now decidable to check whether the property holds in the abstract system and, if so, this entails that it also holds in the original system.

Before presenting the transformation, notice that, in the following, we consider *event formulas*, that is *primed* FO$_=$ formulas of the shape $\phi = \exists y_1 : s_{y_1}, \ldots, y_n : s_{y_n} \cdot \forall x_1 : s_{x_1}, \ldots, x_m : s_{x_m} \cdot \psi$, where ψ is in NNF and does not contain any first-order quantifiers. These formulas naturally arise when putting the semantics of events in prenex normal form. We also suppose we have a supply of fresh relation symbols, written $\mathbb{E}_i$ (one for every y_i, $1 \leqslant i \leqslant n$).

To devise the transformation and prove its soundness, we first introduce a formula specifying that the $\mathbb{E}$ relations are functional. This schema appears in the final abstract specification.

Definition 22 (Functional $\mathbb{E}$ relations). *Given an event formula $\phi = \exists y_1 : s_{y_1}, \ldots, y_n : s_{y_n} \cdot \forall x_1 : s_{x_1}, \ldots, x_m : s_{x_m} \cdot \psi$, we define the* functional formula based on ϕ *as:* $\mathrm{Ax}^{\mathbb{E}}(\phi) = \mathbf{G}\Big(\bigwedge_{i=1}^{n} \forall z_1, z_2 : s_{y_i} \cdot (\mathbb{E}_i(z_1) \wedge \mathbb{E}_i(z_2)) \Rightarrow z_1 = z_2\Big)$ *where $\mathbb{E}_1, \ldots, \mathbb{E}_n$ are fresh unary relation symbols.*

As we introduce these $\mathbb{E}$ relations, we also define an enrichment of the event formula accounting for the extended signature. This new formula appears as a link between the two lemmas entailing soundness.

Definition 23 (Enriched event formula). *Given an event formula $\phi = \exists y_1 : s_{y_1}, \ldots, y_n : s_{y_n} \cdot \forall x_1 : s_{x_1}, \ldots, x_m : s_{x_m} \cdot \psi$, we define the* enriched event formula based on ϕ *as:*

$$\overline{\phi} = \mathrm{Ax}^{\mathbb{E}}(\phi) \wedge \Big[\exists y_1 : s_{y_1}, \ldots, y_n : s_{y_n} \cdot \Big(\bigwedge_{i=1}^{n} \mathbb{E}_i(y_i) \wedge \forall x_1 : s_{x_1}, \ldots, x_m : s_{x_m} \cdot \psi\Big)\Big]$$

where $\mathbb{E}_1, \ldots, \mathbb{E}_n$ are fresh unary relation symbols.

We now present the essential part of the transformation, transforming an event formula ϕ into a purely universal one $\mathbb{U}(\!|\phi|\!)$, more general than $\overline{\phi}$. In other words, $\mathbb{U}(\!|\phi|\!)$ allows more transitions than ϕ if we ignore the specification of $\mathbb{E}_1, \ldots \mathbb{E}_n$. To do that, for any variable y whose corresponding fresh relation is $\mathbb{E}$, we proceed with the following steps. First: equality between y and another variable is replaced with the relation $\mathbb{E}$ applied to the latter; and any other literal ℓ containing y is replaced by $\mathbb{E}(y) \Rightarrow \ell$. Once these transformation are done, it is possible to replace existential quantification over y by a universal quantification.

Definition 24 (Transformation). *Given an event formula ϕ of shape $\exists y_1 : s_{y_1}, \ldots, y_n : s_{y_n} \cdot \forall x_1 : s_{x_1}, \ldots, x_m : s_{x_m} \cdot \psi$, we define the* (TEA) transformation function *on ϕ as:*

$$\mathbb{U}(\!|\phi|\!) = \forall y_1 : s_{y_1}, \ldots, y_n : s_{y_n} \cdot \mathbb{U}_{\vec{y}}(\!|\forall x_1 : s_{x_1}, \ldots, x_m : s_{x_m} \cdot \psi|\!)$$

where $\vec{y} = \{y_1, \ldots, y_n\}$ and where $\mathbb{E}_1, \ldots, \mathbb{E}_n$ are fresh relation symbols (one for every $y \in \vec{y}$); with $\mathbb{U}_{\vec{y}}(\!|\psi|\!)$ defined recursively as follows:

- $\mathbb{U}_{\vec{y}}(\!|y_i = y_j|\!) = (\mathbb{E}_i(y_i) \Rightarrow \mathbb{E}_j(y_i)) \wedge (\mathbb{E}_j(y_j) \Rightarrow \mathbb{E}_i(y_j)) = (\neg\mathbb{E}_i(y_i) \vee \mathbb{E}_j(y_i)) \wedge (\neg\mathbb{E}_j(y_j) \vee \mathbb{E}_i(y_j))$
- $\mathbb{U}_{\vec{y}}(\!|y_i \neq y_j|\!) = (\mathbb{E}_i(y_i) \Rightarrow \neg\mathbb{E}_j(y_i)) \wedge (\mathbb{E}_j(y_j) \Rightarrow \neg\mathbb{E}_i(y_j)) = (\neg\mathbb{E}_i(y_i) \vee \neg\mathbb{E}_j(y_i)) \wedge (\neg\mathbb{E}_j(y_j) \vee \neg\mathbb{E}_i(y_j))$
- $\mathbb{U}_{\vec{y}}(\!|y_i = d|\!) = \mathbb{U}_{\vec{y}}(\!|d = y_i|\!) = \mathbb{E}_i(d)$ *where $d \notin \vec{y}$ (d is either a constant or a variable in $\vec{x}$)*
- $\mathbb{U}_{\vec{y}}(\!|y_i \neq d|\!) = \mathbb{U}_{\vec{y}}(\!|d \neq y_i|\!) = \neg\mathbb{E}_i(d)$ *where $d \notin \vec{y}$ (d is either a constant or a variable in $\vec{x}$)*
- $\mathbb{U}_{\vec{y}}(\!|\ell|\!) = (\bigwedge_{k=1}^{i} \mathbb{E}_{a_k}(y_{a_k})) \Rightarrow \ell = (\bigvee_{k=1}^{i} \neg\mathbb{E}_{a_k}(y_{a_k})) \vee \ell$ *where ℓ is a (possibly primed) literal and $\{y_{a_1}, \ldots y_{a_i}\} = \mathrm{FV}(\ell) \cap \vec{y}$*

- *(the rest is just a recursive walk on formulas)*

Example 1. Consider the following event formula, stating that there is an event making R true in the next state for a variable y (other variables remain unchanged w.r.t. R): $\phi = \exists y : A \cdot R'(y) \wedge (\forall x : A \cdot x \neq y \Rightarrow (R(x) \Leftrightarrow R'(x)))$ that is, in prenex form: $\exists y : A \cdot \forall x : A \cdot R'(y) \wedge (x = y \vee (\neg R(x) \wedge \neg R'(x)) \vee (R(x) \wedge R'(x)))$. Then there is only one fresh $\mathbb{E}$ relation, and $\mathbb{U}\llparenthesis\phi\rrparenthesis$ is:

$$\forall y, x : A \cdot (\neg \mathbb{E}(y) \vee R'(y)) \wedge \big(\mathbb{E}(x) \vee (\neg R(x) \wedge \neg R'(x)) \vee (R(x) \wedge R'(x))\big)$$

Now, the following lemma states that every model of the enriched event formula is also a model for the transformed event formula.

Lemma 4. *Given an event formula $\phi = \exists y_1 : s_{y_1}, \ldots, y_n : s_{y_n} \cdot \forall x_1 : s_{x_1}, \ldots, x_m : s_{x_m} \cdot \psi$, we have $\overline{\phi} \models \mathbb{U}_{\vec{y}}\llparenthesis\phi\rrparenthesis$.*

Proof. Proof validated in Coq.

Lemma 5 applies to a formula representing a whole specification: if such a specification is satisfiable, then a certain transformed version of it is satisfiable too.

Lemma 5. *Let θ be an FOLTL$_=$ formula, and ϕ be an event formula on the same signature. Then if $\theta \wedge \mathbf{G}\,\phi$ is satisfiable, $\theta \wedge \mathbf{G}(\mathbb{U}_{\vec{y}}\llparenthesis\phi\rrparenthesis) \wedge \mathrm{Ax}^{\mathbb{E}}(\phi)$ is also satisfiable.*

Proof. Proof validated in Coq.

Definition 25 (Abstract semantics). *Given a Cervino machine Mch such that the relations enabling* ***btw*** *are $r_1, \ldots, r_l$, we define $\mathbb{U}\llparenthesis Mch\rrparenthesis = \phi_0 \wedge \mathbf{G}\,\mathbb{U}\llparenthesis\phi_{tr}\rrparenthesis \wedge \phi_{\boldsymbol{btw}}$, where ϕ_0 and ϕ_{tr} are defined as in Def. 15 and $\phi_{\boldsymbol{btw}} = \bigwedge_{1 \leqslant i \leqslant l} \llparenthesis \boldsymbol{btw}[r_l]\rrparenthesis$. Also, given an FOLTL$_=$ formula ϕ, we define $\mathbb{U}_\phi\llparenthesis Mch\rrparenthesis = Abs_*(\mathbb{U}\llparenthesis Mch\rrparenthesis \wedge \neg\phi)$.*

Theorem 4 (Soundness). *If $[\![Mch]\!]_\phi$ is satisfiable, then $\mathbb{U}_\phi\llparenthesis Mch\rrparenthesis$ is also satisfiable.*

Proof. This is a direct application of Lemma 5.

Theorem 5. *Given a Cervino machine Mch such that ϕ_0 and ϕ_{tr} are defined as in Def. 15, if $\phi_0, \phi \in \mathsf{LTR}$ then $\mathbb{U}_\phi\llparenthesis Mch\rrparenthesis \in \mathsf{LTR}$.*

Proof. Directly follows from the definition of $\mathbb{U}\llparenthesis.\rrparenthesis$.

6 TFC: Transforming Frame Conditions

The TEA transformation has the advantage of being fully automatic but it can be inconclusive in a number of cases. For instance, the verification of a distributed system involving strong interactions between its components, which induces events with two or more parameters, is likely to be inconclusive using TEA. This is because the universal quantifiers that are introduced by TEA are abstracting these interactions (which are naturally expressed with existential quantifiers) in a too drastic way.

In this section, we present another transformation, called TFC, which overcomes these limitations but requires some intervention from the specifier.

Instead of targeting the LTR fragment, we now target the Geneva one, which allows for existential quantifiers in the scope of $\mathbf{G}$, but forbids temporal formulas in the scope of a universal quantifier. As a consequence, frame conditions, which are typically of shape $\forall x : s \cdot \varphi_{cond} \Rightarrow (r(x) \Leftrightarrow \mathbf{X}\, r(x))$, are not expressible in Geneva. In order to fit into it, such universal quantifiers are instantiated over constants (see $Abs_{\forall, Const}(\cdot)$ defined in Sect. 4.3). But then a large part of the information included in the frame conditions is lost. Therefore, we associate some particular kind of invariant properties, called stability axioms, with each event, as a finer transformation of frame conditions. Intuitively, a stability axiom is a pure FO formula that is preserved by an event. Since it is expressed in pure FO, the preservation of a stability axiom is then expressible in Geneva.

Definition 26 (Stability Axiom). *Given a set of frame conditions $\mathcal{C}$, an FO formula ϕ is a* stability axiom *for $\mathcal{C}$ if $\mathcal{C} \models \phi \Rightarrow \mathbf{X}\, \phi$.*

$\mathrm{St}_{\mathcal{C}}$ *denotes the set of stability axioms for $\mathcal{C}$.*

The specification of stability axioms is a creative step, but it can be eased with the help of a syntactic condition, which is sufficient to be a stability axiom. The idea is that a formula of the following shape is necessarily a stability axiom: $\varphi_{\text{hyp}} \Rightarrow \varphi$, were φ_{hyp} corresponds to the guard of a frame condition that leaves a relation r unchanged, and φ only refers to the relation r.

Example 2. In order to illustrate the use of stability axioms, let us consider the leader election distributed system, introduced in Sect. 2. Since TEA does not succeed in proving the safety property, we can try TFC with the following stability axiom for event send:

∀ x,y: Node · !succ(src,x) ⇒ (!toSend(x, y) ∨ (x ≠ lmax ∧ **btw**[succ](x, lmax, y)))

This axiom expresses that if a node x different from the successor of src has an ID y in its mailbox, then the node with the greatest ID is located between x and y (recall that a node and its ID are conflated). This means that outside the scope of the event, an ID cannot jump over the node with the greatest identifier.

Exhibiting this stability axiom requires some work. It would also be possible to proceed using an inductive invariant but, since the property to check is not inductive, doing so would also require some effort.

Example 3. In order to illustrate the difference between stability axioms and inductive invariants, we take a toy token protocol as an example. For the sake of simplicity, we consider a property to check that is already inductive. The protocol features one token passing from nodes to nodes with only one send event send(x,y), with body: token(x) ∧ !token'(x) ∧ token'(y) ∧ frame, where frame := ∀ z · (z ≠ x ∧ z ≠ y) ⇒ (token(z) ⇔ token'(z)).

The (inductive) property to check is that there is always at most one node holding the token. To prove this property without relying on its inductiveness, we can use the following stability axiom: stab := ∀ z · (z ≠ x ∧ z ≠ y) ⇒ !token(z). Contrary to the inductive invariant, the stability axiom has free variables matching the parameters of the event (which are implicitly quantified existentially). Also the preservation of the stability axiom follows from the frame condition as frame ⊨ stab ⇒ $\mathbf{X}$ stab, while the preservation of the inductive invariant follows from the whole transition.

Remark 2. Notice that this property is also true for the nodes that are in the scope of the event, *i.e.*, src and its successor. So in this case, the stability axiom is very close to an invariant property. But this is not the case in general. A distinguishing aspect is that TFC with this stability axiom succeeds in proving the safety property, whereas it would not be possible to deduce it from the "invariant" version of this stability axiom.

The TFC transformation is performed in two phases:

1. Stability axioms, which are provided by the specifier, are added to the body of each event. At this step, the semantics of Cervino is strengthened by the transformation. The obtained formula is not in the Geneva fragment, in particular because of the frame conditions.
2. The Geneva transformation, which is presented in Sect. 4, is applied. In particular, the frame conditions are abstracted by equality transformation and instantiation, but the stability axioms are left unchanged.

Definition 27 (Event enrichment with a stability axiom). *Let ev be an event of a Cervino machine declared as:* ***event*** *ev[$\vec{y}$: $\vec{s}$] modif{τ} and $\mathcal{C}$ be the frame condition of ev,* i.e., $\mathcal{C} = [\![modif]\!]$. *Given a stability axiom $\mathcal{I}$ for $\mathcal{C}$, we define the enrichment $\rho(\!|ev, \mathcal{I}|\!)$ of ev with $\mathcal{I}$ as: $\rho(\!|ev, \mathcal{I}|\!) = \exists \vec{y} : \vec{s} \cdot \tau \wedge \mathcal{C} \wedge (\mathcal{I} \Rightarrow \mathbf{X}\,\mathcal{I})$.*

Definition 28 (Cervino machine enrichment with stability axioms). *Let Mch be a Cervino machine with axioms $\psi_1, \ldots, \psi_n$, events $ev_1, \ldots, ev_m$ declared as* ***event*** *ev_i [$\vec{y}_1$: $\vec{s}_1$] modif {τ_i} for each $i \in 1..m$ and such that the relations enabling* ***btw*** *are $r_1, \ldots, r_l$. Let* sta *be a function mapping each event to a stability axiom for the according frame condition. Using the same notation as def. 27, we define the stability axiom enrichment $\rho(\!|Mch, \mathsf{sta}|\!)$ of Mch as $\rho(\!|Mch, \mathsf{sta}|\!) = \phi_0 \wedge \mathbf{G}\,\phi_{tr} \wedge \phi_{\boldsymbol{btw}}$*

where $\phi_0 = \bigwedge_{i=1}^{n} \psi_i$, $\phi_{tr} = \bigvee_{i=1}^{m} \rho(\!|ev_i, \mathsf{sta}(ev_i)|\!)$ and $\phi_{\boldsymbol{btw}} = \bigwedge_{1 \leqslant i \leqslant l} (\!|\boldsymbol{btw}[r_l]|\!)$.

Definition 29 (Abstract semantics). *Given a Cervino machine Mch and a function* sta*, mapping each event to a stability axiom, we define the stability axiom semantics as $\mathbb{F}(\!|Mch|\!)_\phi = Abs_{Gen}(\rho(\!|Mch, \mathsf{sta}|\!) \wedge \neg\phi)$*

Theorem 6 (Soundness). *If $[\![Mch]\!]_\phi$ is satisfiable then $\mathbb{F}(\!|Mch|\!)_\phi$ is satisfiable.*

Proof. Follows from Lemmas 1, 2 and 3.

Theorem 7. *If $\neg\phi \in$ LTR then $\mathbb{F}(\!|Mch|\!)_\phi \in$ Geneva.*

Proof. Follows from Theorem 3.

7 TTC: Transforming Reflexive-Transitive Closure

We now present a simple, effective transformation technique to approximate reflexive-transitive closure (which is present in Cervino and its FOLTL$^{*}_{=}$ semantics). This technique has shown to be useful to prove some liveness properties.

As is well known, transitive closure cannot be fully specified in pure FO. On the other hand, it *can* be specified in pure FOLTL, but the axiomatization we are aware of does not fit in the fragments considered here. However, it is possible to define an interesting *approximation* that does fit in the Geneva fragment.

Informally, the crux of our technique relies on the following observation: *any property propagating along a binary relation will eventually propagate to the reflexive-transitive closure thereof.* This is proved (see Theorem 8 below) by following the definitions of the transitive closure and of the eventually connective.

Definition 30 (Propagation schema). *Given binary relations $\boldsymbol{r}$ and $\boldsymbol{t}$ on a sort s, given a formula P with $k+1$ free variables ($k \geqslant 0$), the first of which (of sort s) is distinguished in the following. Given k variables $\vec{x}$ of appropriate typing, we define the* propagation *and* closure schemas *as follows:*

$$\begin{aligned}\boldsymbol{Propagates}[\boldsymbol{r}, P, \vec{x}] &= \forall u, v : s \cdot \boldsymbol{r}(u,v) \Rightarrow \mathbf{G}\,(P[u,\vec{x}] \Rightarrow \mathbf{F}\,P[v,\vec{x}]) \\ \boldsymbol{Closure}[\boldsymbol{r}, \boldsymbol{t}, P, \vec{x}] &= \boldsymbol{Propagates}[\boldsymbol{r}, P, \vec{x}] \Rightarrow \boldsymbol{Propagates}[\boldsymbol{t}, P, \vec{x}]\ \ .\end{aligned}$$

Theorem 8 (Propagation). *Given a binary relations $\boldsymbol{r}$ on a sort s, the following property over its reflexive-transitive $\boldsymbol{r}^{\star}$ closure is valid: $\boldsymbol{Closure}[\boldsymbol{r}, \boldsymbol{r}^{\star}, P, \vec{x}]$.*

Proof. Proof validated in Coq.

The proof sketch is the following : we consider the set of element to which the property eventually propagates. Then we use the hypothesis that the property propagates along a binary relation r to show that this set is closed under the relation r. Then as the transitive closure from some element is the smallest set closed under the relation r, we know that the property propagates to any element in the transitive closure.

We prove that under the $\mathbf{Propagates}[\mathbf{r}, P, \vec{x}]$ hypothesis, for any u, the set of v's satisfying $\mathbf{G}\,(P[u,\vec{x}] \Rightarrow \mathbf{F}\,P[v,\vec{x}])$ is included in the set of v's that are reachable from u along $\mathbf{r}$. Let $\mathcal{M}$ be a structure and $\mathcal{C}$ an assignment s.t. $\mathcal{M}, \mathcal{C} \models \mathbf{Propagates}[\mathbf{r}, P, \vec{x}]$. We assume that there is an instant i such that $P[u,\vec{x}]$ holds (otherwise the satisfaction of the axiom is trivial). Then $\mathcal{M}, i, \mathcal{C} \models \mathbf{F}\,P[u,\vec{x}]$. Also, given v s.t. $\mathcal{M}, i, \mathcal{C} \models \mathbf{F}\,P[v,\vec{x}]$, there exists $k \geqslant i$ s.t. $\mathcal{M}, k, \mathcal{C} \models P[v,\vec{x}]$. For any v' s.t. $\mathcal{M}, 0, \mathcal{C} \models \mathbf{r}(v,v')$ $\mathbf{Propagates}[\mathbf{r}, P, \vec{x}]$ implies $\mathcal{M}, k, \mathcal{C} \models \mathbf{F}\,P[v',\vec{x}]$. Thus $\mathcal{M}, i, \mathcal{C} \models P[v',\vec{x}]$. Then $\mathcal{M}, 0, \mathcal{C} \models \mathbf{r}^{\star}(u,v)$ implies $\mathcal{M}, i, \mathcal{C} \models \mathbf{F}\,P[v,\vec{x}]$. Hence $\mathbf{Closure}[\mathbf{r}, \mathbf{r}^{\star}, P, \vec{x}]$ is valid. □

Given this theorem, the technique we propose consists in replacing the reflexive-transitive closure of a relation (which fits in Cervino and FOLTL$^{*}_{=}$) by an uninterpreted relation satisfying the closure schema shown above, for some property P that depends on the sort of the considered binary relation as well as, possibly, other arguments. Remark that finding such a property P requires creativity: the specifier must come up with a relevant propagating property.

Example 4. In the case of the leader election example, we use `TTC` to check that a leader will be elected at some point. The property we use is propagation along *succ* of having a given ID in one's mailbox ($\textbf{Propagates}[\mathit{succ}, \mathit{toSend}, \mathit{id}]$).

Definition 31 (Abstract semantics). *Let Mch be a Cervino machine, such that $r_{j_1}, \ldots, r_{j_l}$ are binary relations enabling **btw**, and $r_{k_1}, \ldots, r_{k_m}$ are binary relations whose reflexive-transitive closure is used in Mch. Now, given formulas $P_1, \ldots, P_\ell$, where for every $1 \leqslant i \leqslant m$, $\mathrm{FV}(P_i) = \{x, x_1, \ldots, x_{n_i}\}$ (with x the distinguished free variable), we define the transitive closure transformation as:*

$$\mathbb{T}(\!|Mch|\!) = \phi_0 \wedge \mathbf{G}\,\phi_{tr} \wedge \phi_{\boldsymbol{btw}} \\ \wedge \Big(\bigwedge_{1 \leqslant i \leqslant m} \bigwedge_{(c_1, \ldots c_{n_i}) \in Const^{n_i}} \boldsymbol{Closure}[r_{k_i}, r^*_{k_i}, P_i, (c_1, \ldots c_{n_i})]\Big)$$

where ϕ_0 and ϕ_{tr} are defined as in Def 15 and $\phi_{\boldsymbol{btw}} = \bigwedge_{1 \leqslant i \leqslant l} (\!|\boldsymbol{btw}[r_{j_i}]|\!)$.

We also define $\mathbb{T}(\!|Mch|\!)_\phi = Abs_{Gen}(\mathbb{T}(\!|Mch|\!) \wedge \neg\phi)$ (notice that, due to the application of the Geneva transformation, the r^*_i relations become uninterpreted).

Theorem 9 (Soundness). *If $[\![Mch]\!]_\phi$ is satisfiable then $\mathbb{T}(\!|Mch|\!)_\phi$ is satisfiable.*

Proof. Follows directly from Theorem 8 and Lemmas 1, 2 and 3.

Theorem 10. *If $\neg\phi \in$ LTR then $\mathbb{T}(\!|Mch|\!)_\phi \in$ Geneva.*

Proof. Follows from Theorem 3.

8 Evaluation

To evaluate the relevance of our three tactics, we applied them to several models of distributed protocols. Our research questions were (1) to check that our methods were applicable to real models; (2) to check whether our approach was efficient enough; and (3) to assess the effort for the specifier to come up with parameters for TFC and TTC. Our strategy was always first to apply the TEA tactic. If TEA failed, then in the case of safety properties, we devised stability axioms in order to apply TFC. Otherwise, for liveness properties and for systems relying on transitive closure, we relied on TTC.

The Cervino prototype takes a Cervino specification as input and generates Electrum models which are then fed to the Electrum Analyzer [4], which itself calls a complete procedure in nuXmv [5]. On a general note, efficiency can be compromised in the case of the TTC and TFC tactics due to larger inferred bounds than for TEA. Furthermore, the size of LTL formulas generated by Electrum for nuXmv grows quickly as the tool merely unfolds quantifiers into conjunction and disjunctions, depending on the bounds. For this reason, we leveraged some properties of the Geneva fragment to end up with smaller models: (1) the size of each domain is an *exact* bound rather than just an upper one; (2) all constants are distinct; (3) existential quantifiers can be unfolded on

Specification	Type	Technique	Bound	Effort
TLB shootdown	Safety	TEA	2	–
Dining philosophers	Safety	TEA	2	–
Lock server	Safety	TEA	2	–
Gset (CRDT)	Liveness	TEA	2	–
2Pset (CRDT)	Liveness	TEA	2	–
Leader election	Safety	TFC	7	4
	Liveness	TTC	6	1
Token ring	Safety	TFC	7	7
	Liveness	TTC	6	1
FIFO	Liveness	TTC	6	1

Fig. 3. All verifications take less than 20 s ("effort": estimation of user effort with the number of atoms (literals and equality tests) used in the TTC or TFC parameters).

a limited part of the domain. This is the case because the proof of the BDP for the Geneva fragment [24] shows that, if there is a model of a Geneva formula, there is a model satisfying these properties. The specifications we evaluated are of moderate complexity but are not just toy models:

TLB shootdown The TLB Shootdown algorithm [3] is part of the Mach operating system. Processors keep a cache of page tables in a Translation Lookaside Buffer (TLB). The safety property we prove is that whenever the protocol ensures that the page table is updated, the corresponding update will be flushed by either the initiator or the responder.

Dining philosophers This classic protocol features an unbounded number of philosophers sharing forks. We prove a mutual exclusion property, that is that a fork cannot be simultaneously held by two different philosophers.

Lock server We present a simple lock server protocol studied in Verdi [27] and Ivy [21]. The protocol features a single server and an unbounded number of clients willing to hold a lock. The safety property we verify is that two different clients cannot simultaneously hold the lock.

Gset and 2Pset Conflict-free Replicated Data Types (CRDTs) are a family of concurrent protocols where a data structure is replicated in a network and where the replicas can be independently and concurrently updated. We model the Grow-only Set (G-Set) [26] and the 2-Phase Set [26] CRDTs. In both cases, we prove that any update is eventually delivered to all replicas.

Leader election The leader election protocol, presented in Sec. 2, is inspired by [6]. We notice however that a node sends all the contents of its mailbox at once, which is a strong simplification.

Token ring Token Ring is a classic protocol where a token is passed through nodes with mailboxes in a ring. We prove a safety and a liveness property. In the first case, we use the TFC tactic with 2 stability axioms, one for each event, which basically state that if there is no token apart from the one transferred, then no token can appear on unmodified nodes during the event. The TTC parameter says that the property of holding the token is the one that should propagate, under strong fairness.

FIFO This protocol is a simple mutual exclusion protocol based on a FIFO strategy. We prove a liveness property using TTC, stating that for any integer i, being in the i-th position of the list is a propagating property.

Our conclusion to these case studies is the following (Fig. 3). First the TEA tactic is only efficient for models involving few interactions, which can be attributed to the loss of precision when using universal quantifiers. Regarding TFC, the effort required to find stability axioms seems to be similar to finding an inductive invariant. For TTC, all propagating properties were very simple. Finally, we noticed that, for more complex systems, TTC and TFC can lead to problems that are too large for the model-checker to answer in time (*e.g.* 1 h.)

9 Related Work

The usual way to check a safety property is to exhibit an inductive invariant for the system. The TEA tactic is completely automatic and can handle safety properties but remains quite limited. In our experiments, the TFC tactic showed to be as flexible as an invariant to prove safety properties. Finding stability axioms or an inductive invariant appear similar in difficulty. However, once found, checking an inductive invariant is quicker in computation time than checking the abstract system obtained with stability axioms. On the other hand, stability axioms allow to check complex temporal properties.

Regarding liveness properties, important approaches are based on exhibiting a variant or using the Liveness-to-Safety reduction method proposed in [19]. For the simple examples done with TEA, such methods would allow to prove the properties with little efforts, if done right, but are not fully automatic contrary to the TEA tactic. In both case the computation time is really low.

With the TTC approach, we do not need to exhibit any sort of invariant and the propagating property to exhibit has always been straightforward. To our knowledge there is no easiest method to prove some of the examples we presented. For example, the liveness property of the leader election protocol requires to exhibit a variant and an invariant and both are harder to exhibit than the propagating property. The Liveness-to-Safety reduction method also applies here, but it requires to find an invariant on the system obtained by reduction, as well as finding an axiomatization of the reflexive-transitive closure preserving the Liveness property (while this axiomatization is embedded in TTC tactic). However, despite being more immediate in our examples, the TTC tactic is less flexible than these two alternatives since it applies for liveness properties based on the reflexive-transitive closure.

Our approach can also be compared with the specification of parameterized systems. Cubicle [9, 10, 7, 8] is an SMT-based model-checker for the verification of safety properties on parameterized systems. Cubicle is efficient for challenging systems but, contrary to our techniques, it enforces strict syntactic constraints on guards and on the checked property. Others techniques based on labelled proof systems have also been proposed [2]. In [15], the safety of the TLB Shootdown algorithm is proved using such a technique. The user must exhibit the correct invariant for the proof system to conclude; while the TEA tactic is automatic. Also, some methods, such as invisible invariants [25], rely on finding automatically a candidate for being an inductive invariant and then checking if this is the case without needing any input from the user. Such an approach is automatic and efficient but only applies to Bounded-Data Parameterized Systems while our methods applies to a wider context. While most work on parameterized systems focuses on safety properties, [11] addresses liveness properties, but remains essentially theoretical. We remark that the techniques mainly used for parameterized systems are mostly orthogonal to those presented in this paper, and a combination of both could be fruitful.

10 Conclusion

We devised three original, sound (but incomplete) transformations, that allow to check that a state machine specification, expressed in a rather expressive fragment of $\text{FOLTL}^*_=$, enjoys a *temporal* property, expressed in the same setting, whatever the bounds on domains (associated with sorts) are. The transformations were proved correct in Coq. We evaluated our approach on several case studies and found that the transformations were effective and, for the semi-automatic ones, demanded an effort comparable to other approaches. A drawback is that the computed bounds can sometimes grow too much for model-checking to be feasible with the back-end tools we used. Notice that our approach is orthogonal to the main other approaches (for instance, inference of invariants) and could certainly be combined with some of them. Once a universally quantified inductive invariant Inv is found, such a combination would be possible by adding an axiom of the form $\mathbf{G}\,\mathrm{Inv}$ to our abtract specification. This refines the abstraction while fitting in both LTR and Geneva. This is left for future work.

Acknowledgements

We thank the article and artifact anonymous reviewers for their remarks that helped improve this work, as well as Nuno Macedo for his technical assistance.

References

1. Abrial, J.R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press, USA, 1st edn. (2010). https://doi.org/10.1017/cbo9781139195881
2. Arons, T., Pnueli, A., Ruah, S., Xu, Y., Zuck, L.: Parameterized verification with automatically computed inductive assertions? In: International Conference on Computer Aided Verification. pp. 221–234. Springer (2001). https://doi.org/10.1007/3-540-44585-4_19
3. Black, D.L., Rashid, R.F., Golub, D.B., Hill, C.R.: Translation lookaside buffer consistency: A software approach. ACM SIGARCH Computer Architecture News **17**(2), 113–122 (1989). https://doi.org/10.1145/68182.68193
4. Brunel, J., Chemouil, D., Cunha, A., Macedo, N.: The Electrum Analyzer: Model Checking Relational First-Order Temporal Specifications. In: 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE '18). Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ACM Press, Montpellier, France (Sep 2018). https://doi.org/10.1145/3238147.3240475
5. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuxmv symbolic model checker. In: Biere, A., Bloem, R. (eds.) Computer Aided Verification. pp. 334–342. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_22
6. Chang, E., Roberts, R.: An improved algorithm for decentralized extrema-finding in circular configurations of processes. Communications of the ACM **22**(5), 281–283 (1979). https://doi.org/10.1145/359104.359108
7. Conchon, S., Declerck, D., Zaïdi, F.: Cubicle-$\mathcal{W}$: Parameterized model checking on weak memory. In: International Joint Conference on Automated Reasoning. pp. 152–160. Springer (2018). https://doi.org/10.1007/978-3-319-94205-6_11
8. Conchon, S., Declerck, D., Zaïdi, F.: Parameterized model checking on the tso weak memory model. Journal of Automated Reasoning **64**(7), 1307–1330 (2020). https://doi.org/10.1007/s10817-020-09565-w
9. Conchon, S., Goel, A., Krstić, S., Mebsout, A., Zaïdi, F.: Cubicle: A parallel smt-based model checker for parameterized systems. In: International Conference on Computer Aided Verification. pp. 718–724. Springer (2012). https://doi.org/10.1007/978-3-642-31424-7_55
10. Conchon, S., Mebsout, A., Zaïdi, F.: Certificates for parameterized model checking. In: International Symposium on Formal Methods. pp. 126–142. Springer (2015). https://doi.org/10.1007/978-3-319-19249-9_9
11. Farzan, A., Kincaid, Z., Podelski, A.: Proving liveness of parameterized programs. In: 2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). pp. 1–12 (2016). https://doi.org/10.1145/2933575.2935310
12. Hawblitzel, C., Howell, J., Kapritsos, M., Lorch, J.R., Parno, B., Roberts, M.L., Setty, S., Zill, B.: Ironfleet: proving practical distributed systems correct. In: Proceedings of the 25th Symposium on Operating Systems Principles. pp. 1–17 (2015). https://doi.org/10.1145/2815400.2815428
13. Hodkinson, I., Wolter, F., Zakharyaschev, M.: Decidable fragments of first-order temporal logics. Annals of Pure and Applied logic **106**(1-3), 85–134 (2000). https://doi.org/10.1016/s0168-0072(00)00018-x
14. Hodkinson, I., Wolter, F., Zakharyaschev, M.: Monodic fragments of first-order temporal logics: 2000–2001 a.d. In: Logic for Programming, Artificial Intelligence,

and Reasoning. pp. 1–23. Springer (2001). https://doi.org/10.1007/3-540-45653-8_1

15. Hoenicke, J., Majumdar, R., Podelski, A.: Thread modularity at many levels: a pearl in compositional verification. ACM SIGPLAN Notices **52**(1), 473–485 (2017). https://doi.org/10.1145/3009837.3009893
16. Kuperberg, D., Brunel, J., Chemouil, D.: On finite domains in first-order linear temporal logic. In: Automated Technology for Verification and Analysis. pp. 211–226. Springer (2016). https://doi.org/10.1007/978-3-319-46520-3_14
17. Lamport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley Professional (2002)
18. Padon, O.: Deductive Verification of Distributed Protocols in First-Order Logic. Ph.D. thesis, Ph. D. Dissertation. Tel Aviv University (2018)
19. Padon, O., Hoenicke, J., Losa, G., Podelski, A., Sagiv, M., Shoham, S.: Reducing liveness to safety in first-order logic. Proceedings of the ACM Conference on Principles of Programming Languages (POPL) **2**, 26 (2017). https://doi.org/10.1145/3158114
20. Padon, O., Losa, G., Sagiv, M., Shoham, S.: Paxos made epr: decidable reasoning about distributed protocols. Proceedings of the ACM on Programming Languages **1**(OOPSLA), 108 (2017). https://doi.org/10.1145/3140568
21. Padon, O., McMillan, K.L., Panda, A., Sagiv, M., Shoham, S.: Ivy: safety verification by interactive generalization. ACM SIGPLAN Notices **51**(6), 614–630 (2016). https://doi.org/10.1145/2980983.2908118
22. Peyras, Q., Bodeveix, J.P., Brunel, J., Chemouil, D.: Cervino prototype, Coq formalization and Benchmarks (CAV 2021 artifact) (Apr 2021). https://doi.org/10.5281/zenodo.4725675
23. Peyras, Q., Brunel, J., Chemouil, D.: A bounded domain property for an expressive fragment of first-order linear temporal logic. In: Gamper, J., Pinchinat, S., Sciavicco, G. (eds.) 26th International Symposium on Temporal Representation and Reasoning, TIME 2019, October 16-19, 2019, Málaga, Spain. LIPIcs, vol. 147, pp. 15:1–15:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPIcs.TIME.2019.15
24. Peyras, Q., Brunel, J., Chemouil, D.: A decidable and expressive fragment of many-sorted first-order linear temporal logic. Information and Computation p. 104641 (2020). https://doi.org/10.1016/j.ic.2020.104641
25. Pnueli, A., Ruah, S., Zuck, L.: Automatic deductive verification with invisible invariants. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. pp. 82–97. Springer (2001). https://doi.org/10.1007/3-540-45319-9_7
26. Shapiro, M., Preguiça, N., Baquero, C., Zawirski, M.: A comprehensive study of convergent and commutative replicated data types. Ph.D. thesis, Inria–Centre Paris-Rocquencourt; INRIA (2011)
27. Wilcox, J.R., Woos, D., Panchekha, P., Tatlock, Z., Wang, X., Ernst, M.D., Anderson, T.: Verdi: a framework for implementing and formally verifying distributed systems. In: Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 357–368 (2015). https://doi.org/10.1145/2737924.2737958