UNIVERSITÀ DEGLI STUDI DI MILANO

Dipartimento di Scienze dell'Informazione

RAPPORTO INTERNO N° 318-08

Towards SMT Model Checking of Array-based Systems

Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, Daniele Zucchelli

Towards SMT Model Checking of Array-based Systems

Silvio Ghilardi[1], Enrica Nicolini[2], Silvio Ranise[1,2], and Daniele Zucchelli[1]
[1]Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano (Italy)
[2]LORIA & INRIA-Lorraine, Nancy (France)

April 20, 2008

Abstract

We introduce the notion of array-based system as a suitable abstraction of infinite state systems such as broadcast protocols or sorting programs. By using a class of quantified first-order formulae to symbolically represent array-based systems, we propose methods to check safety (invariance) and liveness (recurrence) properties on top of Satisfiability Modulo Theories solvers. We find hypotheses under which the verification procedures for such properties can be fully mechanized.

Contents

1 Introduction

Model checking of infinite state systems manipulating arrays – e.g., broadcast protocols, lossy channel systems, or sorting programs – is a hot topic in verification. The key problem is to verify the correctness of such systems regardless of the number of elements (processes, data, or integers) stored in the array, called *uniform verification problem*. In this paper, we propose *array-based systems* as a suitable abstraction of broadcast protocols, lossy channel systems, or, more in general, programs manipulating arrays (Section 3). The notion of array-based system is parametric with respect to a theory of indexes (which, for parameterized systems, specify the topology of processes) and a theory of elements (which, again for parameterized systems, specify the data manipulated by the system). Then (Section 3.1), we show how states and transitions of a large class of array-based systems can be symbolically represented by a class of *quantified* first-order formulae whose satisfiability problem is decidable under reasonable hypotheses on the theories of indexes and elements (Section 4). We also sketch how to extend the *lazy Satisfiability Modulo Theories* (SMT) techniques [26] by a suitable *instantiation strategy* to handle universally quantified variables (over indexes) so as to implement the satisfiability procedure for the class of quantified formulae under consideration (Figure 1). The capability to handle a (limited form) of universal quantification is crucial to reduce entailment between formulae representing states to satisfiability of a formula in the class of quantified formulae previously defined. This observation together with closure under pre-image computation of the kind of formulae representing unsafe states allows us to implement a *backward* reachability procedure (see, e.g., [19], or also [25] for a declarative approach) for checking safety properties of array-based systems (Section 5). In general, the procedure may not terminate; but, under some additional assumptions on the theory of elements, we are able to prove termination by revisiting the notion of configuration (see, e.g., [1]) in first-order model-theory (Section 5.2). Finally (Section 6), we show the flexibility of our approach by studying the problem of checking the class of liveness properties called *recurrences* [21]. We devise deductive methods to check such properties based on the synthesis of so-called progress conditions at quantifier free level and we show how to fully mechanize our technique under the same hypotheses for the termination of the backward reachability procedure.

All proofs have been put in Appendix A. Concerning examples, our framework covers all examples discussed in [2, 3];[1] in addition, we have further applications (e.g. to sorting algorithms), because our approach is parametric with respect to the theory describing process 'topology' (we can arrange processes as a set, as a linear order, as a tree, as a graph, etc.). Examples of various kinds are all discussed in the final Section 7, where it is also shown how

[1] For some of them, we need the approximated transition trick introduced in these papers.

to recover well-known decidability results for broadcast protocols and lossy channels systems as corollaries of our Theorem 5.7.

2 Formal Preliminaries

We assume the usual first-order syntactic notions of signature, term, formula, quantifier free formula, and so on; equality is always included in our signatures. If $\underline{x}$ is a finite set of variables and Σ is a signature, by a $\Sigma(\underline{x})$-term, -formula, etc. we mean a term, formula, etc. in which at most the $\underline{x}$ occur free (notations like $t(\underline{x})$, $\varphi(\underline{x})$ emphasize the fact that the term t or the formula φ in in fact a $\Sigma(\underline{x})$-term or a $\Sigma(\underline{x})$-formula, respectively). The notions of interpretation, satisfiability, validity, and logical consequence are also the standard ones; when we speak about satisfiability (resp. validity) of a formula containing free variables, we mean satisfiability of its existential (resp. universal) closure. If $\mathcal{M} = (M, \int)$ is a Σ-structure, a Σ-*substructure* of $\mathcal{M}$ is a Σ-structure having as domain a subset of M which is closed under the operations of Σ (in a Σ-substructure, moreover, the interpretation of the symbols of Σ is given by restriction). The Σ-structure *generated by a subset* X of M (which is not assumed now to be closed under the Σ-operations) is the smallest Σ-substructure of $\mathcal{M}$ whose domain contains X and, if this Σ-substructure coincides with the whole $\mathcal{M}$, we say that X *generates* $\mathcal{M}$. A Σ-*embedding* (or, simply, an embedding) between two Σ-structures $\mathcal{M} = (M, \int)$ and $\mathcal{N} = (N, \mathcal{J})$ is any mapping $\mu : M \longrightarrow N$ among the corresponding support sets which is an isomorphism between $\mathcal{M}$ and the Σ-substructure of $\mathcal{N}$ whose underlying domain is the image of μ (thus, in particular, for μ to be an embedding, the image of μ must be closed under the Σ-operations). A class $\mathcal{C}$ of structures is *closed under substructures* iff whenever $\mathcal{M} \in \mathcal{C}$ and $\mathcal{N}$ is (isomorphic to) a substructure of $\mathcal{M}$, then $\mathcal{N} \in \mathcal{C}$.

Contrary to previous papers of ours, like [17], we prefer to have here a more liberal notion of a theory, so we identify a *theory* T with a pair $(\Sigma, \mathcal{C})$, where Σ is a signature and $\mathcal{C}$ is a class of Σ-structures (the structures in $\mathcal{C}$ are called the *models* of T).

The notion of T-*satisfiability* of φ means satisfiability of φ in a Σ-structure from $\mathcal{C}$; similarly, T-*validity* of a sentence φ (noted $T \models \varphi$) means truth of φ in all $\mathcal{M} \in \mathcal{C}$. The *Satisfiability Modulo Theory* T, SMT(T), problem amounts to establish the T-satisfiability of an arbitrary first-order formula (hence possibly containing quantifiers), w.r.t. some background theory T. A *theory solver* for the theory T (T-solver) is any procedure capable of establishing whether any given finite conjunction of literals is T-satisfiable or not. The so-called lazy approach to solve SMT(T) problems for quantifier-free formulae consists of integrating a Boolean enumerator (usually based on a refinement of the DPLL algorithm) with a T-solver (see [22, 26] for an overview). Hence, by assuming the existence of a T-solver,

it is always possible to build a (lazy SMT) solver capable of checking the T-satisfiability of arbitrary Boolean combinations of atoms in T (or, equivalently, quantifier-free formulae). For efficiency, a T-solver is required to provide a more refined interface, such as returning a sub-set of a T-unsatisfiable input set of literals which is still T-unsatisfiable, called conflict set (again see [26] for details).

We say that T admits *quantifier elimination* iff for every formula $\varphi(\underline{x})$ one can compute a quantifier-free formula $\varphi'(\underline{x})$ which is T-equivalent to it (i.e. such that $T \models \forall \underline{x}\,(\varphi(\underline{x}) \leftrightarrow \varphi'(\underline{x}))$). Linear Arithmetics, Real Arithmetics, acyclic lists, and enumerated datatype theories (see below) admit elimination of quantifiers.

A theory $T = (\Sigma, \mathcal{C})$ is said to be *locally finite* iff Σ is finite and, for every finite set of variables $\underline{x}$, there are finitely many $\Sigma(\underline{x})$-terms $t_1, \ldots, t_{k_{\underline{x}}}$ such that for every further $\Sigma(\underline{x})$-term u, we have that $T \models u = t_i$ (for some $i \in \{1, \ldots, k_{\underline{x}}\}$). The terms $t_1, \ldots, t_{k_{\underline{x}}}$ are called $\Sigma(\underline{x})$-representative terms; if they are effectively computable from $\underline{x}$ (and t_i is computable from u), then T is said to be *effectively locally finite* (in the following, when we say 'locally finite', we in fact always mean 'effectively locally finite'). If Σ is finite and does not contain any function symbol (i.e. Σ is a purely relational signature), then any Σ-theory is effectively locally finite; other effectively locally finite theories are Boolean algebras and Linear Arithmetic modulo a fixed integer.

Let Σ be a finite signature; an *enumerated datatype theory* in Σ is the theory whose class of models contains only a single finite Σ-structure $\mathcal{M} = (M, \mathcal{I})$; we require $\mathcal{M}$ to have the additional property that for every $m \in M$ there is a constant $c \in \Sigma$ such that $c^{\mathcal{I}} = m$. It is easy to see that an enumerated datatype theory admits quantifier elimination (since $\exists x\, \varphi(x)$ is T-equivalent to $\varphi(c_1) \vee \cdots \vee \varphi(c_n)$, where $c_1, \ldots, c_n$ are interpreted as the finitely many elements of the enumerated datatype) and is effectively locally finite.

In the following, it is more natural to adopt a many-sorted language. All notions introduced above (such as term, formula, structure, and satisfiability) can be easily adapted to many-sorted logic, see e.g. Chapter XX of [13].

3 Array-based Systems and their Symbolic Representation

We develop a framework to state and solve model checking problems for safety and liveness of a particular class of infinite state systems by using deductive (more precisely, SMT) techniques. We focus on the class of systems whose state can be described by a finite collections of arrays, which we call *array-based* systems. As an example, consider parameterized systems, i.e. systems consisting of an arbitrary number of identical finite-state processes organized in a linear array: a state a of such a parameterized system can be seen as a state of an array-based

system (in the formal sense of Definitions 3.2, 3.3 below), where the indexes of the domain of the function assigned to the state variable a are the identifiers of the processes and the elements of a are the data describing one of the finitely many states of each process.

Array-based Systems. To develop our formal model, we introduce three theories. We use two mono-sorted theories $T_I = (\Sigma_I, \mathcal{C}_I)$ (of indexes) and $T_E = (\Sigma_E, \mathcal{C}_E)$ (of elements), whose only sort symbol are called `INDEX` and `ELEM`, respectively. T_I specifies the "topology" of the processes, e.g., in the case of parameterized systems informally discussed above, Σ_I contains only the equality symbol '=' and $\mathcal{C}_I$ is the class of all finite sets. Other interesting examples of T_I can be obtained by taking Σ_I to contain a binary predicate symbol R (besides =) and $\mathcal{C}_I$ to be the class of structures where R is interpreted as a total order, a graph, a forest, etc. For parameterized systems with finite-state processes, T_E can be the theory of an enumerated datatype. For the larger class of parameterized systems (e.g., the one considered in [2, 3]) admitting integer (or real) variables local to each process, T_E can be the theory whose class of models consists of a single structure (like real numbers under addition and/or ordering). Notice that in concrete applications, T_E has a single model (e.g. an enumerate datatype, or the structure of real/natural numbers under suitable operations and relations), whereas T_I has many models (e.g. it has as models all sets, all finite sets, all graphs, all finite graphs, etc.). The technical hypotheses we shall need in Sections 4 and 5 on T_I and T_E are fixed in the following:

Definition 3.1. An *index theory* $T_I = (\Sigma_I, \mathcal{C}_I)$ is a mono-sorted theory (let us call `INDEX` its sort) which is locally finite, closed under substructures and whose quantifier free fragment is decidable for T_I-satisfiability. An *element theory* $T_E = (\Sigma_E, \mathcal{C}_E)$ is a mono-sorted theory (let us call `ELEM` its sort) which admits quantifier elimination and whose quantifier free fragment is decidable for T_E-satisfiability.

One may wonder how restrictive are the assumptions of the above definition: it turns out that they are very light (for instance, they do not rule out any of the examples considered in [2, 3]). In fact, quantifier elimination holds for common datatype theories (integers, reals, enumerated datatypes, etc.) and the hypotheses on index theory are satisfied for most process 'topologies' (a case in which they fail is when processes are arranged in a ring, like in the 'dining philosophers' example: such rings require a unary function symbol to be formalized and the bijectivity constraint to be imposed on it is insufficient to make the theory locally finite).

The third theory $A_I^E = (\Sigma, \mathcal{C})$ we need is obtained by combining an index theory T_I and an element theory T_E as follows. First, A_I^E has three sort symbols: `INDEX`, `ELEM`, and `ARRAY`; the

signature Σ contains all the symbols in the disjoint union $\Sigma_I \cup \Sigma_E$ and a further binary function symbol *apply* of sort $\texttt{ARRAY} \times \texttt{INDEX} \longrightarrow \texttt{ELEM}$. (In the following, we abbreviate $apply(a, i)$ with $a[i]$, for a term of sort ARRAY and i term of sort INDEX.) Second, a three-sorted structure $\mathcal{M} = (\texttt{INDEX}^{\mathcal{M}}, \texttt{ELEM}^{\mathcal{M}}, \texttt{ARRAY}^{\mathcal{M}}, \int)$ is in the class $\mathcal{C}$ iff $\texttt{ARRAY}^{\mathcal{M}}$ is the set of (total) functions from $\texttt{INDEX}^{\mathcal{M}}$ to $\texttt{ELEM}^{\mathcal{M}}$, the function symbol *apply* is interpreted as function application (i.e. as the standard reading operation for arrays), and $(\texttt{INDEX}^{\mathcal{M}}, \int_{|\Sigma_I}), (\texttt{ELEM}^{\mathcal{M}}, \int_{|\Sigma_E})$ are models of T_I and T_E, respectively – here $\int_{|\Sigma_I}, \int_{|\Sigma_E}$ are the restriction of $\int$ to the symbols of Σ_I, Σ_E. (In the following, we use the notations $\mathcal{M}_I$ and $\mathcal{M}_E$ for $(\texttt{INDEX}^{\mathcal{M}}, \int_{|\Sigma_I})$ and $(\texttt{ELEM}^{\mathcal{M}}, \int_{|\Sigma_E})$, respectively.) If the model $\mathcal{M}$ of A_I^E is such that $\texttt{INDEX}^{\mathcal{M}}$ is a finite set, then $\mathcal{M}$ is called a *finite index model*.

For the remaining part of the paper, **we fix an index theory $T_I = (\Sigma_I, \mathcal{C}_I)$ and an element theory, $T_E = (\Sigma_E, \mathcal{C}_E)$** (we also let $A_I^E = (\Sigma, \mathcal{C})$ be the corresponding combined theory).

Once the theories constraining indexes and elements are fixed, we need the notions of state, initial state and state transition to complete our picture. In a symbolic setting, these are provided by the following definitions:

Definition 3.2. An *array-based (transition) system (for (T_I, T_E))* is a triple $\mathcal{S} = (\underline{a}, I, \tau)$ where

- $\underline{a}$ is a tuple of variables of sort ARRAY (these are the *state variables*);
- $I(\underline{a})$ is a $\Sigma(\underline{a})$-formula (this is the *initial state formula*);
- $\tau(\underline{a}, \underline{a}')$ is a $\Sigma(\underline{a}, \underline{a}')$-formula – here $\underline{a}'$ is a renamed copy of the tuple $\underline{a}$ (this is the *transition* formula).

Definition 3.3. Let $\mathcal{S} = (\underline{a}, I, \tau)$ be an array-based transition system. Given a model $\mathcal{M}$ of the combined theory A_I^E, an *$\mathcal{M}$-state* (or, simply, a state) of $\mathcal{S}$ is an assignment mapping the state variables $\underline{a}$ to total functions $\underline{s}$ from the domain of $\mathcal{M}_I$ to the domain of $\mathcal{M}_E$. A *run* of the array-based system is a (possibly infinite) sequence $\underline{s}_0, \underline{s}_1, ...$ of states such that[2] $\mathcal{M} \models I(\underline{s}_0)$ and $\mathcal{M} \models \tau(\underline{s}_k, \underline{s}_{k+1})$ for $k \geq 0$.

For simplicity, below, we assume that the tuple of array state variables is a single variable a: all definitions and results of the paper can be easily generalized to the case of finitely many array variables (one can also prove that the limitation to a single array variable is not formally restrictive, provided T_E is allowed to be many sorted as well – see Section 7 for a discussion on this topic and also on how to model global variables).

[2]Notations like $\mathcal{M} \models I(\underline{s}_0)$ means that the formula $I(\underline{a})$ is true in $\mathcal{M}$ under the assignment mapping the $\underline{a}$'s to the $\underline{s}$'s.

3.1 Symbolic Representation of States and Transitions

The next step is to identify suitable syntactic restrictions for the formulae I, τ appearing in the definition of an array-based system. Preliminarily, we introduce some notational conventions that alleviate the burden of writing and understanding the various formulae for states and transitions: $d, e, \ldots$ range over variables of sort `ELEM`, $a, b \ldots$ over variables of sort `ARRAY`, $i, j, k, \ldots$ over variables of sort `INDEX`, and $\alpha, \beta, \ldots$ over variables of either sort `ELEM` or sort `INDEX`. An underlined variable name abbreviates a tuple of variables of unspecified (but finite) length; by subscripting with an integer an underlined variable name, we indicate the corresponding component of the tuple (e.g., $\underline{i}$ may abbreviate $i_1, \ldots, i_n$ and $\underline{i}_k$ indicates i_k, for $1 \leq k \leq n$). We also use $a[\underline{i}]$ to abbreviate the tuple of terms $a[i_1], \ldots, a[i_n]$. If $\underline{j}$ and $\underline{i}$ are tuples with the same length n, then $\underline{i} = \underline{j}$ abbreviates $\bigwedge_{k=1}^{n}(\underline{i}_k = \underline{j}_k)$. Possibly sub/super-scripted expressions of the forms $\phi(\underline{\alpha}), \psi(\underline{\alpha}), \ldots$ (with sub-/super-scripts) always denote **quantifier-free $(\Sigma_I \cup \Sigma_E)$-formulae in which at most the variables $\underline{\alpha}$ occur** (notice in particular that no array variable and no apply constructor $a[i]$ can occur here). Also, $\phi(\underline{\alpha}, \underline{t}/\underline{\beta})$ (or simply $\phi(\underline{\alpha}, \underline{t})$) abbreviates the substitution of the terms $\underline{t}$ for the variables $\underline{\beta}$ (now apply constructors may appear in t). Thus, for instance, when we write $\phi(\underline{i}, a[\underline{i}])$, we mean the formula obtained by the replacements $\underline{e} \mapsto a[\underline{i}]$ in the quantifier free formula $\phi(\underline{i}, \underline{e})$ (the latter contains at most the element variables $\underline{e}$ and at most the index variables $\underline{i}$).

We are now ready to introduce suitably restricted classes of formulae for representing states and transitions of array-based systems.

States. An **$\exists^I$-formula** is a formula of the form $\exists \underline{i}\, \phi(\underline{i}, a[\underline{i}])$: such a formula may be used to model sets of *unsafe* states of parameterized systems, such as violations of mutual exclusion:

$$\exists i\, \exists j\, (i \neq j \wedge a[i] = \texttt{critical} \wedge a[j] = \texttt{critical}),$$

where `critical` is a constant symbol of sort `ELEM` in an enumerated datatype theory.

A **$\forall^I$-formula** is a formula of the form $\forall \underline{i}\, \phi(\underline{i}, a[\underline{i}])$: this is logically equivalent to the negation of an $\exists^I$-formula and may be used to model *initial* sets of states of parameterized systems, such as "all processes are in a given state":

$$\forall i\, (a[i] = \texttt{idle}),$$

where `idle` is a constant symbol of sort `ELEM` in an enumerated datatype theory. We mention that $\forall^I$-formulae can also be used to model *invariant* assertions (like "every process in critical section must have a nonzero ticket").

Transitions. The intuition underlying the class of formulae representing transitions of array-based systems can be found by analyzing the structure of transitions of parameterized systems. In such systems, a transition has typically two components. The *local* component specifies the transitions of a given fixed number of processes; e.g., one of the transitions of the Bakery protocol for mutual exclusion requires that a process with process identifier i may return to the state `idle` upon exiting the `critical` section. The *global* component specifies the transitions done by all the other process in the array as a reaction to those taken by the processes involved in the local component; e.g., continuing with the transition of the Bakery protocol, any other process $j \neq i$ persists in the same state after process i is returned to state `idle` after having been in state `critical`. The transition of the Bakery protocol can be specified by the following formula:

$$\exists i \left(\begin{array}{c} (a[i] = \texttt{critical} \wedge a'[i] = \texttt{idle}) \wedge \\ \forall j \, \big(\; (j = i \wedge a'[j] = a'[i]) \vee (j \neq i \wedge a'[j] = a[j]) \; \big) \end{array} \right).$$

If we abbreviate the first conjunct with $\phi_L(i, a[i], a'[i])$ and the quantifier-free part of the second with $\phi_G(j, a[j], a'[j])$, the above formula can be written as

$$\exists i \, (\phi_L(i, a[i], a'[i]) \wedge \forall j \, \phi_G(i, a[i], a'[i], j, a[j], a'[j])),$$

where ϕ_L is called the *local component formula* and ϕ_G is the *global component formula*. We generalize this format by allowing a tuple $\underline{i}$ of existentially quantified variables (instead of a single variable i). Thus we are lead to the following

Definition 3.4. Suppose we are given formulae $\phi_L(\underline{i}, \underline{d}, \underline{d}')$ and $\phi_G(\underline{i}, \underline{d}, \underline{d}', j, e, e')$, where ϕ_G satisfies the following *seriality* requirement:

$$A_I^E \models \forall \underline{i} \, \forall \underline{d} \, \forall \underline{d}' \, \forall j \, \forall e \, \exists e' \, \phi_G(\underline{i}, \underline{d}, \underline{d}', j, e, e'). \tag{1}$$

The ***T-formula*** with local component ϕ_L and global component ϕ_G is the formula

$$\exists \underline{i} \, (\phi_L(\underline{i}, a[\underline{i}], a'[\underline{i}]) \wedge \mathit{Update}_G(\underline{i}, a, a')), \tag{2}$$

where we used the explicit definition (i.e. the abbreviation)

$$\mathit{Update}_G(\underline{i}, a, a') \; :\Leftrightarrow \forall j \, \phi_G(\underline{i}, a[\underline{i}], a'[\underline{i}], j, a[j], a'[j]). \tag{3}$$

Recall that, according to our conventions, the local component ϕ_L and the global component ϕ_G are quantifier-free $(\Sigma_I \cup \Sigma_E)$-formulae. In $\phi_G(\underline{i}, \underline{d}, \underline{d}', j, e, e')$ the variables $\underline{i}, \underline{d}, \underline{d}'$ are called *side* parameters (these variables are instantiated in (2)-(3) by $\underline{i}, a[\underline{i}], a'[\underline{i}]$, respectively) and the variables j, e, e' are called *proper* parameters (these variables are instantiated

in (2)-(3) by $j, a[j], a'[j]$, respectively). The intuition underlying the formula $Update_G(\underline{i}, a, a')$ defined by (3) is that the array a' is obtained as *a global update of the array a according to* ϕ_G. In fact, in (3), the proper parameters of ϕ_G are instantiated as $j, a[j], a'[j]$, i.e. as the index to be updated, the old, and the new content of that index. Notice also that this updating is largely non-deterministic, because a T-formula like (2) does not univocally characterize the system evolution: this is revealed by the fact that ϕ_G is not a definable function, and by the fact that the side parameters $\underline{d}'$ that may occur in ϕ_G are instantiated as the updated values $a'[\underline{i}]$ (this circularity makes sense only if we view the T-formula (2) as expressing a *constraint* – not necessarily an assignment – for the updated array a').

We say that ϕ_G is *functional* if it satisfies the further requirement

$$A_I^E \models \forall \underline{i}\, \forall \underline{d}\, \forall \underline{d}'\, \forall j\, \forall e\, \forall e'\, \forall e''\, (\phi_G(\underline{i}, \underline{d}, \underline{d}', j, e, e') \wedge \phi_G(\underline{i}, \underline{d}, \underline{d}', j, e, e'') \rightarrow e' = e''). \qquad (4)$$

Many global component formulae arising in the applications are indeed functional; for example, it is easy to see that the transition relation of the Bakery formalized above is functional. In Section 7 we shall see significant examples, where serial but not functional transitions naturally arise.

In the remaining part of the paper, **we fix an array-based system $\mathcal{S} = (a, I, \tau)$, in which the initial formula I is a $\forall^I$-formula and the transition formula τ is a *disjunction of* T-formulae**.

4 Symbolic Representation and SMT Solving

To make effective use of our framework, we need to be able to check whether certain requirements are met by a given array-based system. In other words, we need a powerful reasoning engine: it turns out that A_I^E-satisfiability of $\exists^{A,I}\forall^I$-sentences introduced below is just what we need.[3]

Theorem 4.1. *The A_I^E-satisfiability of $\exists^{A,I}\forall^I$-sentences, i.e. of sentences of the kind*

$$\exists a_1 \cdots \exists a_n\, \exists \underline{i}\, \forall \underline{j}\, \psi(\underline{i}, \underline{j}, a_1[\underline{i}], \ldots, a_n[\underline{i}], a_1[\underline{j}], \ldots, a_n[\underline{j}]), \qquad (5)$$

is decidable.

We leave to Appendix A the detailed proof of the above Theorem, here we focus on a discussion oriented to the employment of SMT-solvers in the decision procedure: Figure 1 depicts an SMT-based decision procedure for $\exists^{A,I}\forall^I$-sentences (in the simplified case in which only one variable of sort `ARRAY` occurs).

[3]Decidability of A_I^E-satisfiability of $\exists^{A,I}\forall^I$-sentences plays in this paper a role similar to the Σ_1^0-decidability result employed in [7] (our decidability result is however peculiar, because we are in a declarative multi-sorted context, with two input theories, namely T_I and T_E).

function A_I^E-check($\exists a\ \exists \underline{i}\ \forall \underline{j}\ \psi(\underline{i}, \underline{j}, a[\underline{i}], a[\underline{j}])$)
  $\underline{t} \longleftarrow$ compute-reps$_{T_I}(\underline{i})$; $\phi \longleftarrow \underline{t} = \underline{l}$; $Atoms \longleftarrow \mathsf{IE}(\underline{l})$
  for each substitution σ mapping the $\underline{j}$ into the $\underline{t}$'s **do**
    $\phi' \longleftarrow$ purify($\psi(\underline{i}, \underline{j}\sigma, a[\underline{i}], a[\underline{j}\sigma])$); $\phi \longleftarrow \phi \wedge \phi'$; $Atoms \longleftarrow Atoms \cup$ atoms(ϕ');
  end for each
  while Bool(ϕ) = sat **do**
    $\beta_I \wedge \beta_E \wedge \eta_I \longleftarrow$ pick-assignment($Atoms, \phi$)
    $(\rho_I, \pi_I) \longleftarrow T_I$-solver($\beta_I \wedge \eta_I$)
    $(\rho_E, \pi_E) \longleftarrow T_E$-solver($\beta_E \wedge \bigwedge_{\underline{l}_{s_1} = \underline{l}_{s_2} \in \eta_I} (\underline{e}_{s_1} = \underline{e}_{s_2})$)
    if ρ_I = sat **and** ρ_E = sat **then return** sat
    if ρ_I = unsat **then** $\phi \longleftarrow \phi \wedge \neg \pi_I$
    if ρ_E = unsat **then** $\phi \longleftarrow \phi \wedge \neg \pi_E$
  end while
  return unsat
end

Figure 1: The SMT-based decision procedure for $\exists^{A,I}\forall^I$-sentences

The set $\underline{t}$ of representative terms over $\underline{i}$ is computed by compute-reps$_{T_I}$ (to simplify the matter, we can freely assume $\underline{t} \supseteq \underline{i}$). This function is guaranteed to exist since T_I is effectively locally finite (for example, when Σ_I contains no function symbol, compute-reps$_{T_I}$ is the identity function). In order to purify formulae, we shall need below fresh index variables $\underline{l}$ abstracting the $\underline{t}$ and fresh element variables $\underline{e}$ abstracting the terms $a[\underline{l}]$: the conjunction of the 'defining equations' $\underline{t} = \underline{l}$ is stored as the formula ϕ, whereas the further defining equations $a[\underline{l}] = \underline{e}$ (not to be sent to the Boolean solver) will be taken into account inside the second loop body.

Then, the first loop is entered where we instantiate in all possible ways the universally quantified index variables $\underline{j}$ with the representative terms in the set $\underline{t}$. For every such substitution σ, the formula $\psi(\underline{i}, \underline{j}\sigma, a[\underline{i}], a[\underline{j}\sigma])$ is purified by replacing the terms $a[\underline{t}]$ with the element variables $\underline{e}$; after such purification, the resulting formula is added as a new conjunct to ϕ. Notice that this ϕ is a quantifier-free formula whose atoms are pure, i.e. they are either Σ_I- or Σ_E-atoms. The function $\mathsf{IE}(\underline{l})$ returns the set of all possible equalities among the index variables $\underline{l}$. Such equalities are added to the set of atoms occurring in ϕ as the T_I- and T_E-solvers need to take into account an equivalence relation over the index variables $\underline{l}$ so as to synchronize.

We can now enter the second (and main) loop: the Boolean solver, by invoking the interface function pick-assignment inside the loop, generates an assignment over the set of

atoms occurring in ϕ and $\mathsf{IE}(\underline{l})$. The loop is exited when ϕ is checked unsatisfiable (test of the while). The T_I-solver checks for unsatisfiability the set β_I of Σ_I-literals in the Boolean assignment and the literals of the possible partition η_I on $\underline{l}$. Then, only the equalities $\bigwedge\{\underline{e}_{s_1} = \underline{e}_{s_2} \mid (\underline{l}_{s_1} = \underline{l}_{s_2}) \in \eta_I\}$ (and not also inequalities) among the purification variables $\underline{e}$ induced by the partition η_I on $\underline{l}$ are passed to the T_E-solver together with the set β_E of Σ_E-literals in the Boolean assignment (this is to take into account the congruence of $a[\underline{l}] = \underline{e}$). If both solvers detect satisfiability, then the loop is exited and A_I^E-check also returns satisfiability. Otherwise (i.e. if at least one of the solvers returns unsat), the negation of the computed conflict set (namely, π_I or π_E) is conjoined to ϕ so as to refine the Boolean abstraction and the loop is resumed. If in the second loop, satisfiability is never detected for all considered Boolean assignments, then A_I^E-check returns unsatisfiability. The second loop can be seen as a refinement of the Delayed Theory Combination technique [8].

The critical point of the above procedure is the fact that the first loop may produce a very large ϕ: in fact, even in the most favourable case in which compute-reps$_{T_I}$ is the identity function, the purified problem passed to the solvers of the second loop has exponential size. This is consequently the dominating cost of the whole procedure. In fact, in order to better evaluate the complexity, we can reformulate the second loop as follows (see the proof of Theorem 4.1 in the Appendix A): guess in advance a partition η_I (this is just a *polynomial* guess) and simply check $\phi_I \wedge \eta_I$ for T_I-satisfiability and $\phi_E \wedge \bigwedge_{\underline{l}_{s_1} = \underline{l}_{s_2} \in \eta_I} (\underline{e}_{s_1} = \underline{e}_{s_2})$ for T_E-satisfiability (here we assumed that ϕ is splitted into the pure components $\phi_I \wedge \phi_E$). These T_E-satisfiability and T_I-satisfiability problems have both exponential size, because $\phi_I \wedge \phi_E$ is of exponential size (in the most favourable case in which compute-reps$_{T_I}$ is the identity function): this means that if both T_I, T_E-satisfiability are in P (resp. NP, PSPACE, EXPTIME), then the whole complexity of our procedure is one exponential higher, i.e. it is EXPTIME (resp. NEXPTIME, EXPSPACE, 2EXPTIME, etc.). This is because the total number of guesses is exponential and each guess leads to an expsize instance of a satisfiability problem modulo T_I and T_E.

To improve performances, an *incremental* approach is desirable: in the incremental approach, the second loop may be entered before all possible substitution have been examined. If the second loop returns unsat, one can exit the whole algorithm and only in case it returns sat further substitutions (producing new conjuncts for ϕ) are taken into consideration. Since when A_I^E-check is called in conclusive steps of our model checking algorithms (for fixpoint, safety, or progress checks), the expected answer is unsat, this incremental strategy may be considerably convenient. Other promising suggestions for reducing the number of instantiations (in the case of particular index and elements theories) come from recent decision procedures for fragments of theories of arrays, see [9, 18].

5 Safety Model Checking

Checking safety properties means checking that a set of 'bad states' (e.g., of states violating the mutual exclusion property) cannot be reached by a system. This can be formalized in our settings as follows:

Definition 5.1. Let $K(a)$ be an $\exists^I$-formula (whose role is that of symbolically representing the set of bad states). The *safety model checking problem* for K is the problem of deciding whether there is an $n \geq 0$ such that the formula

$$I(a_0) \wedge \tau(a_0, a_1) \wedge \cdots \wedge \tau(a_{n-1}, a_n) \wedge K(a_n) \tag{6}$$

is A_I^E-satisfiable. If such an n does not exist, K is *safe*; otherwise, it is *unsafe*.

Notice that the A_I^E-satisfiability of (6) means precisely the existence of a finite run leading from a state in I to a state in K.

5.1 Backward Reachability

If a bound for n in the safety model checking is known a priori, then safety can be decided by checking the A_I^E-satisfiability of suitable instances of (6) (this is possible because each formula (6) is equivalent to a $\exists^{A,I}\forall^I$-formula). Unfortunately, this is rarely the case and we must design a more refined method. In the following, we adapt algorithms that incrementally maintain a representation of the set of reachable states in an array-based system.

Definition 5.2. Let $K(a)$ be an $\exists^I$-formula. An $\mathcal{M}$-state s_0 is *backward reachable from K in n steps* iff there exists a sequence of $\mathcal{M}$-states $s_1, \ldots, s_n$ such that

$$\mathcal{M} \models \tau(s_0, s_1) \wedge \cdots \wedge \tau(s_{n-1}, s_n) \wedge K(s_n).$$

We say that s_0 is *backward reachable from K* iff it is backward reachable from K in n steps for some $n \geq 0$.

Before designing our backward reachability algorithm, we must give a symbolic representation of the set of backward reachable states. Preliminarily, notice that if $K(a)$ is an $\exists^I$-formula, the set of states which are backward reachable in one step from K can be represented as follows:

$$Pre(\tau, K) := \exists a' \, (\tau(a, a') \wedge K(a')). \tag{7}$$

Although $Pre(\tau, K)$ is not an $\exists^I$-formula anymore, we are capable of finding an equivalent one in the class:

function BReach(K)
  $i \longleftarrow 0$; $BR^0(\tau, K) \longleftarrow K$; $K^0 \longleftarrow K$
  if A_I^E-check$(BR^0(\tau, K) \wedge I) =$ sat **then return** unsafe
  repeat
    $K^{i+1} \longleftarrow$ Pre(τ, K^i)
    $BR^{i+1}(\tau, K) \longleftarrow BR^i(\tau, K) \vee K^{i+1}$
    if A_I^E-check$(BR^{i+1}(\tau, K) \wedge I) =$ sat **then return** unsafe
    else $i \longleftarrow i + 1$
  until A_I^E-check$(\neg(BR^{i+1}(\tau, K) \rightarrow BR^i(\tau, K)) =$ unsat
  return safe
end

Figure 2: Backward Reachability for Array-based Systems

Proposition 5.3. *Let $K(a)$ be an $\exists^I$-formula; then $Pre(\tau, K)$ is A_I^E-equivalent to an (effectively computable) $\exists^I$-formula $K'(a)$.*

The proof of this Proposition can be obtained by a tedious syntactic computation while using (1) together with the fact that T_E admits quantifier elimination.

Abstractly, the set of states that can be backward reachable from K can be recursively characterized as follows: (a) $Pre^0(\tau, K) := K$ and (b) $Pre^{n+1}(\tau, K) := Pre(\tau, Pre^n(\tau, K))$. The formula $BR^n(\tau, K) := \bigvee_{s=0}^n Pre^s(\tau, K)$ (having just one free variable of type `ARRAY`) symbolically represents the states that can be backward reached from K in n steps. The sequence of $BR^n(\tau, K)$ allows us to rephrase the safety model checking problem (Definition 5.1) using symbolic representations by saying that K *is safe iff the formulae* $I \wedge BR^n(\tau, K)$ *are all not* A_I^E*-satisfiable.* This still requires infinitely many tests, *unless there is an* n *such that the formula* $\neg(BR^{n+1}(\tau, K) \rightarrow BR^n(\tau, K))$ *is* A_I^E*-unsatisfiable.*[4] The key observation now is that both $I \wedge BR^n(\tau, K)$ and $\neg(BR^{n+1}(\tau, K) \rightarrow BR^{n+1}(\tau, K))$ can be easily transformed into $\exists^{A,I}\forall^I$-formulae so that their A_I^E-satisfiability is decidable and checked by the procedure A_I^E-check of Figure 1.

This discussion suggests the adaptation of a standard backward reachability algorithm (see, e.g., [19]) depicted in Figure 2, where Pre takes a formula, it computes *Pre* as defined in (7), and then applies the required syntactic manipulations to transform the resulting formula into an equivalent one according to Proposition 5.3.

[4]Notice that the A_I^E-unsatisfiability of $\neg(BR^n(\tau, K) \rightarrow BR^{n+1}(\tau, K))$ is obvious by definition. Hence, the A_I^E-unsatisfiability of $\neg(BR^{n+1}(\tau, K) \rightarrow BR^n(\tau, K))$ implies that $A_I^E \models BR^{n+1}(\tau, K) \leftrightarrow BR^n(\tau, K)$.

Theorem 5.4. *Let $K(a)$ be an $\exists^I$-formula; then the function* BReach *in Figure 2 semi-decides the safety model checking problem for K.*

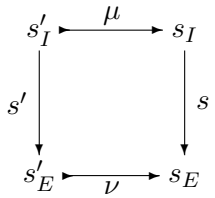
5.2 Termination of Backward Reachability

Indeed, the main problem with the semi-algorithm of Figure 2 is termination; in fact, it may not terminate when the system is safe. In the literature, termination of infinite state model checking is often obtained by defining a well-quasi-ordering (wqo – see, e.g., [1]) on the configurations of the system. Here, we adapt this approach to our setting, *by defining model-theoretically a notion of configuration and then introducing a suitable ordering on configurations*: once this is done, it will be immediate to prove that termination follows whenever the ordering among configurations is a wqo.

An *A_I^E-configuration* (or, briefly, a configuration) is an $\mathcal{M}$-state in a finite index model $\mathcal{M}$ of A_I^E: a configuration is denoted as $(s, \mathcal{M})$, or simply as s, leaving $\mathcal{M}$ implicit. Moreover, we associate a Σ_I-structure s_I and a Σ_E-structure s_E with an A_I^E-configuration $(s, \mathcal{M})$ as follows: the Σ_I-structure s_I is simply the finite structure $\mathcal{M}_I$, whereas s_E is the Σ_E-substructure of $\mathcal{M}_E$ generated by the image of s (in other words, if $\texttt{INDEX}^{\mathcal{M}} = \{c_1, \ldots, c_k\}$, then s_E is generated by $\{s(c_1), \ldots, s(c_k)\}$).

We remind few definitions about preorders. A preorder $(P, \leq)$ is a set endowed with a reflexive and transitive relation; an *upset* of such a preorder is a subset $U \subseteq P$ such that ($p \in U$ and $p \leq q$ imply $q \in U$). An upset U is *finitely generated* iff it is a finite union of cones, where a *cone* is an upset of the form $\uparrow p = \{q \in P \mid p \leq q\}$ for some $p \in P$. A preorder $(P, \leq)$ is a *well-quasi-ordering* (wqo) iff every upset of P is finitely generated (this is equivalent to the standard definition, see Appendix A). We define now a preorder among our configurations:

Definition 5.5. Let s, s' be configurations; $s' \leq s$ holds iff there are a Σ_I-embedding $\mu : s'_I \longrightarrow s_I$ and a Σ_E-embedding $\nu : s'_E \longrightarrow s_E$ such that the set-theoretical compositions of μ with s and of s' with ν are equal.



Let $K(a)$ be an $\exists^I$-formula: we denote by $[\![K]\!]$ the set of A_I^E-configurations satisfying K; in symbols, $[\![K]\!] := \{(\mathcal{M}, s) \mid \mathcal{M} \models K(s)\}$.

Proposition 5.6. *For every $\exists^I$-formula $K(a)$, the set $[\![K]\!]$ is upward closed; moreover, for every $\exists^I$-formulae K_1, K_2, we have $[\![K_1]\!] \subseteq [\![K_2]\!]$ iff $A_I^E \models K_1 \rightarrow K_2$.*

The set of configurations $\mathcal{B}(\tau, K)$ which are backward reachable from a given $\exists^I$-formula K is thus an upset, being the union of infinitely many upsets; however, even in case the latter are finitely generated, $\mathcal{B}(\tau, K)$ needs not be so. Under the hypothesis of local finiteness of T_E, this is precisely what characterizes termination of backward reachability search:

Theorem 5.7. *Assume that T_E is locally finite; let K be an $\exists^I$-formula. If K is safe, then* BReach *in Figure 2 terminates iff $\mathcal{B}(\tau, K)$ is a finitely generated upset.*[5] *As a consequence,* BReach *always terminates when the preorder on A_I^E-configurations is a wqo.*

Theorem 5.7 covers many well-known terminating cases (for some of them, a primitive recursive lower bound does not even exist). We just give few examples here.

- Take T_E to be an enumerated datatype theory and T_I to be the pure equality theory in the signature $\Sigma_I = \{=\}$ (in this setting one can formalize broadcast protocols): the preorder on A_I^E-configurations is a wqo by Dikson's Lemma.
- Take T_E to be an enumerated datatype theory and T_I to be the theory of total orders (in this setting one can formalize Lossy Channel Systems): the preorder on A_I^E-configurations is a wqo by Higman's Lemma.
- Take T_E to be the theory of rationals (under the natural ordering $<$) and T_I to be the pure equality theory in the signature $\Sigma_I = \{=\}$ (in this setting one can formalize the version of the Bakery algorithm mentioned in [2]): using Kruskal's theorem, it is possible to show that the preorder on A_I^E-configurations is a wqo.[6]

6 Progress Formulae for Recurrence Properties

Liveness problems are difficult and even more so for infinite state systems. In the following, we consider a special kind of liveness properties, which falls into the class of recurrence properties

[5] If K is unsafe, we already know that BReach terminates because it detects unsafety (cf. Theorem 5.4).

[6] We give details for the last example (for the first two, see Section 7 below). In the third example, we can represent a configuration $(\mathcal{M}, s)$ as a list $n_1, \ldots, n_k$ of natural numbers (of variable length k): such a list encodes the information that s_E is a k-element chain and that n_1 elements from s_I are mapped by s into the first element of the chain, n_2 elements from s_I are mapped by s into the second element of the chain, etc. If w is the list for s and v is the list for s', we have $s' \leq s$ iff w is componentwise less or equal to a subword of v. We can get termination by Kruskal's theorem by representing numbers as numerals and by using a binary function symbol f to encode the precedence (thus, for instance, the list 1,2,2 is represented as $f(s(0), f(s(s(0)), s(s(0))))$: in fact, it is easily seen that, on these terms, the homeomorphic embedding behaves like our configuration ordering.

in the classification introduced in [21]. Our method for dealing with such properties consists in synthesizing progress functions definable at quantifier free level.

A recurrence property is a property which is *infinitely often* true in every infinite run of a system; in other words, to check that R is a recurrence property it is sufficient to show that it cannot happen that there is an infinite run $s_0, s_1, \ldots, s_i, \ldots$ such that R is true at most in the states from a finite prefix $s_0, \ldots, s_m$. This can be formalized in our framework as follows.

Definition 6.1. Suppose that $R(a)$ is a $\forall^I$-formula; let us use the abbreviations $K(a)$ for $\neg R(a)$ and $\tau_K(a, a')$ for $\tau(a, a') \wedge K(a) \wedge K(a')$.[7] We say that R is a *recurrence property* iff for every m the infinite set of formulae

$$I(b_1), \tau(b_1, b_2), \ldots, \tau(b_m, a_0), \tau_K(a_0, a_1), \tau_K(a_1, a_2), \ldots \tag{8}$$

is not satisfiable in a finite index model of A_I^E.

The finite prefix $I(b_1), \tau(b_1, b_2), \ldots, \tau(b_m, a_0)$ in (8) above ensures that the states assigned to $a_0, a_1, \ldots$ are (forward) reachable from an initial state, whereas the infinite suffix $\tau_K(a_0, a_1)$, $\tau_K(a_1, a_2), \ldots$ expresses the fact that property R is never attained. Observe that, whereas it can be shown that the A_I^E-satisfiability of (6) for safety problems (cf. Definition 5.1) is equivalent to its satisfiability in a finite index model of A_I^E, this is not the case for (8). This imposes the above explicit restriction to finite index models since, otherwise, recurrence might unnaturally fail in many concrete situations.

Example 6.2. In a mutual exclusion protocol, we can choose K to be

$$\exists i \, (i = c \wedge a[i] = \texttt{waiting})$$

(here c is a fresh constant[8] of type Σ_I and, of course, $T_E \models \texttt{waiting} \neq \texttt{critical}$). Then the A_I^E-satisfiability of (8) implies that the protocol cannot guarantee absence of starvation.

Let R be a recurrence property; we say that R has *polynomial complexity* iff there exists a polynomial $f(n)$ such that for every m and for $k > f(n)$ the formulae

$$I(b_1) \wedge \tau(b_1, b_2) \wedge \cdots \wedge \tau(b_m, a_0) \wedge \tau_K(a_0, a_1) \wedge \cdots \wedge \tau_K(a_{k-1}, a_k) \tag{9}$$

are all unsatisfiable in the models $\mathcal{M}$ of A_I^E such that the cardinality of the support of $\mathcal{M}_I$ is at most n (in other words, $f(n)$ gives an upper bound for the waiting time needed to reach R by a system consisting of less then n processes).

[7] Notice that τ_K is easily seen to be equivalent to a disjunction of T-formulae, like the original τ.

[8] Notice that the addition of free constants preserves the hypotheses (local finiteness, closure under substructures) we have for T_E.

Definition 6.3. Let $R(a)$ be a $\forall^I$-formula and let K and τ_K be as in Definition 6.1. A formula $\psi(\underline{t}(\underline{j}), a[\underline{t}(\underline{j})])$ is an *index invariant* iff there exists a safe $\exists^I$-formula H such that

$$A_I^E \models \forall a_0 \, \forall a_1 \, \forall \underline{j} \, \big(\neg H(a_0) \wedge \tau_K(a_0, a_1) \rightarrow (\psi(\underline{t}, a_0[\underline{t}]) \rightarrow \psi(\underline{t}, a_1[\underline{t}]))\big). \tag{10}$$

A *progress condition for* R is a finite set of index invariant formulae

$$\psi_1(\underline{t}(\underline{j}), a[\underline{t}(\underline{j})]), \ldots, \psi_c(\underline{t}(\underline{j}), a[\underline{t}(\underline{j})])$$

for which there exists a safe $\exists^I$-formula H such that

$$A_I^E \models \forall a_0 \, \forall a_1 \, \big(\neg H(a_0) \wedge \tau_K(a_0, a_1) \rightarrow \exists \underline{j} \bigvee_{k=1}^{c} (\neg \psi_k(\underline{t}, a_0[\underline{t}])) \wedge \psi_k(\underline{t}, a_1[\underline{t}])\big) \tag{11}$$

Suppose there is a progress condition as above for R; if $\mathcal{M}$ is a model of A_I^E such that the support of $\mathcal{M}_I$ has cardinality at most n, then the formula (9) cannot be satisfiable in $\mathcal{M}$ for $k > c \cdot n^\ell$ (here ℓ is the length of the tuple $\underline{j}$). This argument shows the following.

Proposition 6.4. *If there is a progress condition for R, then R is a recurrence property with polynomial complexity.*

We provide significant examples (like the Bakery protocol) of application of Proposition 6.4 in Section 7 below; let us now discuss how to mechanize it. The validity tests (10) and (11) can be reduced to unsatisfiability tests covered by Theorem 4.1; however, the search space for progress conditions looks to be unbounded for various reasons (the number of the ψ's of Definition 6.3, the length of the string $\underline{j}$, the 'oracle' safe H's, etc.). Nevertheless, the proof of the following unexpected result shows (in the appropriate hypotheses) how to find progress conditions whenever they exist:

Theorem 6.5. *Suppose that T_E is locally finite and that safety of $\exists^I$-formulae can be effectively checked. Then the existence of a progress condition for a given $\forall^I$-formula R is decidable.*

7 Case Analysis and Further Examples

In this section, we first give a complete analysis of two examples and then we show how two well-known classes of problems (broadcast protocols and lossy channel systems) can be formalized in our framework. The claims we make about the Simplified Bakery Algorithm and the Insertion Sort Algorithm below have been checked by a tool generating proofs obligations for the SMT solver Yices: the tool is currently still under development and it has been assisted with manual simplifications steps.

We first make few notational remarks which are useful when formalizing the examples:

Remark 7.1. Sometimes it happens that the index signature Σ_I contains some constant c. In this case we would not be allowed in principle to use the term $a[c]$ in an $\exists^I$-formula like

$$\exists \underline{i}\, \phi(\underline{i}, a[\underline{i}]) \tag{12}$$

This is because, according to our conventions, $\phi(\underline{i}, a[\underline{i}])$ should be obtained from a quantifier free $\Sigma_I \cup \Sigma_E$-formula $\phi(\underline{i}, \underline{e})$ by replacing the $\underline{e}$ by the terms $a[\underline{i}]$. Nevertheless, the term $a[c]$ can be used in practice, because if it occurs in ϕ, we can replace (12) by

$$\exists \underline{i} \exists j\, (j = c \wedge \phi'(\underline{i}, j, a[\underline{i}], a[j])) \tag{13}$$

where ϕ' is obtained from ϕ by substituting c with j (one should be aware, however, that algorithms like that of Proposition 5.3 apply not to (12) but to the amended version (13)). A similar remark applies to $\forall^I$-formulae and to T-formulae (on the contrary, we recall that in the index invariant formulae of Definition 6.3, the above transformation is not needed, because $\psi(\underline{t}, a[\underline{t}])$ is the legal format for such formulae).

Remark 7.2. When we specify global components of T-formulae we need a formula of the kind $\phi_G(\underline{i}, \underline{d}, \underline{d}', j, e, e')$ satisfying the seriality requirement (1). In practice, the parameters e, e' are instantiated to $a[j], a'[j]$, respectively, and moreover the side parameters $\underline{d}, \underline{d}'$ are instantiated to $a[\underline{i}], a'[\underline{i}]$. Thus, for clarity, we directly use the format

$$\phi_G(\underline{i}, a[\underline{i}], a'[\underline{i}], j, a[j], a'[j])$$

when introducing global components.

Remark 7.3. In Definition 3.2, we assumed that array-based systems can have multiple local variable arrays; however, in the remaining part of the paper, we limited ourselves for simplicity to array based systems with only one local variable array. In order to formalize the examples, however, we need multiple local variable arrays: this can be achieved by restoring the original Definition 3.2, but can be achieved also by allowing a multisorted T_E, in the way sketched below. If we want for instance each process to have two local variables of sorts E_1 and E_2, respectively, we can take as T_E a three-sorted theory where, in addition to E_1, E_2 we have a third sort representing the cartesian product of E_1 and E_2 (the signature Σ_E will include symbols for projections and pairing). It can be shown that this multisorted T_E has quantifier elimination in case the component theories have quantifier elimination. Whatever formal solution is adopted, in case of multiple local variable arrays, we shall use the notation $a[i].1, \ldots, a[i].n$ to access the content of the n local variables of the process i (thus, depending on the formal solution the reader prefers, $a[i].k$ may be the k-th projection of the content of the unique local variable of the process i or the real k-th local variable of the process i).

Remark 7.4. Global variables can be modeled by adding extra local variables: if the transition (and the initial condition) are suitably chosen, one can obtain the effect that processes keep, among their local variables, an identical copy of each global variable. Global variables will be needed only in Subsection 7.4 below.

7.1 The Simplified Bakery Algorithm

We present here the formalization into our framework of the *Simplified Bakery Algorithm* that can be found for instance in [3]. When formalizing the component transitions τ_1, τ_2 below, we use the *approximation* trick (see [3, 2]): transitions τ_1, τ_2, in order to fire, would require a universally quantified guard which cannot be expressed in our format. We circumvent the problem, by imposing that all the processes that are counterexamples to the guard go into a 'crash' state: this trick augments the possible runs of the system by introducing 'spurious runs', however if a safety property holds for the augmented 'approximate' system, then it also holds for the original system (the same is true for the recurrence properties of Section 6).[9] We specify suitable theories T_I and T_E for the formalization of (the approximate version of) the Simplified Bakery Algorithm. The theory T_I is the pure equality theory in the signature $\Sigma_I = \{=\}$; to introduce T_E, we analyze which kind of local data we need. The data e appearing in an array of processes are records consisting of the following two fields:

- the field $e.s$ represents the status ($e.s \in \{\texttt{idle}, \texttt{wait}, \texttt{use}, \texttt{crash}\}$);
- the field $e.t$ represents the ticket (tickets range in the real interval $[0, \infty)$, seen as a linear order with first element 0).

Thus we need a two-sorted T_E and two array variables in the language (or, a single array variable and a three-sorted T_E).[10]

The transition $\tau(a, a')$ is the disjunction $\tau_1(a, a') \vee \tau_2(a, a') \vee \tau_3(a, a')$, where the T-formulae τ_n ($n \leq 3$) have the standard format from Definition 3.4

$$\exists i \, (\phi_L^n(i, a[i], a'[i]) \wedge \mathit{Update}_G^n(i, a, a'));$$

and the local and the global components ϕ_L^n, ϕ_G^n are specified below.

[9] The approximate transition trick has also a rationale coming from textbooks on distributed algorithms [20], where the so called 'stopping failures' assumption is widely discussed: in the stopping failures assumption, every process can go into a crash state at any time. If you like, you can add a fourth T-formula disjunct to our formalization below in order to take into account unlimited stopping failures: our analysis of the simplified bakery algorithm carries out without substantial modification within such full stopping failures assumption.

[10] In the latter case Σ_E contains constants for $\texttt{idle}, \texttt{wait}, \texttt{use}, \texttt{crash}$, the linear order relation $<$, the constant 0, projections and pairing functions (all these symbols are appropriately sorted). The class of models of T_E is formed by the unique three-sorted structure given by the set $\{\texttt{idle}, \texttt{wait}, \texttt{use}, \texttt{crash}\}$, the interval $[0, \infty)$, and the cartesian product of the latter. Notice that this T_E has quantifier elimination and is locally finite.

Local Components. The local components ϕ_L^n are

- $\phi_L^1 \quad :\equiv \quad a[i].s = \texttt{idle} \wedge a'[i].s = \texttt{wait} \wedge a'[i].t > 0$ (the selected process goes from the state `idle` to `wait` and takes a nonzero ticket);
- $\phi_L^2 \quad :\equiv \quad a[i].s = \texttt{wait} \wedge a'[i] = \langle \texttt{use}, a[i].t \rangle$ (the selected process goes from the state `wait` to `use` and keeps the same ticket);
- $\phi_L^3 \quad :\equiv \quad a[i].s = \texttt{use} \wedge a'[i] = \langle \texttt{idle}, 0 \rangle$ (the selected process goes from the state `use` to `idle` and the ticket is reset to zero).

Global Components. The global components $\phi_G^n(i, a[i], a'[i], j, a[j], a'[j])$ make case distinction as follows (the first case delegates the update to the local component):

- $\phi_G^1 \quad :\equiv \quad (j = i \wedge a'[j] = a'[i]) \vee (j \neq i \wedge a[j].t < a[i].t \wedge a'[j] = a[j]) \vee (j \neq i \wedge a[j].t \geq a[i].t \wedge a'[j] = \langle \texttt{crash}, a[j].t \rangle)$ (if the selected process i goes from `idle` to `wait`, then any other process j keeps the same state/ticket, unless it has a ticket bigger or equal to the ticket of i – in which case j crashes);
- $\phi_G^2 \quad :\equiv \quad (j = i \wedge a'[j] = a'[i]) \vee (j \neq i \wedge (a[j].t > a[i].t \vee a[j].t = 0) \wedge a'[j] = a[j]) \vee (j \neq i \wedge \neg(a[j].t > a[i].t \vee a[j].t = 0) \wedge a'[j] = \langle \texttt{crash}, a[j].t \rangle)$ (if the selected process i goes from `wait` to `use`, then any other process j keeps the same state/ticket, unless it has a nonzero ticket smaller than the ticket of i – in which case j crashes);
- $\phi_G^3 \quad :\equiv \quad (j = i \wedge a'[j] = a'[i]) \vee (j \neq i \wedge a'[j] = a[j])$ (if the selected process i goes from `use` to `idle`, then any other process keeps the same state/ticket).

Safety Problem. The safety problem we want to consider is mutual exclusion. We impose the initial condition $\forall i\, (a[i].s = \texttt{idle})$ and ask whether it is possible to reach states satisfying the $\exists^I$-formula:

$$\exists i\, \exists j\, (i \neq j \wedge a[i].s = \texttt{use} \wedge a[j].s = \texttt{use}).$$

The backward analysis algorithm of Theorem 5.4 terminates after 4 steps saying that the unsafe states cannot be reached. Subsumption and simplification tests help calculations (for instance, they keep the number of existentially quantified variables in the current $\exists^I$-formula always equal to 2). The fixpoint describing backward reachable states is symbolically described (up to A_E^I-equivalence) by the existential closure following formula written in SMT-LIB syntax [24]:

```
( or
  ( and ( not ( = i j ) ) ( = ( a0.s i ) use ) ( = wait ( a0.s j ) ) ( = 0 ( a0.t j ) ) )
  ( and ( not ( = i j ) ) ( = ( a0.s i ) idle ) ( = wait ( a0.s j ) ) ( = 0 ( a0.t j ) ) )
```

```
  ( and ( not ( = i j ) ) ( = idle ( a0.s i ) ) ( = ( a0.s j ) use ) ( = ( a0.t j ) 0 ) )
  ( and ( not ( = i j ) ) ( = wait ( a0.s i ) ) ( = ( a0.s j ) wait ) ( = ( a0.t j ) 0 ) )
  ( and
    ( not ( = i j ) ) ( = ( a0.s i ) wait ) ( = use ( a0.s j ) )
    ( or ( = ( a0.t j ) 0 ) ( < ( a0.t i ) ( a0.t j ) ) )
  )
  ( and ( not ( = i j ) ) ( = ( a0.s i ) use ) ( = ( a0.s j ) use ) )
)
```

Recurrence Problem. We introduce a free index constant d and inquire about the existence of an infinite run keeping the system in the starvation state

$$a[d].s = \texttt{wait}.$$

We use, as a progress condition, the following four index invariant formulae:

$$a[j].s = \texttt{crash}$$
$$a[j].t > a[d].t$$
$$\neg(a[j].t < a[d].t \ \wedge \ a[j].s = \texttt{wait})$$
$$a[j].t = 0 \ \vee \ a[j].t > a[d].t.$$

To check that these four formulae are a progress condition, we need further safety checks (as explained in Definition 6.3): in fact, the information concerning unreachability of the states satisfying the following $\exists^I$-formulae is needed when checking conditions (10), (11):

$$\exists i \, (a[i].s = \texttt{use} \wedge a[i].t = 0)$$
$$\exists i \, (a[i].s = \texttt{idle} \wedge a[i].t \neq 0)$$
$$\exists i \, (a[i].s = \texttt{wait} \wedge a[i].t = 0)$$
$$\exists i \exists j \, (a[i].s = \texttt{use} \wedge a[j].s = \texttt{wait} \wedge a[j].t \leq a[i].t).$$

Recurrence of $a[d].s \neq \texttt{wait}$ (with a linear complexity bound) follows, as explained in Section 6: this proves that each non faulty process d is guaranteed to get the resource each time it asks for it (in fact, if d does not crashes, it can exit the `wait` state only by going to `use`).

7.2 The Insertion Sort Algorithm

We analyze here a simple algorithm inserting an element $a[0]$ into a sorted array $a[1], \ldots, a[n]$. We take as T_I the theory in the signature $\Sigma_I = \{0, S\}$ whose class of models are disjunct

unions[11] of finite linear-like structures of the kind

$$i_0\, S\, i_1\, S \cdots S\, i_{k-1}\, S\, i_k$$

satisfying the further condition that 0 does not have predecessor (i.e. $\forall i\, \neg S(i,0)$ must hold).

For local data, we need pairs, whose first component in a Boolean flag and whose second component is an element from a linear order with quantifier elimination (let us take for example the linear order over rationals): we write $e.1$ and $e.2$ for the first and the second field of the record e.

The algorithm is initialized to a situation where $S(i,j) \rightarrow a[i].2 \leq a[j].2$ holds for all i, j, except for the case $i = 0$; moreover, in the initial situation, the Boolean flag is set as $a[i].1 = 0$, except for $i = 0$ where we have $a[0].1 = 1$. This means that everything is arranged in an increasing order (as far as second field data is concerned), except for the index 0; however, the index 0 owns a token which will be passed to the right till the insertion of $a[0].2$ is completed. The node i getting the token changes its flag from $a[i].1 = 0$ to $a[i].1 = 1$; the node passing the token to its successor keeps its flag set to 1. We want to check that *this algorithm halts in polynomial time and ends into a situation where the second field data are appropriately sorted.*

The transition specifying our array based system has local component $\phi_L(i_1, i_2, a[i_1], a[i_2])$ given by

$$S(i_1,i_2) \wedge a[i_1].1 = 1 \wedge a[i_2].1 = 0 \wedge a[i_1].2 > a[i_2].2 \wedge$$
$$\wedge\ a'[i_1] = \langle 1, a[i_2].2\rangle \wedge a'[i_2] = \langle 1, a[i_1].2\rangle$$

and global component given by

$$(j \neq i_1 \wedge j \neq i_2 \wedge a'[j] = a[j]) \vee (j = i_1 \wedge a'[j] = a'[i_1]) \vee (j = i_2 \wedge a'[j] = a'[i_2]).$$

The initial state description is the following

$$\forall i\, (a[i].1 = 0 \leftrightarrow i \neq 0) \wedge \forall i_1 \forall i_2\, (S(i_1,i_2) \rightarrow i_1 = 0 \vee a[i_1].2 \leq a[i_2].2).$$

We specify a safety and a recurrence problem for this system: the conjunction of the claims of the two problems states precisely that the system behaves as intended.

The recurrence problem is simply termination: we take the contradictory formula $false$ as R (so that the τ_K of Definition 6.3 is τ) and apply Proposition 6.4, by using the progress condition

$$a[j].1 = 1.$$

[11] We need disjoint unions because closure under substructures is required by Definition 3.1 of an index theory (this enlargement of the class of intended models widens – but in a non significant way – our total correctness results below).

Thus, we know that the system must halt[12] (actually in linear time because our progress condition has just one free variable).

For the safety problem, we use the following $\exists^I$-formula for describing unsafe states:

$$\exists i_1\, \exists i_2\, (S(i_1, i_2) \wedge \neg(a[i_1].1 = 1 \wedge a[i_2].1 = 0) \wedge a[i_1].2 > a[i_2].2). \tag{14}$$

Once this safety problem is positively solved, we know that the array of every reachable state cannot possibly be sorted just because there is an order inversion between the state owning the token and its immediate neighbour: this is precisely the situation in which the transition can fire. Thus, when the transition cannot fire, the array is sorted (and we know by the above recurrence check that the system always reaches a state in which the transition cannot fire).

Unfortunately, the backward analysis of the algorithm of Theorem 5.4 applied to (14) diverges:[13] the reason for non termination is the lack of the information that configurations satisfying $S(i, j) \wedge a[i].1 = 0 \wedge a[j].1 = 1$ are not reachable (during execution, the flag array is always of the kind 1^*0^*). Thus we replace the original unsafe states description by the following stronger one:

$$\begin{aligned}\exists i_\exists i_2\, ((S(i_1, i_2) \wedge a[i_1].1 = 0 \wedge a[i_2].1 = 1) \vee \\ \vee\, (S(i_1, i_2) \wedge \neg(a[i_1].1 = 1 \wedge a[i_2].1 = 0) \wedge a[i_1].2 > a[i_2].2)).\end{aligned}$$

Now backward reachability analysis terminates in two steps and safety can be tested (thus completing the formal proof of the total correctness of our insertion algorithm). The fixpoint describing backward reachable states is symbolically described (up to A^I_E-equivalence) by the existential closure following formula written in SMT-LIB syntax:

```
( or
  ( and ( S x y ) ( = (a0.1 x) 1 ) ( = ( a0.1 y ) 0 ) ( > ( a0.2 x ) ( a0.2 y ) ) )
  ( and ( S x y ) ( = ( a0.1 x ) 0 ) ( = ( a0.1 y ) 1 ) )
  ( and
    ( S x y ) ( S y z ) ( = 1 ( a0.1 x ) ) ( = ( a0.1 y ) 1 ) ( = ( a0.1 z ) 0 )
    ( > ( a0.2 y ) ( a0.2 z ) ) ( > ( a0.2 x ) ( a0.2 z ) )
  )
)
```

The same formula is a fixpoint also in case T_I is replaced by the weaker theory having (disjoint unions of) finite *trees* as models: notice that, in this case, the algorithm becomes non deterministic and does a less trivial job.

7.3 The Formalization of Broadcast Protocols

A *broadcast protocol* (following the formalization in [14, 12]) is a triple (S, L, R) where:

[12] The fact that the contradiction $false$ is recurrent means precisely that there are no infinite runs at all.

[13] This shows that local finiteness of T_E is not sufficient for guaranteeing termination of backward search.

– S is a finite set of *locations*;[14]

– L is a set of *labels*, composed out of a set Σ_l of *local* labels, two sets $\Sigma_r \times \{?\}$ and $\Sigma_r \times \{!\}$ of *input* and *output rendez-vous* labels, and two sets $\Sigma_b \times \{??\}$ and $\Sigma_b \times \{!!\}$ of *input* and *output broadcast* labels, where $\Sigma_l, \Sigma_r, \Sigma_b$ are disjoint finite sets;

– $R \subseteq S \times L \times S$ is a set of *transitions* satisfying the following properties: (a) for every $a \in \Sigma_b$ and every location $\ell \in S$ there exists a location $\ell' \in S$ such that $(\ell, a??, \ell')$ belongs to R; (b) every label of the kind $a, a!, a?, a!!$ appears in exactly one transition.

The system evolves according three types of moves: *local* (meaning that a single process moves in isolation to a new location), *rendez-vous* (two processes exchange a message and move to new locations) and *broadcast* (a process send a message to all other processes and all processes move to new locations). More precisely, shortening the formalism for the translations $(\ell, l, \ell') \in R$ into $\ell \xrightarrow{l} \ell'$:

(i) the meaning of the transition $\ell_i \xrightarrow{a} \ell_j$ is that a single process moves from the location ℓ_i to the location ℓ_j;

(ii) the meaning of the rendez-vous $\ell_i \xrightarrow{a!} \ell_j$ and $\ell_k \xrightarrow{a?} \ell_l$ is that a process moves from the location ℓ_i to the location ℓ_j, another process moves from ℓ_k to ℓ_l, and all the other processes keep their location;

(iii) the meaning of the broadcast $\ell_i \xrightarrow{a!!} \ell_j$ is that a process leaves ℓ_i and reaches ℓ_j and all the other processes moves from their locations according to transitions of the kind $\ell_l \xrightarrow{a??} \ell_k$.

We can translate this description into our settings as follows: first of all, we choose as T_I the theory $(\Sigma_I, \mathcal{C})$, where $\Sigma_I = \{=\}$ and $\mathcal{C}$ is the class of all finite sets; as T_E, we take the theory of the enumerated data-type that describes the finite set of locations S.

In order to specify the transition $\tau(a_0, a_1)$ of our system, we take a disjunction of T-formulae, *one disjunct for each* local, rendez-vous and broadcast transition. These T-formulae are specified as follows.

Let us consider first a local transition of the kind $\ell_l \xrightarrow{a} \ell_r$: this is translated into the T-formula:

$$\exists i \left(a_0[i] = \ell_l \wedge a_1[i] = \ell_r \wedge \forall j \left((j = i \wedge a_1[j] = a_1[i]) \vee (j \neq i \wedge a_1[j] = a_0[j])\right)\right).$$

[14] We use the name 'location' here (and not 'state' as in [14, 12]), because the word 'state' is used in this paper to denote the state of an array-based system in the sense of Definition 3.3.

A rendez-vous of the kind $\ell_l \xrightarrow{a!} \ell_r$ and $\ell_k \xrightarrow{a?} \ell_w$ is translated into the T-formula:

$$\exists i, j \Big(i \neq j \wedge a_0[i] = \ell_l \wedge a_0[j] = \ell_k \wedge a_1[i] = \ell_r \wedge a_1[j] = \ell_w \wedge$$
$$\wedge \forall h \big((h = i \wedge a_1[h] = a_1[i]) \vee (h = j \wedge a_1[h] = a_1[j]) \vee (h \neq i \wedge h \neq j \wedge a_1[h] = a_0[h]) \big) \Big).$$

The T-formula that represents a broadcast of the kind $\ell_l \xrightarrow{a!!} \ell_r$ is the following:

$$\exists i \Big(a_0[i] = \ell_l \wedge a_1[i] = \ell_r \wedge \forall j \big((j = i \wedge a_1[j] = a_1[i]) \vee$$
$$\vee \bigvee_{\ell_k \xrightarrow{a??} \ell_w} (j \neq i \wedge a_0[j] = \ell_k \wedge a_1[j] = \ell_w) \big) \Big),$$

varying the last big disjunction over all the transitions of the kind $\ell_k \xrightarrow{a??} \ell_w$ in R. Notice that the requirement that for every $a \in \Sigma_b$ and every location $\ell \in S$ there exists a location $\ell' \in S$ such that $\ell \xrightarrow{a??} \ell'$ belongs to R is sufficient here to guarantee the seriality requirement for the global component contained in the T-formula above (but, since we do not require in principle the uniqueness of such an ℓ', the functionality property may not hold).

In order to specify the initial set of states of our array-based system, we would like to use formulae of the kind

$$\exists \underline{i} \forall \underline{j}\, \phi(\underline{i}, \underline{j}, a[\underline{i}], a[\underline{j}]), \tag{15}$$

because only with the help of such formulae we can say that the initial states are precisely those that belong to a given parameterized configuration in the sense of [14, 12].[15] Formulae (15) are not $\forall^I$-formulae, hence they do not fit the requirement of Definition 3.2 for being initial formulae. However, this is not a real problem: one could for instance skolemize (15) and expand the signature of T_I with finitely many free constants (the new T_I is trivially again an index theory). Alternatively, one can simply observe that the initial formula I is needed only in the consistency tests A_I^E-$\mathsf{check}(BR^i(\tau, K) \wedge I)$ of the algorithm of Figure 2: such tests fall within the decision problem of Theorem 4.1 even in case I is of the form (15). Thus, we can freely assume our initial formula I to have the desired form (15) and complete the specification of our array-based systems $\mathcal{S} = (a, I, \tau)$ for broadcast protocols.

Before analyzing safety problems, let us investigate what kind of configurations we get as a consequence of our current choice of T_I and T_E. Suppose $S = \{\ell_1, \ldots, \ell_k\}$; now, an A_I^E-configuration $(s, \mathcal{M})$ is determined uniquely by a k-tuple of integers $n_1, \ldots, n_k$ (these integers are the cardinalities of the sets of indices which are mapped by s onto $\ell_1, \ldots, \ell_k$, respectively).

[15] According to [14, 12], a parameterized configuration is the set $X_{\mathbf{p}}$ of states (in the formal sense of our Definition 3.3) which is induced by a partial function $\mathbf{p} : S \longrightarrow \mathbb{N}$ as follows. A state s belongs to $X_{\mathbf{p}}$ iff it satisfies the following condition: (*) if ℓ is in the domain of $\mathbf{p}$, then there are exactly $\mathbf{p}(\ell)$ indexes i such that $s(i) = \ell$.

If $(n_1, \ldots, n_k)$ is the k-tuple for s and $(n'_1, \ldots, n'_k)$ is the k-tuple for another configuration s', we have $s' \leq s$ iff $n'_1 \leq n_1 \,\&\cdots\&\, n'_k \leq n_k$. This preorder among configurations is a wqo, by Dikson's lemma, hence the backward search algorithm of Figure 2 terminates by Theorem 5.7.

Safety problems for broadcast protocols are formulated in [14, 12] as reachability problems for upsets of configurations; since all upsets of configurations are finitely generated by Dikson's Lemma, safety problems as formulated in [14, 12] *coincide with safety problems for* $\exists^I$*-formulae* in the sense of the present paper, by Proposition A.6.

The decidability argument in [12] is based on Dikson's Lemma, as our argument coming from a direct application of Theorem 5.7;[16] the formalization of broadcast protocols employed in [14, 12] is however different, less declarative but more efficient, and directly based on the formalism of configurations. We sketch how to reproduce this alternative formalization in our setting too. We take as T_I the enumerated datatype theory of the set S and as T_E the theory $(\Sigma_E, \mathcal{C})$, where $\Sigma_E = \{0, S, \leq\}$ and $\mathcal{C} = \{\mathbb{N}\}$ (this is the set of natural numbers equipped here with the natural interpretation of zero, successor and ordering). To specify the T-formulae which are the disjunctions of the transition τ, we use formulae expressing the values $a'[\ell_1], \ldots, a'[\ell_k]$ in terms of suitable affine transformations applied to the values $a[\ell_1], \ldots, a[\ell_k]$ (see again [14, 12] for the affine transformations corresponding to local, rendezvous and broadcast transitions). As initial formulae we now use formulae of the kind[17] $a[\ell_{i_1}] = \overline{n_1} \wedge \cdots \wedge a[\ell_{i_h}] = \overline{n_h}$, whereas the safety problems we are interested in are expressed by formulae of the kind[18] $a[\ell_1] \geq \overline{m_1} \wedge \cdots \wedge a[\ell_k] \geq \overline{n_k}$. This formal settings fits our framework: termination of the backward search algorithm of Figure 2 cannot be justified by Theorem 5.7 anymore, however it follows anyway because this new formalization is 'semantically equivalent' to the previous one.

7.4 The Formalization of Lossy Channels Systems

The formalization of Lossy Channels Systems in our framework is indirect and up to bisimulation, but it is still sufficient to formulate relevant model checking problems. Decidability of reachability problems follows as a direct application of Theorem 5.7 and of Higman's lemma; our argument is similar (but not quite the same) to the argument used for the proof of the same result in [5] (the latter argument is also based on Higman's lemma, but the ordering employed for S-configurations is in a sense dual to ours).

[16] As mentioned in [14, 12] however, the complexity of the reachability problem is quite high, because the only known upper bound for the number of iterations needed until termination is non-primitive recursive.

[17] Here $\overline{n}$ is the numeral of n, that is the term obtained by applying n-times S to 0.

[18] Notice that we do not even need quantified index variables in order to deal with these safety model checking problems.

We first recall the standard definition of a channel system, but for simplicity, we limit ourselves to a single automaton using a single channel as a fifo buffer (for more information the reader is referred to [5]).

A *channel system* is a triple $S = (Q, \Sigma, \delta)$, where Q is a finite set of control locations, Σ is a finite alphabet and $\delta \subseteq Q \times \{!, ?\} \times \Sigma^* \times Q$ is a set of transitions. We also assume (without loss of generality for the problems we are interested in) that δ contains the idle transitions $(q, *, \varepsilon, q)$ for all $q \in Q$ (here ε is the empty word and $*$ is either ! or ?).

An *S-configuration* is a pair $\gamma = (q, w)$, where $q \in Q$ is a control state and $w \in \Sigma^*$ is a word over Σ. We say that γ has a *perfect step* to γ' (in symbols $\gamma \to \gamma'$) iff the following happens: (i) γ is of the kind (q, w); (ii) γ' is of the kind (q', w') and either (iii.1) there is $(q, !, u, q') \in \delta$ such that $w' = wu$ or (iii.2) there is $(q, ?, u, q') \in \delta$ such that $w = uw'$. That is, in case (iii.1) the word u has been written to the tail of the channel, whereas in the case (iii.2) the word u has been read from the head of the channel; in both cases, the control location changes from q to q'.

Lossy steps simulate unreliability of the channel. Recall that v is said to be a subword of w (in symbols, $v \sqsubseteq w$) iff v results from w by deleting some (possibly zero, possibly non consecutive) occurrences of letters. We say that $\gamma = (q, w)$ has a *lossy step* to $\gamma' = (q', w')$ (in symbols $\gamma \rightsquigarrow \gamma'$) iff there are v, v' such that $v \sqsubseteq w$ and $w' \sqsubseteq v'$ and $(q, v) \to (q', v')$. That is, in a lossy step, the channel again performs an exact step but it may loose data both before and after it.

From the algorithmic point of view, we are interested in the following *lossy reachability* problem:[19] given a channel system S and given S-configurations γ_I, γ_f, is there a *finite lossy run* from γ_I to γ_f, i.e. are there $\gamma_0, \ldots, \gamma_n$ such that $\gamma_0 = \gamma_I$, $\gamma_n = \gamma_f$ and

$$\gamma_0 \rightsquigarrow \gamma_1 \rightsquigarrow \cdots \rightsquigarrow \gamma_n \ ? \tag{16}$$

Notice that, since we allow idle transitions (possibly causing loss of data in a lossy step), we can relax the above conditions $\gamma_0 = \gamma_I$ and $\gamma_n = \gamma_f$ a little: if $\gamma_I = (q_I, w_I)$ and $\gamma_f = (q_f, w_f)$, we can equivalently ask $\gamma = (q_0, w_0)$ (resp. $\gamma_f = (q_f, w_f)$) to be only such that $q_0 = q_I$ and $w_0 \sqsubseteq w_I$ (resp. $q_n = q_f$ and $w_f \sqsubseteq w_n$).

In order to formalize the lossy reachability problem in our framework, we first identify suitable theories T_I and T_E. The theory T_I must describe the cells of the channel, hence a natural choice is the theory whose signature contains only a binary predicate symbol $<$ and whose models are all linear orders.[20]

[19] Another interesting problem is termination: given γ_I, are all lossy runs (not involving identical idle steps) starting from γ_I finite? This problem is decidable, however our methods for liveness are unable to solve it, because termination can have no primitive recursive bound, as shown in [23].

[20] Only a finite part of the linear order which is a model of T_I will be used in a finite run, see below.

Before making our choice for T_E, we recall that an S-configuration $\gamma = (q, w)$ as described above, consists of an array of letters from Σ and of a *global* control variable ranging over Q. For simplicity, we did not model global variables in our setting,[21] but we can indirectly model them by asking indexes to keep identical copies of global variables. Thus local data modeled by T_E are records e, having a field $e.g \in Q$ and a field $e.d \in \Sigma + \{\mathsf{b}\}$ (we added to the alphabet Σ the 'blank' symbol b); in other words, T_E is just the enumerated datatype theory of the set $Q \times (\Sigma + \{\mathsf{b}\})$. The reason why we added the blank symbol b to the alphabet Σ is because the content of the channel is a word of variable length: in our setting we must choose a model of T_I and keep it constant for a whole run, that's why we need to fill with b the cells whose content is empty.

Given the above choices for T_I and T_E, a state of an array-based system built up on them (see Definition 3.3) is a function $s : L \longrightarrow Q \times (\Sigma + \{\mathsf{b}\})$, where L is a linear order (we recall that, when L is finite, such a state s is also called a configuration, see Subsection 5.2). After transition and initial conditions are introduced, it will be clear that only very special states will be reachable, namely those for which we have

(i) $a(i).d = \mathsf{b}$ for all but finitely many i;

(ii) $a(i).g = a(j).g$ for all i, j.

The states satisfying (i)-(ii) are called *standard*. With a standard state s we can associate an S-configuration $\pi(s) = (q, w)$ as follows: q is the common value of the $s(i).g$'s, whereas w is the ordered sequence – alias the Σ-word – $s(i_1).d, \ldots, s(i_k).d$ of the values of the $s(i).d$'s which are different from b (the sequence is ordered by the L-linear order of the cells $i_1 < \cdots < i_k$).

We are now ready to introduce a suitable transition formula τ, a suitable initial formula I and to show that runs of the resulting array-based system bisimulate lossy runs.

Given a channel system $S = (Q, \Sigma, \delta)$, our transition $\tau(a_0, a_1)$ is a disjunction of T-formulae and to form this disjunction we need *one T-formula for each transition of the kind* $(q, !, u, q') \in \delta$ *and one T-formula for each transition of the kind* $(q, ?, u, q') \in \delta$. Let us first examine the first kind of transitions: these transitions write the word u to the tail of the channel (and change the control location from q to q'). If we had to express exactly this, we would be in trouble, because a universally quantified condition would be unavoidable ("to be in the channel's tail" is expressible by something like "the content of *all* cells to the right is b"). However, the universal condition is not needed: since the channel may loose cells content, we can equivalently say that we write (up to message lost) u to the right of a certain cell i of the channel and that everything on the right of i gets lost or overwritten by u (the latter is

[21] A direct modeling of global variables is possible and not difficult, our choice of avoiding them in this paper was motivated only by notational simplicity.

the same as 'lost and then re-written'). To sum up, if $u = \sigma_1 \cdots \sigma_k$ $(k > 0)$,[22] the T-formula associated with $(q, !, u, q')$ has the existentially quantified index variables $i_1, \ldots, i_k$, the local component formula

$$i_1 < i_2 \wedge i_2 < i_3 \wedge \cdots \wedge i_{k-1} < i_k \wedge \bigwedge_{s=1}^{k} (a_0[i_s].g = q \wedge a_1[i_s].g = q') \wedge$$
$$\wedge \bigwedge_{s=1}^{k} (a_1[i_s].d = \sigma_s \vee a_1[i_s].d = \mathsf{b})$$

and the global component formula

$$a_1[j].g = q' \wedge \bigvee_{s=1}^{k} (j = i_s \wedge a_1[j].d = a_1[i_s].d) \vee$$
$$\vee (j < i_1 \wedge (a_1[j].d = a_0[j].d \vee a_1[j].d = \mathsf{b})) \vee (j > i_1 \wedge \bigwedge_{s=1}^{k} (j \neq i_s) \wedge (a_1[j].d = \mathsf{b}))$$

(notice that the above global component formula is serial but not functional).

For the second kind of transitions (those of the kind $(q, ?, u, q')$ with $u = \sigma_1 \cdots \sigma_k$ and $k > 0$), the local component is

$$i_1 < i_2 \wedge i_2 < i_3 \wedge \cdots \wedge i_{k-1} < i_k \wedge \bigwedge_{s=1}^{k} (a_0[i_s].g = q \wedge a_1[i_s].g = q') \wedge$$
$$\wedge \bigwedge_{s=1}^{k} (a_0[i_s].d = \sigma_s \wedge a_1[i_s].d = \mathsf{b})$$

and the T-update formula is

$$a_1[j].g = q' \wedge (j > i_k \wedge (a_1[j].d = a_0[j].d \vee a_1[j].d = \mathsf{b})) \vee (j < i_k \wedge (a_1[j].d = \mathsf{b})).$$

We have to treat separately the case $k = 0$, i.e. the case in which u is the empty word: in that case, we have one existentially quantified index variable i, the local component is $a_0[i].g = q \wedge a_1[i].g = q'$ and global component is

$$a_1[j].g = q' \wedge (a_1[j].d = a_0[j].d \vee a_1[j].d = \mathsf{b})$$

(that is, the control location goes from q to q', the content of some cells gets lost, and the content of the remaining cells is not varied).

The following facts (concerning 'bisimulation' of our standard states and S-configurations) should be clear. Consider a standard state s (taken from a model $\mathcal{M}$) and its projected S-configurations $\pi(s)$:

[22] The case $k = 0$ is treated separately below.

(a) for every s' such that $\mathcal{M} \models \tau(s, s')$, we have that $\pi(s) \rightsquigarrow \pi(s')$;

(b) for every γ such that $\pi(s) \rightsquigarrow \gamma$ there exists s' such that $\mathcal{M} \models \tau(s, s')$ and $\pi(s') = \gamma$, provided $\texttt{INDEX}^{\mathcal{M}}$ is sufficiently large (this is always the case when $\texttt{INDEX}^{\mathcal{M}}$ is infinite);

(c) for every S-configuration γ there exists a standard state t such that $\pi(t) = \gamma$: actually, there are many such t and we can always succeed in picking such a t from every model $\mathcal{M}$ such that $\texttt{INDEX}^{\mathcal{M}}$ is sufficiently large, e.g. infinite.

The initial condition can be expressed as the $\forall^I$-formula I below, given a choice $\gamma_I = (q_I, w_I)$ for an initial S-configuration (let w_I be $\sigma_1 \cdots \sigma_k$):

$$\forall i\,(a[i].g = q_I) \wedge \forall i_1 \cdots \forall i_k \forall i_{k+1} \Big(\bigwedge_{s=1}^{k} i_s < i_{s+1} \rightarrow \bigvee_{s=1}^{k+1} a[i_s].d = \mathsf{b} \Big) \wedge$$
$$\wedge \bigwedge_{l \leq k} \forall i_1 \cdots \forall i_l \Big(\bigwedge_{s=1}^{l} i_s < i_{s+1} \wedge \bigwedge_{s=1}^{l} a[s].d \neq \mathsf{b} \rightarrow \bigvee_{\rho_1 \cdots \rho_k \sqsubseteq w_I} \bigwedge_{s=1}^{l} a[i_s].d = \rho_s \Big).$$

This formula says that the global variable is set to q_I, that there is no non-blank sequence of cells whose length is bigger than the length of w_I, and that the content of all non-blank sequences of cells whose length is smaller or equal to the length of w_I must be a subword of w_I: in other words, every status s satisfying this formula is standard and its projected configuration $\pi(s)$ must be of the kind (q_I, w) with $w \sqsubseteq w_I$.

Suppose we are now given an S-configuration $\gamma_f = (q_f, w_f)$ (let w_f be $\sigma_1 \cdots \sigma_k$); from (a)-(b)-(c) above it follows that *lossy unreachability of γ_f from γ_I can be equivalently stated as the safety of the $\exists^I$-formula K given by*

$$\exists i_1 \cdots \exists i_k \left(a[i_1].g = q_f \wedge \bigwedge_{s=1}^{k-1} (i_s < i_{s+1} \wedge a[i_s].d = \sigma_s) \wedge a[i_k].d = \sigma_k \right)$$

(for $k = 0$, simply use $\exists i(a[i].g = q_f)$ as K).

We now apply Theorem 5.7 to prove that safety of the above K is decidable.[23] To this aim, it is sufficient to show that the ordering among configurations is a wqo. Since T_I is the theory of total orders and T_E is the enumerated datatype theory for the set $Y := Q \times (\Sigma + \{\mathsf{b}\})$, an A_I^E-configuration $(\mathcal{M}, s)$ is determined by a finite word over Y (a word over Y is in fact a map from a finite totally ordered set into Y). If w is the word for s and v is the word for s', we have $s \leq s'$ iff w is a subword of v: Higman's lemma immediately implies that this is a wqo.

[23] As shown in [23], however, the complexity of the problem is quite high, being non primitive recursive.

8 Related Work and Conclusions

A popular approach to uniform verification (e.g., [2, 3, 7]) consists of defining a finitary representation of the (infinite) set of states and then to explore the state space by using such a suitable data structure. Although such a data structure may contain declarative information (e.g., constraints over some algebraic structure or first-order formulae), a substantial part is non-declarative. Typically, the part of the state handled non-declaratively is that of indexes (which, for parameterized systems, specify the topology), thereby forcing to re-design the verification procedures whenever their specification is changed (for the case of parameterized systems, whenever the topology of the systems changes). Since our framework is fully declarative, we avoid this problem altogether.[24] In this perspective, the expressivity potential of our approach is really considerable, as exemplified by the following considerations: we have seen in Section 7, how one can profitably include many array variables and/or many-sorted elements theories in our settings. The inclusion of many-sorted index theories is also important (and straightforward): suppose for instance we have in the signature Σ_I two sorts called `INDEX`, $\texttt{INDEX}^2$, together with pairing and projection function symbols, and suppose that the sort $\texttt{INDEX}^2$ is constrained to be interpreted in the models of T_I as the cartesian square of `INDEX` (notice that this is locally finite and closed under substructures, so it matches Definition 3.1). Now, an array variable a indexed by $\texttt{INDEX}^2$ with value in an enumerated datatype element sort `ELEM` represents an edge-labelled graph on `INDEX`: in fact $a[< i, j >] = e$ expresses the existence of an edge $i \xrightarrow{e} j$ labelled e between i and j (a special label, say 0, may represent the absence of an edge). This observation can be used to formalize the sophisticated (and more realistic) protocols with non-atomic global conditions recently introduced in [4].

There has been some attempts to use a purely logical techniques to check both safety and liveness properties of infinite state systems (e.g., [11, 15]). The hard part of these approaches is to guess auxiliary assertions (either invariants, for safety, or ranking functions, for liveness). Indeed, finding such auxiliary assertions is not easy and automation requires techniques adapted to specific domains (e.g., integers). In this paper, we have shown how our approach does not require auxiliary invariants for safety and proved that the search for certain progress conditions can be fully mechanized (Theorem 6.5).

In order to test the viability of our approach, we have built a prototype implementation of the backward reachability algorithm of Figure 2. The function A_I^E-check has been implemented by using a naive instantiation algorithm and a state-of-the art SMT solver. We have also implemented the generation of the proof obligations (10) and (11) for recognizing

[24] As a precursor of our approach, we can quote e.g. [16], where however formal developments leading to the relevant technical results are not pursued.

progress conditions. This allowed us to automatically verify all the examples discussed in this paper. We are currently planning to build a robust implementation of our techniques.

References

[1] Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*, pages 313–321, New Brunswick (NJ, USA), 1996. IEEE Computer Society Press.

[2] Parosh Aziz Abdulla, Giorgio Delzanno, Noomene Ben Henda, and Ahmed Rezine. Regular model checking without transducers (on efficient verification of parameterized systems). In O. Grumberg and M. Huth, editors, *Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007)*, volume 4424 of *Lecture Notes in Computer Science*, pages 721–736, Braga (Portugal), 2007. Springer.

[3] Parosh Aziz Abdulla, Giorgio Delzanno, and Ahmed Rezine. Parameterized verification of infinite-state processes with global conditions. In W. Damm and H. Hermanns, editors, *Proceedings of the 19th International Conference on Computer Aided Verification (CAV 2007)*, volume 4590 of *Lecture Notes in Computer Science*, pages 145–157, Berlin (Germany), 2007. Springer.

[4] Parosh Aziz Abdulla, Noomene Ben Henda, Giorgio Delzanno, and Ahmed Rezine. Handling parameterized systems with non-atomic global conditions. In F. Logozzo, D. Peled, and L. D. Zuck, editors, *Proceedings of the 9th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2008)*, volume 4905 of *Lecture Notes in Computer Science*, pages 22–36, San Francisco (CA, USA), 2008. Springer.

[5] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[6] Franz Baader and Silvio Ghilardi. Connecting many-sorted theories. *Journal of Symbolic Logic*, 72:535–583, 2007.

[7] Ahmed Bouajjani, Peter Habermehl, Yan Jurski, and Mihaela Sighireanu. Rewriting systems with data. In E. Csuhaj-Varjú and Z. Ésik, editors, *Proceedings of the 16th International Symposium on Fundamentals of Computation Theory (FCT 2007)*, volume 4639 of *Lecture Notes in Computer Science*, pages 1–22, Budapest (Hungary), 2007. Springer.

[8] Marco Bozzano, Roberto Bruttomesso, Alessandro Cimatti, Tommi A. Junttila, Silvio Ranise, Peter van Rossum, and Roberto Sebastiani. Efficient theory combination via boolean search. *Information and Computation*, 204(10):1493–1525, 2006.

[9] Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. What's decidable about arrays? In E. A. Emerson and K. S. Namjoshi, editors, *Proceedings of the 7th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2006)*, volume 3855 of *Lecture Notes in Computer Science*, pages 427–442, Charleston (SC, USA), 2006. Springer.

[10] Chen-Chung Chang and Jerome H. Keisler. *Model Theory*. North-Holland, Amsterdam-London, third edition, 1990.

[11] Leonardo M. de Moura, Harald Rueß, and Maria Sorea. Lazy theorem proving for bounded model checking over infinite domains. In A. Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction (CADE 2002)*, volume 2392 of *Lecture Notes in Computer Science*, pages 438–455, Copenhagen (Denmark), 2002. Springer.

[12] Giorgio Delzanno, Javier Esparza, and Andreas Podelski. Constraint-based analysis of broadcast protocols. In J. Flum and M. Rodríguez-Artalejo, editors, *Proceedings of the 13th International Workshop on Computer Science Logic (CSL 1999)*, volume 1683 of *Lecture Notes in Computer Science*, pages 50–66, Madrid (Spain), 1999. Springer.

[13] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, New York-London, 1972.

[14] Javier Esparza, Alain Finkel, and Richard Mayr. On the verification of broadcast protocols. In *Proceedints of the 14th Annual IEEE Symposium on Logic in Computer Science (LICS 1999)*, pages 352–359, Trento (Italy), 1999. IEEE Computer Society.

[15] Yi Fang, Nir Piterman, Amir Pnueli, and Lenore D. Zuck. Liveness with invisible ranking. *Software Tools for Technology*, 8(3):261–279, 2006.

[16] Xiang Fu, Tevfik Bultan, and Jianwen Su. Formal verification of e-services and workflows. In C. Bussler, R. Hull, S. A. McIlraith, M. E. Orlowska, B. Pernici, and J. Yang, editors, *Revised Papers of the CAiSE 2002 International Workshop on Web Services, E-Business, and the Semantic Web (WES 2002)*, volume 2512 of *Lecture Notes in Computer Science*, pages 188–202, Toronto (Canada), 2002. Springer.

[17] Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Combination methods for satisfiability and model-checking of infinite-state systems. In *Proceedings of the 21st International Conference on Automated Deduction Automated Deduction (CADE 2007)*, volume 4603 of *Lecture Notes in Computer Science*, pages 362–378, Bremen (Germany), 2007. Springer.

[18] Carsten Ihlemann, Swen Jacobs, and Viorica Sofronie-Stokkermans. On local reasoning in verification. In C. R. Ramakrishnan and J. Rehof, editors, *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*, volume 4963 of *Lecture Notes in Computer Science*, pages 265–281, Budapest (Hungary), 2008. Springer.

[19] Yonit Kesten, Oded Maler, Monica Marcus, Amir Pnueli, and Elad Shahar. Symbolic model checking with rich assertional languages. *Theoretical Computer Science*, 256(1-2):93–112, 2001.

[20] Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[21] Zohar Manna and Amir Pnueli. A hierarchy of temporal properties. In *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing (PODC 1990)*, pages 377–410, Quebec City (Canada), 1990. ACM Press.

[22] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving SAT and SAT Modulo Theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, 2006.

[23] Schnoebelen Philippe. Verifying lossy channel systems has nonprimitive recursive complexity. *Information Processing Letters*, 83(5):251–261, 2002.

[24] Silvio Ranise and Cesare Tinelli. The SMT-LIB standard: Version 1.2. Technical report, 2006. Available at `http://combination.cs.uiowa.edu/smtlib/papers/format-v1.2-r06.08.30.pdf`.

[25] Tatiana Rybina and Andrei Voronkov. A logical reconstruction of reachability. In M. Broy and A. V. Zamulin, editors, *Revised Papers of the 5th International Andrei Ershov Memorial Conference on Perspectives of Systems Informatics (PSI 2003)*, volume 2890 of *Lecture Notes in Computer Science*, pages 222–237, Akademgorodok (Russia), 2003. Springer.

[26] Roberto Sebastiani. Lazy satisfiability modulo theories. *Journal on Satisfiability, Boolean Modeling and Computation*, 3:141–224, 2007.

A Appendix: Proofs of the Main Results

In this appendix, we collect the proofs of the technical results of the paper, by grouping them according to the section in which they were stated.

Results from Section 4

The following result implies Theorem 4.1 and proves also soundness and correctness of the procedure A_I^E-check of Figure 1:[25]

Theorem A.1. *The A_E^I-satisfiability of a sentence of the kind*

$$\exists a_1 \cdots \exists a_n \; \exists \underline{i} \; \exists \underline{e} \; \forall \underline{j} \; \psi(\underline{i}, \underline{j}, \underline{e}, a_1[\underline{i}], \ldots, a_n[\underline{i}], a_1[\underline{j}], \ldots, a_n[\underline{j}]) \tag{17}$$

is decidable. Moreover, the following conditions are equivalent:

(i) *the sentence* (17) *is A_E^I-satisfiable;*

(ii) *the sentence* (17) *is satisfiable in a finite index model of A_E^I;*

(iii) *the sentence*

$$\exists a_1 \cdots \exists a_n \; \exists \underline{i} \; \exists \underline{e} \bigwedge_{\sigma} \psi(\underline{i}, \underline{j}\sigma, \underline{e}, a_1[\underline{i}], \ldots, a_n[\underline{i}], a_1[\underline{j}\sigma], \ldots, a_n[\underline{j}\sigma]) \tag{18}$$

is A_E^I-satisfiable (here σ ranges onto the substitutions mapping the variables $\underline{j}$ into the set of representative $\Sigma_I(\underline{i})$-terms and $\underline{j}\sigma$ means the simultaneous application of σ to the variables of the tuple $\underline{j}$).

Proof. In order to prevent notational complications, we limit to the case $n = 1$ (the reader may check that there is no loss of generality in that).[26] We first show that A_E^I-satisfiability of

$$\exists a \; \exists \underline{i} \; \exists \underline{e} \; \forall \underline{j} \; \psi(\underline{i}, \underline{j}, \underline{e}, a[\underline{i}], a[\underline{j}]) \tag{19}$$

is equivalent to A_E^I-satisfiability of

$$\exists a \; \exists \underline{i} \; \exists \underline{e} \bigwedge_{\sigma} \psi(\underline{i}, \underline{j}\sigma, \underline{e}, a[\underline{i}], a[\underline{j}\sigma]) \tag{20}$$

[25] The sentences of the kind (17) are slightly more general than the $\exists^{A,I}\forall^I$-sentences mentioned in Theorem 4.1, because elements quantifiers $\exists \underline{e}$ occur now. We do not need decidability of this larger class of sentences in the paper, but we nevertheless report this more general result too.

[26] Since the addition of vacuous existential quantifiers does not affect satisfiability, we can also assume that the tuple $\underline{i}$ is not empty (this observation is needed if we want to prevent the structure $\mathcal{N}$ defined below from having empty index domain).

where σ ranges onto the substitutions mapping the variables $\underline{j}$ into the set of representative $\Sigma_I(\underline{i})$-terms.

That A_E^I-satisfiability of (19) implies A_E^I-satisfiability of (20) follows from trivial logical manipulations, so let us assume A_E^I-satisfiability of (20) and show A_E^I-satisfiability of (19). Let $\mathcal{M}$ be a model of (20); we can assign elements in this model to the variables $a, \underline{i}, \underline{e}$ in such a way that (under such an assignment $\mathtt{a}$) we have $\mathcal{M}, \mathtt{a} \models \bigwedge_\sigma \ \psi(\underline{i}, \underline{j}\sigma, \underline{e}, a[\underline{i}], a[\underline{j}\sigma])$. Consider the model $\mathcal{N}$ which is obtained from $\mathcal{M}$ by restricting the interpretation of the sort `INDEX` (and of all function and relation symbols for indexes) to the Σ_I-substructure generated by the elements assigned by $\mathtt{a}$ to the $\underline{i}$: since models of T_I are closed under substructures, this substructure is a model of T_I and consequently $\mathcal{N}$ is still a model of A_E^I. Now let s be the restriction of $\mathtt{a}(a)$ to the new smaller index domain and let $\tilde{\mathtt{a}}$ be the assignment differing from $\mathtt{a}$ only for assigning s to a (instead of $\mathtt{a}(a)$); since ψ is quantifier free and since, varying σ, the elements assigned to the terms $\underline{j}\sigma$ covers all possible $\underline{j}$-tuples of elements in the interpretation of the sort `INDEX` in $\mathcal{N}$, we have $\mathcal{N}, \tilde{\mathtt{a}} \models \forall \underline{j} \psi(\underline{i}, \underline{j}, a[\underline{i}], a[\underline{j}])$. This shows that $\mathcal{N} \models \exists a \ \exists \underline{i} \ \forall \underline{j} \ \psi(\underline{i}, \underline{j}, a[\underline{i}], a[\underline{j}])$, i.e that (19) holds. Notice that $\mathcal{N}$ is a finite index model,[27] hence we proved also the equivalence between (i) and (ii).

We now need to decide A_E^I-satisfiability of sentences (20). Let $\underline{t}$ be the representative $\Sigma(\underline{i})$-terms and let us put them in bijective correspondence with fresh variables $\underline{l}$ of sort `INDEX`; let $\psi_\sigma(\underline{i}, \underline{l}, \underline{e}, a[\underline{i}], a[\underline{l}])$ be the formula obtained by replacing in $\psi(\underline{i}, \underline{j}\sigma, \underline{e}, a[\underline{i}], a[\underline{j}\sigma])$ the $\Sigma(\underline{i})$-terms $\underline{j}\sigma$ by the corresponding $\underline{l}$. We first rewrite (20) as

$$\exists a \ \exists \underline{i} \ \exists \underline{e} \ \exists \underline{l} \ (\underline{l} = \underline{t} \wedge \bigwedge_\sigma \psi_\sigma(\underline{i}, \underline{l}, \underline{e}, a[\underline{i}], a[\underline{l}])) \tag{21}$$

Notice that T_I and T_E are disjoint (they do not have even any sort in common), which means that $\underline{l} = \underline{t} \wedge \bigwedge_\sigma \psi_\sigma(\underline{i}, \underline{l}, \underline{e}, a[\underline{i}], a[\underline{l}])$ is a boolean combination of Σ_I-atoms and of Σ_E-atoms (in the latter kind of atoms, the variables for elements are replaced by the terms $a[\underline{i}], a[\underline{l}]$). This means that our decision problem can be further rephrased in terms of the problem of deciding for A_E^I-satisfiability formulae like

$$\psi_I(\underline{j}) \wedge a[\underline{j}] = \underline{d} \wedge \psi_E(\underline{d}, \underline{e}) \tag{22}$$

where $\psi_I(\underline{j})$ is a conjunction of Σ_I-literals and $\psi_E(\underline{e})$ is a conjunction of Σ_E-literals.[28]

Since we are looking for a model of T_I, a model of T_E and for a function connecting their domains (the function interpreting the variable a), this is a satisfiability problem for a theory

[27] This is because T_I is locally finite and the Σ_I reduct of $\mathcal{N}$ is a structure which is generated by finitely many elements.

[28] We renamed the concatenation of the tuples $\underline{i}, \underline{l}$ as $\underline{j}$, to have a more compact notation.

connection (in the sense of [6]):[29] since the signatures of T_I, T_E are disjoint, the problem is decided by propagating equalities.[30] In other words, to decide (22), just apply the following steps:

- guess an equivalence relation Π on the index variables $\underline{j}$ (let us assume $\underline{j} = j_1, \ldots . j_n$);
- check $\psi_I(\underline{j}) \cup \{j_k = j_l \mid (j_k, j_l) \in \Pi\} \cup \{j_k \neq j_l \mid (j_k, j_l) \notin \Pi\}$ for T_I-satisfiability;
- check $\psi_E(\underline{d}, \underline{e}) \cup \{d_k = d_l \mid (j_k, j_l) \in \Pi\}$ for T_E-satisfiability;
- return 'unsatisfiable' iff failure is reported in the previous two steps for all possible guesses.

Soundness and completeness of the above procedure are easy. □

Results from Section 5

Lemma A.2. *Every formula of the kind $\exists \underline{e}\, \phi(\underline{i}, a[\underline{i}], \underline{e})$ is A_E^I-equivalent to a (effectively computable) quantifier-free formula $\psi(\underline{i}, a[\underline{i}])$.*

Proof. We show that for every formula of the kind $\exists \underline{e}\, \phi(\underline{i}, \underline{d}, \underline{e})$ there is a quantifier-free formula $\psi(\underline{i}, \underline{d})$ such that $A_E^I \models \psi(\underline{i}, \underline{d}) \leftrightarrow \exists \underline{e}\, \phi(\underline{i}, \underline{d}, \underline{e})$ (the result then follows by replacing $\underline{d}$ with $a[\underline{i}]$).

By distributing existential quantifiers over disjunctions, we can freely assume that $\phi(\underline{i}, \underline{d}, \underline{e})$ is primitive, i.e. that it is a conjunction of $\Sigma_I \cup \Sigma_E$-literals; since the variables $\underline{e}$ do not occur in Σ_I-literals, we can further push the existential quantifiers so that their scope is a conjunction of Σ_E-literals and finally apply quantifier elimination in T_E. □

We begin by proving Proposition 5.3:

Proposition 5.3. *Let $K(a)$ be an $\exists^I$-formula; then $Pre(\tau, K)$ is A_I^E-equivalent to an (effectively computable) $\exists^I$-formula $K'(a)$.*

Proof. We can freely assume that τ consists of a single disjunction of a T-formula: from this special case, the general case follows immediately because $\exists^I$-formulae are closed under disjunctions (modulo trivial logical manipulations).

Let $K(a)$ be $\exists \underline{k}\, \phi(\underline{k}, a[\underline{k}])$ and let $\tau(a, a')$ be

$$\exists \underline{j}\ (\phi_L(\underline{j}, a[\underline{j}], a'[\underline{j}]) \wedge \forall l\ \phi_G(\underline{j}, a[\underline{j}], a'[\underline{j}], l, a[l], a'[\underline{l}])).$$

[29] Strictly speaking, one cannot directly apply the results from [6], because we have a 'semantically driven' notion of a theory (otherwise said, the classes of the models we consider need not be elementary).

[30] This is different from disjoint Nelson-Oppen combination, where also inequalities must be propagated.

In the following, we shall use the abbreviation[31]

$$Up_G(\underline{j}, a, a') \;:\Leftrightarrow\; \forall l\; \phi_G(\underline{j}, a[\underline{j}], a'[\underline{j}], l, a[l], a'[\underline{l}]). \tag{23}$$

We need to show that a formula like

$$\exists a'\, \exists \underline{j}\, \exists \underline{k}\, \big(\phi_L(\underline{j}, a[\underline{j}], a'[\underline{j}]) \wedge Up_G(\underline{j}, a, a') \wedge \phi(\underline{k}, a'[\underline{k}])\big)$$

is A_E^I-equivalent to an $\exists^I$-formula. We proceed by equivalent transformations as follows: we first replace the terms $a'[\underline{j}]$ in the subformula $\phi_L(\underline{j}, a[\underline{j}], a'[\underline{j}])$ by the fresh constants $\underline{e}$, thus getting

$$\exists a'\, \exists \underline{j}\, \exists \underline{k}\, \exists \underline{e}\, \big(\underline{e} = a'[\underline{j}] \wedge \phi_L(\underline{j}, a[\underline{j}], \underline{e}) \wedge Up_G(\underline{j}, a, a') \wedge \phi(\underline{k}, a'[\underline{k}])\big)$$

We also explicitly name the $a'[\underline{k}]$ as $\underline{d}$ and get

$$\exists a'\, \exists \underline{j}\, \exists \underline{k}\, \exists \underline{e}\, \exists \underline{d}\, \big(\underline{e} = a'[\underline{j}] \wedge \underline{d} = a'[\underline{k}] \wedge \phi_L(\underline{j}, a[\underline{j}], \underline{e}) \wedge Up_G(\underline{j}, a, a') \wedge \phi(\underline{k}, \underline{d})\big)$$

Now the formulae $\underline{e} = a'[\underline{j}]$ and $Up_G(\underline{j}, a, a')$, according to (23), entail $\bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r)$, where the index r ranges over the components of the tuple $\underline{j}$; similarly, the formulae $\underline{e} = a'[\underline{j}] \wedge \underline{d} = a'[\underline{k}]$ and $Up_G(\underline{j}, a, a')$ entail the formula $\bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s)$, with the index s ranging over the components of the tuple $\underline{k}$. In we insert this extra information into our formula, we get

$$\exists a'\, \exists \underline{j}\, \exists \underline{k}\, \exists \underline{e}\, \exists \underline{d}\, \Big(\underline{e} = a'[\underline{j}] \wedge \underline{d} = a'[\underline{k}] \wedge \phi_L(\underline{j}, a[\underline{j}], \underline{e}) \wedge Up_G(\underline{j}, a, a') \wedge$$
$$\wedge\, \phi(\underline{k}, \underline{d}) \wedge \bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r) \wedge \bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s)\Big)$$

Rearranging the conjuncts and pushing inside the existential quantifiers $\exists a'$, we obtain

$$\exists \underline{j}\, \exists \underline{k}\, \exists \underline{e}\, \exists \underline{d}\, \Big(\phi_L(\underline{j}, a[\underline{j}], \underline{e}) \wedge \phi(\underline{k}, \underline{d}) \wedge \bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r) \wedge$$
$$\wedge \bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s) \wedge \exists a'\big(\underline{e} = a'[\underline{j}] \wedge \underline{d} = a'[\underline{k}] \wedge Up_G(\underline{j}, a, a')\big)\Big)$$

Consider now a partition Π on the set $\underline{j} \cup \underline{k}$ and let $\underline{j}_0 \cup \underline{k}_0$ be a choice of representative elements: we pick one element from each equivalence class, thus we have $\underline{j}_0 \subseteq \underline{j}$, $\underline{k}_0 \subseteq \underline{k}$.[32] Let $\Pi(\underline{j}, \underline{e}, \underline{k}, \underline{d})$ be the formula saying that each index is equal to the representative index of its own equivalence class, that representative indexes are distinct from each other and that

[31] Modulo the alphabetic change $\underline{i} \mapsto \underline{j}$, this is the same abbreviation as (3) from Definition 3.4. We only preferred to write Up_G instead of $Update_G$ to save space when writing long formulae.

[32] Nothing prevents from $\underline{j}_0$ or $\underline{k}_0$ to be in principle empty (e.g. $\underline{j}_0$ is empty in case all representatives are in $\underline{k}$).

elements in $\underline{e} \cup \underline{d}$ corresponding to equivalent indexes are equal too. Our formula is equivalent to the disjunction

$$\bigvee_{\Pi} \exists \underline{j}\, \exists \underline{k}\, \exists \underline{e}\, \exists \underline{d} \Big(\phi_L(\underline{j}, a[\underline{j}], \underline{e}) \wedge \phi(\underline{k}, \underline{d}) \wedge \Pi(\underline{j}, \underline{e}, \underline{k}, \underline{d}) \wedge \bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r) \wedge \\ \wedge \bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s) \wedge \exists a' \big(\underline{e}_0 = a'[\underline{j}_0] \wedge \underline{d}_0 = a'[\underline{k}_0] \wedge Up_G(\underline{j}, a, a') \big) \Big)$$

where $\underline{e}_0, \underline{d}_0$ componentwise match $\underline{i}_0, \underline{j}_0$, respectively.

Now notice that the formula

$$\Pi(\underline{j}, \underline{e}, \underline{k}, \underline{d}) \wedge \bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r) \wedge \bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s) \rightarrow \\ \rightarrow \exists a' (\underline{e}_0 = a'[\underline{j}_0] \wedge \underline{d}_0 = a'[\underline{k}_0] \wedge Up_G(\underline{j}, a, a')) \qquad (\star)$$

is a logical consequence of A_E^I: to see it, argue as follows. Fix an assignment in a model $\mathcal{M}$ of A_E^I making the antecedents of $(\star)$ true (for simplicity, we denote the elements assigned to $a, \underline{j}, \underline{k}, \underline{e}, \underline{d}$ again by $a, \underline{j}, \underline{k}, \underline{e}, \underline{d}$, respectively); we define the array a' as follows. We first write the $\underline{e}_0$ in the positions $\underline{i}_0$ and the $\underline{d}_0$ in the positions $\underline{k}_0$ (this is well-defined by the fact that the $\underline{j}_0, \underline{k}_0$ are pairwise distinct, as implied by the truth of the antecedent $\Pi(\underline{j}, \underline{e}, \underline{k}, \underline{d})$); for the indices l which are different from $\underline{i}_0$ and $\underline{k}_0$, we define $a'(l)$ by taking an element e' satisfying $\phi_G(\underline{j}, a[\underline{j}], \underline{e}, l, a[l], e')$ in $\mathcal{M}$ (this e' exists by the seriality condition (1)). This a' fits the desired requirements because the antecedents of $(\star)$ are true. In more detail, notice that:

(i) $\underline{e}_0 = a'[\underline{j}_0] \wedge \underline{d}_0 = a'[\underline{k}_0]$ is true by construction;

(ii) $\underline{e} = a'[\underline{j}] \wedge \underline{d} = a'[\underline{k}]$ follows from (i) and the fact that $\Pi(\underline{j}, \underline{e}, \underline{k}, \underline{d})$ holds;

(iii) $\bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r)$ and (ii) imply $\bigwedge_r \phi_G(\underline{j}, a[\underline{j}], a'[\underline{j}], \underline{j}_r, a[\underline{j}_r], a'[\underline{j}_r])$;

(iv) $\bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s)$ and (ii) imply $\bigwedge_s \phi_G(\underline{j}, a[\underline{j}], a'[\underline{j}], \underline{k}_s, a[\underline{k}_s], a'[\underline{k}_s])$;

(v) from (ii), (iii), (iv), and the definition of a', we get $\phi_G(\underline{j}, a[\underline{j}], a'[\underline{j}], l, a[l], a'[l])$ for every l, which means that $Up_G(\underline{j}, a, a')$ holds.

Having established that $(\star)$ is A_E^I-valid, we can simplify our big disjunction to

$$\bigvee_{\Pi} \exists \underline{j}\, \exists \underline{k}\, \exists \underline{e}\, \exists \underline{d} \Big(\phi_L(\underline{j}, a[\underline{j}], \underline{e}) \wedge \phi(\underline{k}, \underline{d}) \wedge \Pi(\underline{j}, \underline{e}, \underline{k}, \underline{d}) \wedge \\ \wedge \bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r) \wedge \bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s) \Big)$$

Since the quantifiers $\exists \underline{e}\, \exists \underline{d}$ can be eliminated by Lemma A.2, we eventually get a formula of the desired shape, namely of the kind

$$\bigvee_{\Pi} \exists \underline{j}\, \exists \underline{k}\, \psi_{\Pi}(\underline{j}, \underline{k}, a[\underline{j}], a[\underline{k}])$$

(notice also that $\exists^I$-formulae are closed under disjunctions, modulo trivial logical manipulations). □

Observe that, according to the above proof, the formula $Pre(\tau, K)$ may have an existential quantifier prefix longer than K.

Remark A.3. Quite often, in the applications, the global component formulae ϕ_G are not only serial but also functional, meaning that the uniqueness requirement (4) is satisfied. In that case, the algorithm suggested by the proof of Proposition 5.3 greatly simplifies (there is no need of taking a disjunction over the partitions on indexes). Since this simplified procedure can be very useful in the implementations, we report it. The key observation is that now we can prove, instead of $(\star)$, directly the stronger entailment

$$\begin{aligned} \bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r) \wedge \bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s) \rightarrow \\ \rightarrow \exists a'(\underline{e} = a'[\underline{j}] \wedge \underline{d} = a'[\underline{k}] \wedge Up_G(\underline{j}, a, a')) \end{aligned} \qquad (\star\star)$$

To show $(\star\star)$, define a' as follows (again let a model $\mathcal{M}$ and elements $a, \underline{j}, \underline{k}, \underline{e}, \underline{d}$ from its supports be fixed, so that the antecedent of $(\star\star)$ is true): for every l, we let $a'(l)$ be the unique e' so that $\phi_G(\underline{j}, a[\underline{j}], \underline{e}, l, a[l], e')$ is true in $\mathcal{M}$. Now $\bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r)$ implies $a'[\underline{j}] = \underline{e}$ by the uniqueness requirement (4); similarly $\bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s)$ implies $a'[\underline{k}] = \underline{d}$. Having established that $a'[\underline{j}] = \underline{e}$ holds, by the construction of a', we can also get $\phi_G(\underline{j}, a[\underline{j}], a'[\underline{j}], l, a[l], a'[l])$ for every l, which means that $Up_G(\underline{j}, a, a')$ holds.

Once $(\star\star)$ is proved, the formula

$$\begin{aligned} \exists \underline{j}\, \exists \underline{k}\, \exists \underline{e}\, \exists \underline{d}\, \Big(\phi_L(\underline{j}, a[\underline{j}], \underline{e}) \wedge \phi(\underline{k}, \underline{d}) \wedge \bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r) \wedge \\ \wedge \bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s) \wedge \exists a' \left(\underline{e} = a'[\underline{j}] \wedge \underline{d} = a'[\underline{k}] \wedge Up_G(\underline{j}, a, a') \right) \Big) \end{aligned}$$

directly simplifies to

$$\begin{aligned} \exists \underline{j}\, \exists \underline{k}\, \exists \underline{e}\, \exists \underline{d}\, \Big(\phi_L(\underline{j}, a[\underline{j}], \underline{e}) \wedge \phi(\underline{k}, \underline{d}) \wedge \\ \wedge \bigwedge_r \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{j}_r, a[\underline{j}_r], \underline{e}_r) \wedge \bigwedge_s \phi_G(\underline{j}, a[\underline{j}], \underline{e}, \underline{k}_s, a[\underline{k}_s], \underline{d}_s) \Big) \end{aligned}$$

which has the desired shape, once elements quantifiers have been eliminated.

The proof of Theorem 5.4 reduces to few simple observations which have already been made in Section 5. In detail, notice that: (i) thanks to Proposition 5.3 above, $BR^n(\tau, K)$ is A_E^I-equivalent to an $\exists^I$-formula; (ii) after prenexing quantifiers, the consistency of $J \wedge H$ (where J is a $\forall^I$-formula and H is a $\exists^I$-formula) falls within the decision procedure of Theorem 4.1. Observations (i)-(ii) guarantee that the algorithm of Figure 1 is a real algorithm (in the sense that the A_E^I-consistency tests it requires are effective), which is precisely what we need to prove.

We now supply proofs for Propositions 5.6 and Theorem 5.7:

Proposition 5.6. *For every $\exists^I$-formula $K(a)$, the set $[\![K]\!]$ is upward closed; moreover, for every $\exists^I$-formulae K_1, K_2, we have $[\![K_1]\!] \subseteq [\![K_2]\!]$ iff $A_I^E \models K_1 \rightarrow K_2$.*

Proof. Let us first show that the set $[\![K]\!]$ is upward closed. By using disjunctive normal forms and by distributing existential quantifiers over disjunctions, we can freely suppose that $K(a)$ is of the kind $\exists \underline{i} \phi(\underline{i}, a[\underline{i}])$,[33] where ϕ is a conjunction of $\Sigma_I \cup \Sigma_E$-literals. If we also separate Σ_I- and Σ_E-literals, we can suppose that $\phi(\underline{i}, a[\underline{i}])$ is of the kind $\phi_I(\underline{i}) \wedge \phi_E(a[\underline{i}])$, where ϕ_I is a conjunction of Σ_I-literals and ϕ_E is a conjunction of Σ_E-literals. Suppose now that $(\mathcal{M}, s)$ and $(\mathcal{N}, t)$ are configurations such that $s \leq t$ and $\mathcal{M} \models K(s)$: we wish to prove that $\mathcal{N} \models K(t)$. From $\mathcal{M} \models K(s)$, it follows that there are elements $\underline{i}$ from $\texttt{INDEX}^{\mathcal{M}}$ such that $\mathcal{M} \models \phi_I(\underline{i}) \wedge \phi_E(s[\underline{i}])$, i.e. such that $s_I \models \phi_I(\underline{i})$ and $s_E \models \phi_E(s(\underline{i}))$ (to infer the latter, recall that the operations $a[\underline{i}]$ are interpreted as functional applications in our models and also that truth of quantifier free formulae is preserved when passing to substructures). Now $s \leq t$ says that there are embeddings $\mu : s_I \longrightarrow t_I$ and $\nu : s_E \longrightarrow t_E$ such that $\nu \circ s = t \circ \mu$. Since truth of quantifier free formulae is preserved when passing to superstructures, we get $t_I \models \phi_I(\mu(\underline{i}))$ and $t_E \models \phi_E(\nu(s(\underline{i}))$ (that is, $t_E \models \phi_E(t(\mu(\underline{i})))$ and also $\mathcal{N} \models \phi_I(\mu(\underline{i})) \wedge \phi_E(t[\mu(\underline{i})])$, which implies $\mathcal{N} \models K(t)$, as desired.

Let us now prove the second claim of the Proposition. That $A_E^I \models K_1 \rightarrow K_2$ implies $[\![K_1]\!] \subseteq [\![K_2]\!]$ is trivial. Suppose conversely that $A_E^I \not\models K_1 \rightarrow K_2$, which means that $K_1(a) \wedge \neg K_2(a)$ is A_E^I-satisfiable: since, by Lemma A.1(ii), this implies that $K_1(a) \wedge \neg K_2(a)$ is satisfiable in a finite index model of A_E^I, we immediately get that $[\![K_1]\!] \not\subseteq [\![K_2]\!]$. □

Before continuing, we need some standard model-theoretic background on Robinson diagrams [10]. Let $\mathcal{M} = (M, \int)$ be a Σ-structure which is generated by $G \subseteq M$. Let us take a free variable x_g for every $g \in G$[34] and let us call G_x the set $\{x_g \mid g \in G\}$. The Σ_G-*diagram*

[33] Notice that a union of upsets is an upset.

[34] One may complain because there are only countable many variables and G may not be countable: this trouble can be disposed of either by using free constants instead of variables (this is the standard approach), or by realizing that we won't need in the paper an uncountable G (actually, in all our applications, G is always

$\delta_{\mathcal{M}}(G)$ *of* $\mathcal{M}$ is the set of all $\Sigma(G_x)$-literals L such $\mathcal{M}, \mathtt{a} \models L$, where $\mathtt{a}$ is the assignment mapping x_g to g.

The following celebrated result [10] is simple, but nevertheless very powerful:

Lemma A.4 (Robinson Diagram Lemma)**.** *Let* $\mathcal{M} = (M, \int)$ *be a* Σ*-structure which is generated by* $G \subseteq M$ *and let* $\mathcal{N} = (N, \mathcal{J})$ *be a further* Σ*-structure. Then there is a bijective correspondence given by*

$$\mu(g) = \mathtt{a}(g) \tag{24}$$

(for all $g \in G$*)*[35] *between assignments* $\mathtt{a}$ *on* N *such that* $\mathcal{N}, \mathtt{a} \models \delta_{\mathcal{M}}(G)$ *and* Σ*-embeddings* $\mu : \mathcal{M} \longrightarrow \mathcal{N}$*.*

The diagram $\delta_{\mathcal{M}}(G)$ usually contains infinitely many literals, however there are important cases where we can keep it finite:

Lemma A.5. *Suppose that* $\mathcal{M}$ *is a* Σ*-structure (where* Σ *is a finite signature), whose support* M *is finite; then for every set of generators* $G \subseteq M$*, there are finitely many literals from* $\delta_{\mathcal{M}}(G)$ *having all remaining literals of* $\delta_{\mathcal{M}}(G)$ *as a logical consequence.*

Proof. Choose $\Sigma(G_x)$-terms $t_1, \ldots, t_n$ such that (under the assignment $\mathtt{a} : x_g \mapsto g$), M is equal to the set of the elements assigned by $\mathtt{a}$ to $t_1, \ldots, t_n$ (this is possible because the G are generators and M is finite); we also include the x_g varying $g \in G$ among the $t_1, \ldots, t_n$. We can get the desired finite set S of literals by taking the set of atoms of the form

$$R(t_{i_1}, \ldots, t_{i_k}), \quad f(t_{i_1}, \ldots, t_{i_k}) = t_{i_{k+1}}$$

(as well as their negations), which are true in $\mathcal{M}$ under the assignment $\mathtt{a}$. In fact, modulo S, it is easy to see by induction on u that every $\Sigma(G_x)$-term u is equal to some t_i; it follows that every literal from $\delta_{\mathcal{M}}(G)$ is a logical consequence of S. □

Whenever the conditions of the above Lemma are true, we can take a finite conjunction and treat $\delta_{\mathcal{M}}(G)$ as a single formula: notice that we are allowed to do so whenever G is finite and $\mathcal{M}$ is a model of a locally finite theory.

Proposition A.6. *Suppose that* T_E *is locally finite; the following hold:*

(i) *with every* A^I_E*-configuration* $(\mathcal{M}, s)$ *one can effectively associate an* $\exists^I$*-formula* K_s *such that* $[\![K_s]\!] = \{s' \mid s \leq s'\}$*;*

finite).

[35] In other words, (24) can be used to define μ from $\mathtt{a}$ and conversely. Notice that an embedding $\mu : \mathcal{M} \longrightarrow \mathcal{N}$ is uniquely determined, in case it exists, by the image of the set of generators G: this is because the fact that G generates $\mathcal{M}$ implies (and is equivalent to) the fact that every $c \in M$ is of the kind $t^{\int}(\underline{g})$, for some term t and some $\underline{g}$ from G.

(ii) *with every $\exists^I$-formula K one can effectively associate a finite set $\{s_1, \ldots, s_n\}$ of A^I_E-configurations such that K is A^I_E-equivalent to $K_{s_1} \vee \cdots \vee K_{s_n}$.*

As a consequence of (i)-(ii)*, finitely generated upsets of A^I_E-configurations coincide with sets of A^I_E-configurations of the kind $[\![K]\!]$ (for some $\exists^I$-formula K).*

Proof. Ad (i): we take G, G' to be the support of s_I and the image of the support of s_I under the function s, respectively; clearly G is a set of generators for s_I and G' is a set of generators for s_E. Let us call the set of variables G_x, G'_x as $\underline{i} := \{i_1, \ldots, i_n\}$ and $\underline{e} := \{e_1, \ldots, e_n\}$, respectively. We take K_s to be

$$\exists \underline{i} \, (\delta_{s_I}(\underline{i}) \wedge \delta_{s_E}(a_0[\underline{i}]))$$

(in other words, we take the diagrams $\delta_{s_I}(G), \delta_{s_E}(G')$, make in the latter the replacement $\underline{e} \mapsto a_0[\underline{i}]$, take conjunction, and quantify existentially over the $\underline{i}$). For every configuration $(\mathcal{N}, t)$, we have that $t \in [\![K_s]\!]$ iff $\delta_{s_I}(\underline{i}) \wedge \delta_{s_E}(a_0[\underline{i}])$ is true in $\mathcal{N}$ under some assignment $\mathtt{a}$ mapping the array variable a_0 to t, that is iff that there are embeddings $\mu : s_I \longrightarrow t_I$ and $\nu : s_E \longrightarrow t_E$ as prescribed by Robinson Diagram Lemma. These embeddings map the generators G onto the indexes assigned to the $\underline{i}$ by $\mathtt{a}$ and the generators G' to the elements assigned by $\mathtt{a}$ to the terms $a_0[\underline{i}]$, which means precisely that $t \circ \mu = \nu \circ s$. Thus $t \in [\![K_s]\!]$ is equivalent to $s \leq t$, as wanted.

Ad (ii): modulo taking disjunctive normal forms, we can suppose that $K(a_0)$ is $\exists \underline{i} \bigvee_k (\phi_k(\underline{i}) \wedge \psi_k(a_0[\underline{i}]))$ (where the ϕ_k are Σ_I-formulae, the ψ_k are Σ_E-formulae and we let, for instance, $\underline{i}$ be $i_1, \ldots, i_m$). Now, in a locally finite theory, every quantifier free formula θ having at most m free variables, is equivalent to a disjunction of diagram formulae $\delta_{\mathcal{M}}(G)$, where $\mathcal{M}$ is a substructure of a model of the theory and G is a set of generators for $\mathcal{M}$ of cardinality at most m.[36] If we apply this to both T_I and T_E, we get that our $K(a_0)$ can be rewritten as

$$\bigvee_{\mathcal{A},\mathcal{B}} \exists \underline{i} \, (\delta_{\mathcal{A}}(\underline{i}) \wedge \delta_{\mathcal{B}}(a_0[\underline{i}]))$$

where $\mathcal{A}$ ranges over the m-generated models of T_I and $\mathcal{B}$ over the m-generated submodels of T_E.[37] Every such pair $(\mathcal{A}, \mathcal{B})$ is either A^I_E-inconsistent (in case some equality among the generators of $\mathcal{A}$ is not satisfied by the corresponding generators of $\mathcal{B}$) or it gives raise to a configuration a such that $\exists \underline{i} \, (\delta_{\mathcal{A}}(\underline{i}) \wedge \delta_{\mathcal{B}}(a_0[\underline{i}]))$ is precisely K_a. □

[36] Since the theory is locally finite, there are finitely many atoms whose free variables are included in a given set of cardinality m. Maximal conjunctions of literals built on these atoms are either inconsistent (modulo the theory) of satisfiable in an m-generated substructure of a model of the theory. By maximality, these maximal conjunctions must be diagrams.

[37] Recall that T_I is closed under substructures.

Theorem 5.7. *Assume that T_E is locally finite; let K be an $\exists^I$-formula. If K is safe, then* BReach *in Figure 2 terminates iff $\mathcal{B}(\tau, K)$ is a finitely generated upset.*

Proof. Suppose that $\mathcal{B}(\tau, K)$ is a finitely generated upset. Notice that

$$\mathcal{B}(\tau, K) = \bigcup_n [\![BR^n(\tau, K)]\!],$$

consequently (since we have $[\![BR^0(\tau, K)]\!] \subseteq [\![BR^1(\tau, K)]\!] \subseteq [\![BR^2(\tau, K)]\!] \subseteq \cdots$) we have $\mathcal{B}(\tau, K) = [\![BR^n(\tau, K)]\!] = [\![BR^{n+1}(\tau, K)]\!]$, which means by the second claim of Proposition 5.6 that $A_E^I \models BR^n(\tau, K) \leftrightarrow BR^{n+1}(\tau, K)$, i.e. that the Algorithm halts. Vice versa, if the Algorithm halts, we have $A_E^I \models BR^n(\tau, K) \leftrightarrow BR^{n+1}(\tau, K)$, hence $[\![BR^n(\tau, K)]\!] = [\![BR^{n+1}(\tau, K)]\!] = \mathcal{B}(\tau, K)$ and the upset $\mathcal{B}(\tau, K)$ is finitely generated by Proposition A.6. □

Remark A.7. In the model checking literature, an important property relating configuration ordering and transitions (called 'monotonicity' in [1]) ensures that the preimage of an upset is still an upset. Such a property does not appear in the above proofs (we do not need to mention it because we work symbolically with definable upsets), however it is possible to formulate it in our framework in the following way:

– let $(\mathcal{M}, s), (\mathcal{M}', s'), (\mathcal{M}, t)$ be configurations such that $s \leq s'$ and $\mathcal{M} \models \tau(s, t)$; then there exists $(\mathcal{M}', t')$ such that $t \leq t'$ and $\mathcal{M}' \models \tau(s', t')$.

The proof that such a property holds is easy and left to the reader (it basically depends on the seriality property (1) of our T-formulae).

In the paper, we used a definition of a wqo which is different from the standard one: we recall here how to prove the equivalence.

Proposition A.8. *The following are equivalent for a preordered set $(P, \leq)$:*

(i) *every upset of P is finitely generated;*

(ii) *for every sequence*

$$p_0, p_1, \ldots, p_i, \ldots$$

of elements from P there are $i < j$ with $p_i \leq p_j$.

Proof. Assume (i) and take U to be the upset $\{q \in P \mid p_i \leq q \text{ for some } i\}$: since U must be finitely generated, (ii) follows immediately.

Conversely, assume (ii) and take an upset U. Pick the set S of the minimal elements from U[38] and build S_0 by taking from S one representative for each equivalence class modulo

[38] An element m from S is minimal iff for all $s \in S$, we have that $s \leq m$ implies $m \leq s$ (notice that we do not assume antisymmetry of $\leq$).

$\leq \cap \geq$ (S_0 is finite, otherwise we can build an infinite sequence of mutually incomparable elements of S, which cannot be). Given $r_0 \in U$, if there is no minimal element m from S_0 such that $m \leq r_0$, then we can build an infinite descending chain $r_0 \geq r_1 \geq \cdots$ where there is no $i < j$ with $r_i \leq r_j$, contradicting (ii). □

Results from Section 6

Before proving results from Section 6, we make a little remark which was implicit in the notation used in Definition 6.3. The remark concerns the shape of index invariant formulae. These formulae are of the kind $\psi(\underline{t}(\underline{j}), a[\underline{t}(\underline{j})])$: according to our conventions from Section 3, $\psi(\underline{t}(\underline{j}), a[\underline{t}(\underline{j})])$ means the result of replacing $\underline{i}$ with $\underline{t}(\underline{j})$ and $\underline{e}$ with $a[\underline{t}(\underline{j})]$ in the quantifier-free $\Sigma_I \cup \Sigma_E$-formula $\psi(\underline{i}, \underline{e})$.[39] When dealing with $\exists^I$-formulae (or also with $\forall^I$-formulae), we do not need such a more liberal format because e.g. $\exists \underline{j}\, \psi(\underline{t}(\underline{j}), a[\underline{t}(\underline{j})])$ can be rewritten as $\exists \underline{j}\, \exists \underline{i}\, (\underline{i} = \underline{t}(\underline{j}) \wedge \psi(\underline{i}, a[\underline{i}]))$. The more liberal format is needed here because the formulae we are using for progress conditions are *quantifier-free*.

The relevant hint for the proof of Proposition 6.4 has already been sketched in Section 6: let us repeat the argument in full detail.

Proposition 6.4. *If there is a progress condition for R, then R is a recurrence property with polynomial complexity.*

Proof. Suppose that $\psi_1(\underline{t}(\underline{j}), a[\underline{t}(\underline{j})]), \ldots, \psi_c(\underline{t}(\underline{j}), a[\underline{t}(\underline{j})])$ are a progress condition for R and let ℓ be the length of the tuple $\underline{j}$. By reductio, let $\mathcal{M}$ be a model such that the cardinality of $\mathtt{INDEX}^{\mathcal{M}}$ is at most n and such that

$$\mathcal{M}, \mathtt{a} \models I(b_1) \wedge \tau(b_1, b_2) \wedge \cdots \wedge \tau(b_m, a_0) \wedge \tau_K(a_0, a_1) \wedge \cdots \wedge \tau_K(a_{k-1}, a_k) \tag{25}$$

holds for a suitable assignment $\mathtt{a}$, where we have that $k > c \cdot n^\ell$ (let $s_0, \ldots, s_k$ be the states assigned by $\mathtt{a}$ to the array variables $a_0, \ldots, a_k$, respectively). Now, for every $r < k$, there are a formula $\psi^r \in \{\psi_1, \ldots, \psi_c\}$ and an ℓ-tuple $\underline{j}^r$ of elements from $\mathtt{INDEX}^{\mathcal{M}}$ such that

$$\mathcal{M} \models \neg\psi^r(\underline{t}(\underline{j}^r), s_r[\underline{t}(\underline{j}^r)]), \qquad \mathcal{M} \models \psi^r(\underline{t}(\underline{j}^r), s_{r+1}[\underline{t}(\underline{j}^r)]) \tag{26}$$

(this is according to (11), because $\mathcal{M} \models \tau_K(s_r, s_{r+1})$ and because $\mathcal{M} \models \neg H(s_r)$ is true since s_r is a reachable state and H is safe). Now, $k > c \cdot n^\ell$, hence the conditions (26) must be true at least twice for the same $\psi^r \in \{\psi_1, \ldots, \psi_c\}$ and for the same tuple $\underline{j}^r$ from $\mathtt{INDEX}^{\mathcal{M}}$: this is in contradiction to (10). □

[39] The notation $\underline{t}(\underline{j})$, in turn, means that at most the index variables $\underline{j}$ occur in the terms $\underline{t}$. These terms $\underline{t}$ must be Σ_I-terms because they denote an element of sort `INDEX` and employ only variables of sort `INDEX`: by construction of our signatures, this means that they cannot employ non Σ_I-symbols.

Lemma A.9. *Entailment tests* (10) *and* (11) *are both effective (for any* $K, H, \psi, \psi_1, \dots, \psi_c$*).*

Proof. Both tests in (10) and (11) are special cases of Theorem 4.1. Let us check this in more detail; in both tests the transition $\tau_K(a_0, a_1)$ introduced in Definition 6.1 occurs: it is easy to see that this $\tau_K(a_0, a_1)$ is logically equivalent to a disjunction of n formulae of the kind $\exists \underline{i}\,(\phi_L(\underline{i}, a_0[\underline{i}], a_1[\underline{i}]) \wedge Up_G(\underline{i}, a_0, a_1))$.[40] For simplicity (the reader can realize that there is no loss of generality in that), we assume that $n = 1$. We also let $\neg H(a_0)$ be the $\forall^I$-formula $\forall \underline{r}\ \chi(\underline{r}, a_0[\underline{r}])$.

For (10) we need to decide

$$A_E^I \models \forall a_0 \forall a_1 \forall \underline{j}\ (\forall \underline{r}\ \chi(\underline{r}, a_0[\underline{r}]) \wedge \tau_K(a_0, a_1) \rightarrow (\psi(\underline{t}, a_0[\underline{t}]) \rightarrow \psi(\underline{t}, a_1[\underline{t}]))) \tag{27}$$

(here the $\underline{t}$ are $\Sigma_I(\underline{j})$-terms), which is the same as deciding A_E^I-validity of

$$\forall \underline{r}\ \chi(\underline{r}, a_0[\underline{r}]) \wedge \exists \underline{i}\ \big(\phi_L(\underline{i}, a_0[\underline{i}], a_1[\underline{i}]) \wedge Up_G(\underline{i}, a_0, a_1)\big) \rightarrow \forall \underline{h}\ \big(\underline{h} = \underline{t} \wedge \psi(\underline{h}, a_0[\underline{h}]) \rightarrow \psi(\underline{h}, a_1[\underline{h}])\big)$$

where the external universal quantifiers $\forall a_0 \forall a_1 \forall \underline{j}$ have been omitted. If we put the latter formula in prenex normal form (using (3)), we immediately realize that its negation falls within the (un)satisfiability procedure of Theorem 4.1.

For (11), we need to check the A_E^I-validity of (we omit the outside quantified variables a_0, a_1)

$$\forall \underline{r}\ \chi(\underline{r}, a_0[\underline{r}]) \wedge \exists \underline{i}\ \big(\phi_L(\underline{i}, a_0[\underline{i}], a_1[\underline{i}]) \wedge Up_G(\underline{i}, a_0, a_1)\big) \rightarrow \exists \underline{j}\ \bigvee_{k=1}^{c} (\neg\psi_k(\underline{t}, a_0[\underline{t}]) \wedge \psi_k(\underline{t}, a_1[\underline{t}])),$$

where the $\underline{t}$ are $\Sigma_I(\underline{j})$-terms. Abstracting out the $\underline{t}$ as $\underline{h}$ and moving quantifiers in front, we get (we use (3) and again we remove the outside universal quantifiers $\forall \underline{i}$)

$$\begin{aligned}\exists \underline{j}\ \exists \underline{h}\ \exists \underline{r}\ \exists l\ \big(\chi(\underline{r}, a_0[\underline{r}]) \wedge \phi_L(\underline{i}, a_0[\underline{i}], a_1[\underline{i}]) \wedge \phi_G(\underline{i}, a_0[\underline{i}], a_1[\underline{i}], a_0[l], a_1[l]) \rightarrow \\ \rightarrow \underline{h} = \underline{t}(\underline{j}) \wedge \bigvee_k (\neg\psi_k(\underline{h}, a_0[\underline{h}]) \wedge \psi_k(\underline{h}, a_1[\underline{h}]))\big).\end{aligned}$$

which is a formula falling into the decision procedure of Theorem 4.1. For future use, we recall that, according to Theorem A.1(iii),[41] this formula is A_E^I-valid iff so is the universal formula (we still omit quantifiers $\forall a_0 \forall a_1 \forall \underline{i}$ in front of it)

$$\begin{aligned}\bigvee_{\sigma} \bigvee_{\sigma'} \bigvee_{\sigma''} \bigvee_{v} \Big(\chi(\underline{r}\sigma'', a_0[\underline{r}\sigma'']) \wedge \phi_L(\underline{i}, a_0[\underline{i}], a_1[\underline{i}]) \wedge \phi_G(\underline{i}, a_0[\underline{i}], a_1[\underline{i}], a_0[v], a_1[v]) \rightarrow \\ \rightarrow \underline{h}\sigma' = \underline{t}(\underline{j}\sigma) \wedge \bigvee_k \big(\neg\psi_k(\underline{h}\sigma', a_0[\underline{h}\sigma']) \wedge \psi_k(\underline{h}\sigma', a_1[\underline{h}\sigma'])\big)\Big).\end{aligned} \tag{28}$$

[40] This is the general shape of T-formulae according to Definition 3.4 (again, we write $Up_G(\underline{i}, a_0, a_1)$ instead of $Update_G(\underline{i}, a, a')$ for shortness).

[41] We use here the dual formulation of Theorem 4.1 (i.e. the formulation in terms of A_E^I-validity instead of the formulation in terms of A_E^I-satisfiability).

where v ranges over the set of representative $\Sigma_I(\underline{i})$-terms and $\sigma, \sigma', \sigma''$ over the substitutions mapping the variables $\underline{j}, \underline{h}, \underline{r}$ into representative $\Sigma_I(\underline{i})$-terms. □

Theorem 6.5. *Suppose that T_E is locally finite and that safety of $\exists^I$-formulae can be effectively checked. Then the existence of a progress condition for a given $\forall^I$-formula R is decidable.*

Proof. If not only T_I but also T_E is locally finite, we show that there is a finite search space for determining whether a progress conditions exists or not. Local finiteness implies that there are only finitely many invariant index formulae involving the $\underline{j}$, once the $\underline{j}$ are fixed (simply because there are only finitely many quantifier free formulae of the kind $\psi(\underline{j}, \underline{e})$ at all – recall that here the $\underline{e}$ are placeholders for the $a[\underline{j}]$). In view of the decidability statement of Lemma A.9, in order to prove the theorem, we must: (I) give a bound to the length of the tuple $\underline{j}$ appearing in the index invariant formulae composing a progress condition for R; (II) limit the search space for the safe formulae H used in the tests (10) and (11). Our proof below is divided into two parts (dealing with problems (I) and (II), respectively).

Part (I). Here we show that there is no need to consider tuples $\underline{j}$ whose cardinality is bigger than the cardinality of the $\underline{i}$ (here the $\underline{i}$ are the variables which are existentially quantified in the formula τ_K). Suppose we got invariant index formulae $\psi_1(\underline{t}, a[\underline{t}]), \ldots, \psi_c(\underline{t}, a[\underline{t}])$ involving a tuple $\underline{t}$ of $\Sigma_I(\underline{j})$-terms where the $\underline{j}$ are index variables of unspecified length: for these formulae, the tests (10) and (11) are both successful. This means that for all $k = 1, \ldots, c$ we have

$$A_E^I \models \forall a_0 \forall a_1 \forall \underline{j}\ (\forall \underline{r}^k\ \chi_k(\underline{r}^k, a_0[\underline{r}^k]) \wedge \tau_K(a_0, a_1) \rightarrow (\psi_k(\underline{t}, a_0[\underline{t}]) \rightarrow \psi_k(\underline{t}, a_1[\underline{t}]))) \tag{29}$$

for some safe $\exists \underline{r}^k\ \neg\chi_k(\underline{r}^k, a_0[\underline{r}^k])$. Moreover, we have A_E^I-validity of (11) (for some safe $\exists \underline{r}\ \neg\chi(\underline{r}, a_0[\underline{r}])$); since we proved that the A_E^I-validity of (11) implies (in fact it is equivalent to) the A_E^I-validity of the universal closure of the formula (28), we can rearrange the latter and get the A_E^I-validity of the universal closure of

$$\bigwedge_{\sigma''} \chi(\underline{r}\sigma'', a_0[\underline{r}\sigma'']) \wedge \phi_L(\underline{i}, a_0[\underline{i}], a_1[\underline{i}]) \wedge \bigwedge_{v} \phi_G(\underline{i}, a_0[\underline{i}], a_1[\underline{i}], a_0[v], a_1[v]) \rightarrow$$
$$\rightarrow \bigvee_{\sigma} \bigvee_{\sigma'} \big(\underline{h}\sigma' = \underline{t}(\underline{j}\sigma) \wedge \bigvee_{k} (\neg\psi_k(\underline{h}\sigma', a_0[\underline{h}\sigma']) \wedge \psi_k(\underline{h}\sigma', a_1[\underline{h}\sigma']))\big)$$

hence also of

$$\forall \underline{r}\ \chi(\underline{r}, a_0[\underline{r}]) \wedge \phi_L(\underline{i}, a_0[\underline{i}], a_1[\underline{i}]) \wedge Up_G(\underline{i}, a_0, a_1) \rightarrow$$
$$\rightarrow \bigvee_{\sigma} \bigvee_{\sigma'} \big(\underline{h}\sigma' = \underline{t}(\underline{j}\sigma) \wedge \bigvee_{k} (\neg\psi_k(\underline{h}\sigma', a_0[\underline{h}\sigma']) \wedge \psi_k(\underline{h}\sigma', a_1[\underline{h}\sigma']))\big)$$

and of[42]

$$\forall \underline{r}\ \chi(\underline{r}, a_0[\underline{r}]) \wedge \phi_L(\underline{i}, a_0[\underline{i}], a_1[\underline{i}]) \wedge Up_G(\underline{i}, a_0, a_1) \rightarrow \\ \rightarrow \bigvee_{\sigma} \bigvee_{k} (\neg\psi_k(\underline{t}(\underline{j}\sigma), a_0[\underline{t}(\underline{j}\sigma)]) \wedge \psi_k(\underline{t}(\underline{j}\sigma), a_1[\underline{t}(\underline{j}\sigma)])).$$

If we introduce the existential quantifiers $\exists \underline{i}$ both in the antecedent and in the consequent, we get

$$A_E^I \models \forall a_0 \forall a_1 \Big(\forall \underline{r}\ \chi(\underline{r}, a_0[\underline{r}]) \wedge \tau_K(a_0, a_1) \rightarrow \\ \rightarrow \bigvee_{\sigma} \bigvee_{k} \exists \underline{i} (\neg\psi_k(\underline{t}(\underline{j}\sigma), a_0[\underline{t}(\underline{j}\sigma)]) \wedge \psi_k(\underline{t}(\underline{j}\sigma), a_1[\underline{t}(\underline{j}\sigma)])) \Big).$$

This, together with (29),[43] shows that we found appropriate invariant index formulae

$$\psi_k(\underline{t}(\underline{j}\sigma), a[\underline{t}(\underline{j}\sigma)])$$

(varying k, σ) involving at most the variables $\underline{i}$.

Part (II). Here our problem is that of making deterministic the search for the safe $\exists^I$-formula H needed for the tests (10) and (11). This is a special case of the following problem: suppose we are given a $\forall^I$-formula $J(a_0)$ such that $\neg J$ is safe and that

$$A_E^I \models \forall a_0\, (J(a_0) \rightarrow \forall a_1\, \forall \underline{i}\, \exists \underline{j}\, \psi(\underline{i}, \underline{j}, a_0[\underline{i}], a_0[\underline{j}], a_1[\underline{i}], a_1[\underline{j}])) \tag{30}$$

holds for a given ψ;[44] we want to compute (*out of* ψ *only*, i.e. independently on J) another $\forall^I$-formula $J^*(a_0)$ such that

$$A_E^I \models \forall a_0\, (J(a_0) \rightarrow J^*(a_0)) \tag{31}$$

and that

$$A_E^I \models \forall a_0\, (J^*(a_0) \rightarrow \forall a_1\, \forall \underline{i}\, \exists \underline{j}\, \psi(\underline{i}, \underline{j}, a_0[\underline{i}], a_0[\underline{j}], a_1[\underline{i}], a_1[\underline{j}])). \tag{32}$$

Once this is achieved, we are done because safety of $\neg J$ implies safety of $\neg J^*$ by (31) and hence J can be replaced by J^*.

Assume that (30) holds; by Lemma A.10 below, we have that

$$A_E^I \models \forall a_0\, \big(J(a_0) \rightarrow \forall a_1\, \forall \underline{i} \bigvee_{\sigma} \psi(\underline{i}, \underline{j}\sigma, a_0[\underline{i}], a_0[\underline{j}\sigma], a_1[\underline{i}], a_1[\underline{j}\sigma])\big) \tag{33}$$

[42] This passage (like the former) is up to logical implication, not up to logical equivalence.

[43] More precisely, we need to apply the exemplification schema to (29) (through the substitutions $\underline{j} \mapsto \underline{j}\sigma$) and to take the universal closure $\forall \underline{i}$ again.

[44] Notice that (10) and (11) are both of the kind (30), modulo standard logical manipulations.

where σ ranges as usual over the substitutions having the representative $\Sigma_I(\underline{i})$-terms in the codomain. Notice now that

$$A_E^I \models \forall a_0 \Big(\forall a_1\, \forall \underline{i} \bigvee_\sigma \psi(\underline{i}, \underline{j}\sigma, a_0[\underline{i}], a_0[\underline{j}\sigma], a_1[\underline{i}], a_1[\underline{j}\sigma]) \rightarrow \\ \rightarrow \forall a_1\, \forall \underline{i}\, \exists \underline{j}\, \psi(\underline{i}, \underline{j}, a_0[\underline{i}], a_0[\underline{j}], a_1[\underline{i}], a_1[\underline{j}])\Big)$$

hence we reach our goal if we show that $\forall a_1\, \forall \underline{i} \bigvee_\sigma \psi(\underline{i}, \underline{j}\sigma, a_0[\underline{i}], a_0[\underline{j}\sigma], a_1[\underline{i}], a_1[\underline{j}\sigma]))$ is A_E^I-equivalent to a universal T-constraint formula $J^*(a_0)$. To this aim, first notice that the formula $\forall a_1\, \forall \underline{i} \bigvee_\sigma \psi(\underline{i}, \underline{j}\sigma, a_0[\underline{i}], a_0[\underline{j}\sigma], a_1[\underline{i}], a_1[\underline{j}\sigma]))$ is A_E^I-equivalent to

$$\forall a_1\, \forall \underline{i}\, \forall \underline{h}\, (\bigvee_\sigma (\underline{h} = \underline{j}\sigma \rightarrow \psi(\underline{i}, \underline{h}, a_0[\underline{i}], a_0[\underline{h}], a_1[\underline{i}], a_1[\underline{h}])))$$

and then apply Lemma A.11 below (take negation on both sides of (34) and let $\underline{k}$ and $\underline{e}$ be $\underline{i}, \underline{h}$ and $a_0[\underline{i}], a_0[\underline{h}]$, respectively). □

Lemma A.10. *Condition* (30) *is equivalent to condition* (33).

Proof. By Lemma A.1, the A_E^I-satisfiability of the negations of both formulae (30) and (33) is equivalent to the A_E^I-satisfiability of the formula

$$\exists a_0\, \exists a_1\, \exists \underline{i} \bigwedge_\sigma \bigwedge_{\sigma'} \big(\phi(\underline{k}\sigma', a_0[\underline{k}\sigma']) \wedge \neg\psi(\underline{i}, \underline{j}\sigma, a_0[\underline{i}], a_0[\underline{j}\sigma], a_1[\underline{i}], a_1[\underline{j}\sigma])\big).$$

where we assumed that $J(a_0)$ is $\forall \underline{k}\, \phi(\underline{k}, a_0[\underline{k}])$. □

Lemma A.11. *Given any* $\phi(\underline{k}, \underline{e}, a_1[\underline{k}])$ *it is possible to compute* $\phi^*(\underline{k}, \underline{e})$ *such that the formula*

$$\exists a_1\, \exists \underline{k}\, \phi(\underline{k}, \underline{e}, a_1[\underline{k}]) \leftrightarrow \exists \underline{k}\, \phi^*(\underline{k}, \underline{e}) \tag{34}$$

is A_E^I*-valid.*

Proof. Observe that $\exists a_1\, \exists \underline{k}\, \phi(\underline{k}, \underline{e}, a_1[\underline{k}])$ is logically equivalent to

$$\exists a_1\, \exists \underline{k}\, \exists \underline{d}\, (a_1[\underline{k}] = \underline{d} \wedge \phi(\underline{k}, \underline{e}, \underline{d}))$$

and to

$$\exists \underline{k}\, \exists \underline{d}\, (\exists a_1\, (a_1[\underline{k}] = \underline{d}) \wedge \phi(\underline{k}, \underline{e}, \underline{d})).$$

The formula $\exists a_1\, (a_1[\underline{k}] = \underline{d})$ is A_E^I-equivalent to the quantifier-free formula $\Pi(\underline{k}, \underline{d})$ saying that if two components of the tuple $\underline{k}$ are equal, so are the corresponding components of the tuple $\underline{d}$. We get the desired ϕ^* by eliminating the elements quantifiers $\underline{d}$ from

$$\exists \underline{k}\, \exists \underline{d}\, (\Pi(\underline{k}, \underline{d}) \wedge \phi(\underline{k}, \underline{e}, \underline{d})).$$

(as usual, see Lemma A.2 for such quantifier elimination). □