Local theory extensions, hierarchical reasoning and applications to verification – Preliminary Report –

Viorica Sofronie-Stokkermans, Carsten Ihlemann, Swen Jacobs

Max-Planck-Institut für Informatik, Campus E 1.4, Saarbrücken, Germany
{sofronie,ihlemann,sjacobs}@mpi-inf.mpg.de

Abstract. We present some new results on hierarchical and modular reasoning in complex theories, as well as several examples of application domains in which efficient reasoning is possible. We show, in particular, that various phenomena analyzed in the verification literature can be explained in a unified way using the notion of local theory extension.

Keywords. automated reasoning, decision procedures, verification

1 Introduction

Many problems occurring in verification can be reduced to proving the satisfiability of conjunctions of literals in a background theory. This can be a concrete theory (e.g. the theory of real or rational numbers), the extension of a theory with additional functions (free, monotone, or recursively defined) or a combination of theories. It is therefore very important to have efficient procedures for checking the satisfiability of conjunctions of ground literals in such theories. In particular, it is very important to identify situations where the search space can be controlled without losing completeness. A class of theories in which this is possible are the so-called *local theories* [1,2,3]. In [4,5] we study the more general notion of *locality of extensions of theories* with new functions axiomatized by sets of clauses. Apart from allowing us to address the problem of restricting the search space, in local theory extensions proof tasks can be reduced, hierarchically, to proof tasks in the base theory. Various results in verification can be explained in a unified way – and new results can be obtained – using properties of local theory extensions. The main results presented in this paper can be summarized as follows:

(1) We show that theories important in verification (e.g. the theory of arrays in [6] and the theory of pointer structures in [7]) satisfy locality conditions.
(2) We present a general framework which allows to identify local theories important in verification. This allows us to handle also fragments which do not satisfy all syntactical restrictions imposed in previous papers. In particular, the axiom sets which we consider may contain alternations of quantifiers.
(3) We use these results to give new examples of local theories of data types.

Dagstuhl Seminar Proceedings 07401
Deduction and Decision Procedures
http://drops.dagstuhl.de/opus/volltexte/2007/1250

(4) We discuss the experiments we made with an implementation.

Structure of the paper. The paper is structured as follows. In Sect. 2 local theories and local theory extensions are defined. Results on hierarchical reasoning, parameterized decidability and complexity results, and possibilities of recognizing local extensions are summarized. Section 3 contains a large number of examples, ranging from extensions with monotonicity, injectivity and (guarded) boundedness properties to theories of data structures (pointers, arrays). A general framework for recognizing locality in verification – including a description of the verification problems we address – is presented in Sect. 4. Several examples of applications are presented in Sect. 5. We describe our implementation and some experiments in Sect. 6, and present the conclusions and plans for future work in Sect. 7.

2 Locality

Theories and models. Theories can be regarded as collections of formulae (i.e. can be described as the consequences of a set of axioms), as collections of models (the reals/integers; the set of all models of a set of axioms), or both. If $\mathcal{T}$ is a theory and ϕ, ψ are formulae, we say that $\mathcal{T} \wedge \phi \models \psi$ (written also $\phi \models_{\mathcal{T}} \psi$) if ψ is true in all models of $\mathcal{T}$ which satisfy ϕ. If $\mathcal{T} \wedge \phi \models \bot$ (where $\bot$ is false), there are no models of $\mathcal{T}$ which satisfy ϕ, i.e. ϕ is unsatisfiable w.r.t. $\mathcal{T}$. In what follows we will refer both to total and to partial models of a theory (resp. of a set of clauses). For the necessary definitions (weak validity, Evans validity, weak partial model, (Evans) partial model) we refer to [5].

Local theories. The notion of *local theory* was introduced and studied by Givan and McAllester in [1,2]. A *local theory* is a set of Horn clauses $\mathcal{K}$ such that, for any ground Horn clause C, $\mathcal{K} \models C$ only if $\mathcal{K}[C] \models C$, where $\mathcal{K}[C]$ denotes those instances of $\mathcal{K}$ containing only subterms of ground terms in $\mathcal{K}$ or C. Therefore, for local theories validity of ground Horn clauses can be checked in PTIME.

Local theory extensions. Let $\mathcal{T}_0$ be a theory with signature $\Pi_0 = (S_0, \Sigma_0, \mathsf{Pred})$, where S_0 is a set of sorts, Σ_0 a set of function symbols, and Pred a set of predicate symbols. We consider extensions $\mathcal{T}_1$ of $\mathcal{T}_0$ with new sorts and function symbols (i.e. with signature $\Pi = (S, \Sigma, \mathsf{Pred})$, where $S = S_0 \cup S_1$ and $\Sigma = \Sigma_0 \cup \Sigma_1$), satisfying a set $\mathcal{K}$ of (implicitly universally quantified) clauses. An extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is *local* if satisfiability of a set G of clauses with respect to $\mathcal{T}_0 \cup \mathcal{K}$ only depends on $\mathcal{T}_0$ and those instances $\mathcal{K}[G]$ of $\mathcal{K}$ in which the terms starting with extension functions are in the set $\mathsf{st}(\mathcal{K}, G)$ of ground terms which already occur in G or $\mathcal{K}$. A weaker notion of locality (*stable locality*) exists; it allows to restrict the search to the instances $\mathcal{K}^{[G]}$ of $\mathcal{K}$ in which the variables below extension functions are instantiated with terms in the set $\mathsf{st}(\mathcal{K}, G)$ of ground terms which already occur in G or $\mathcal{K}$.

The extension axioms may, in addition, contain subformulae with alternations of quantifiers. To address such situations, we introduced the notions of *extended*

locality [5]. We consider extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ with a set $\mathcal{K}$ of (implicitly universally quantified) axioms of the form $(\Phi(x_1, \ldots, x_n) \vee C(x_1, \ldots, x_n))$, where $\Phi(x_1, \ldots, x_n)$ is an *arbitrary first-order formula* in the base signature Π_0 with free variables $x_1, \ldots, x_n$, and $C(x_1, \ldots, x_n)$ is a *clause* in the signature Π. We extend the notion of local theory extension accordingly. If $\mathcal{K}$ is a formula with free variables $x_1, \ldots, x_n$, $\mathcal{K}[\Gamma]$ consists of all instances of $\mathcal{K}$ in which the terms starting with extension functions are in the set $\mathsf{st}(\mathcal{K}, G)$ of ground terms which already occur in G or $\mathcal{K}$, and $\mathcal{K}^{[\Gamma]}$ consists of all instances of $\mathcal{K}$ in which the variables below a Σ_1-symbol are instantiated with terms in $T_{\Sigma_0}(\mathsf{st}(\mathcal{K}, G))$.

(ELoc) For every formula $\Gamma = \Gamma_0 \cup G$, where Γ_0 is a Π_0-sentence and G is a set of ground clauses, $\mathcal{T}_1 \cup \Gamma \models \bot$ iff $\mathcal{T}_0 \cup \mathcal{K}[\Gamma] \cup \Gamma$ has no weak partial model in which all terms in $\mathsf{st}(\mathcal{K}, G)$ are defined.

(ESLoc) For every formula $\Gamma = \Gamma_0 \cup G$, where Γ_0 is a Π_0-sentence and G is a set of ground clauses, $\mathcal{T}_1 \cup \Gamma \models \bot$ iff $\mathcal{T}_0 \cup \mathcal{K}^{[\Gamma]} \cup \Gamma$ has no partial model in which all terms in $\mathsf{st}(\mathcal{K}, G)$ are defined.

We also identify a class of locality conditions between locality and stable locality. Let Ψ be a function associating with a set $\mathcal{K}$ of axioms and a set of ground terms T_0 a set $\Psi(\mathcal{K}, T_0)$ of ground terms with (i) All ground subterms in $\mathcal{K}$ and T_0 are also in $\Psi(\mathcal{K}, T_0)$; (ii) Ψ is monotone, i.e. for all sets of ground terms T_0 and T_0' if $T_0 \subseteq T_0'$ then $\Psi(\mathcal{K}, T_0) \subseteq \Psi(\mathcal{K}, T_0')$; (iii) Ψ defines a closure operation, i.e. for all sets of ground terms T_0, $\Psi(\mathcal{K}, \Psi(\mathcal{K}, T_0)) \subseteq \Psi(\mathcal{K}, T_0)$. We can assume, without loss of generality, that $\Psi(\mathcal{K}, T_0)$ is closed under subterms. Let $\mathcal{K}[\Psi(\mathcal{K}, G)]$ be the set of instances of $\mathcal{K}$ in which the extension terms are in $\Psi(\mathcal{K}, \mathsf{subterms}(G))$, which here will be denoted by $\Psi(\mathcal{K}, G)$. We say that an extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ satisfies an extended locality condition w.r.t. Ψ if it satisfies confition (ELoc^{Ψ}):

(ELoc^{Ψ}) for every formula $\Gamma{=}\Gamma_0{\cup}G$, where Γ_0 is a Π_0-sentence and G a set of ground clauses, $\mathcal{T}_1 \cup \Gamma \models \bot$ iff $\mathcal{T}_0 \cup \mathcal{K}[\Psi(\mathcal{K}, G)] \cup \Gamma$ has no weak partial model in which all terms in $\Psi(\mathcal{K}, G)$ are defined.

If we consider only axiomatizations with clauses (i.e. the formulae in $\mathcal{K}$ are just clauses and only satisfiability of sets G of ground clauses is considered) then we obtain a notion of locality (Loc^{Ψ}) extending the usual notion (Loc) of locality of an extension.

2.1 Hierarchical reasoning in local theory extensions

Let $\mathcal{T}_0 \subseteq \mathcal{T}_1{=}\mathcal{T}_0 \cup \mathcal{K}$ be a theory extension satisfying condition $\mathsf{ELoc}, \mathsf{ESLoc}$ or (ELoc^{Ψ}). To check the satisfiability w.r.t. $\mathcal{T}_1$ of a formula $\Gamma = \Gamma_0 \cup G$, where Γ_0 is a Π_0-sentence and G of ground clauses over Π_1 we proceed as follows:

Step 1: Use locality. By the locality assumption, $\mathcal{T}_1 \cup \Gamma_0 \cup G$ is satisfiable iff $\mathcal{T}_0{\cup}\mathcal{K}{*}[G]{\cup}\Gamma_0{\cup}G$ has a (weak) partial model with corresponding properties, where, depending on the type of locality, $\mathcal{K} * [G]$ is $\mathcal{K}[G], \mathcal{K}^{[G]}$ or $\mathcal{K}[\Psi(\mathcal{K}, G)]$.

Step 2: Purification. We purify $\mathcal{K}*[G]\cup G$ by introducing, in a bottom-up manner, new constants c_t for subterms $t = f(g_1, \ldots, g_n)$ with $f \in \Sigma_1$, g_i ground $\Sigma_0 \cup \Sigma_c$-terms (Σ_c is a set of constants containing the constants introduced by purification), together with corresponding definitions $c_t \approx t$. The result of the transformation is $\mathcal{K}_0 \cup G_0 \cup \Gamma_0 \cup D$, where $\mathcal{K}_0, G_0, \Gamma_0$ are formulae without function symbols in Σ_1 and D consists of definitions $f(g_1, \ldots, g_n) \approx c$, where $f \in \Sigma_1$, c is a constant, $g_1, \ldots, g_n$ are ground Σ_0-terms.

Step 3: Reduction to testing satisfiability in $\mathcal{T}_0$. We reduce the problem to testing satisfiability in $\mathcal{T}_0$ by replacing D with the following set of clauses:

$$N_0 = \{\bigwedge_{i=1}^{n} c_i \approx d_i \rightarrow c = d \mid f(c_1, \ldots, c_n) \approx c, f(d_1, \ldots, d_n) \approx d \in D\}.$$

Theorem 1 *Let $\mathcal{K}$ and $\Gamma = \Gamma_0 \wedge G$ be as specified above. Assume that $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ satisfy condition* ELoc *or* ESLoc*. Let $\mathcal{K}_0 \cup G_0 \cup \Gamma_0 \cup D$ be obtained from $\mathcal{K} * [G] \cup \Gamma_0 \cup G$ by purification, as explained above. The following are equivalent:*

(1) $\mathcal{T}_0 \cup \mathcal{K}*[G] \cup \Gamma_0 \cup G$ has a partial model with all terms in $\mathsf{st}(\mathcal{K}, G)$ defined.
(2) $\mathcal{T}_0 \cup \mathcal{K}_0 \cup G_0 \cup \Gamma_0 \cup D$ has a partial model with all terms in $\mathsf{st}(D)$ defined.
(3) $\mathcal{T}_0 \cup \mathcal{K}_0 \cup G_0 \cup \Gamma_0 \cup N_0$ has a (total) model.

Locality allows us to obtain decidability and parameterized complexity results:

Theorem 2 *Assume that the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ either (1) satisfies condition* (ELoc)*, or (2) satisfies condition* (ESLoc) *and $\mathcal{T}_0$ is locally finite. If the fragment of $\mathcal{T}_0$ in which $\mathcal{K}_0 \cup G_0 \cup \Gamma_0 \cup N_0$ is contained is decidable, then satisfiability of ground goals of the form $\Gamma_0 \cup G$ w.r.t. $\mathcal{T}_1$ is decidable.*

If $\mathcal{K}$ consists only of clauses and all variables occur below an extension function and if Γ is a set of ground clauses then $\mathcal{K} * [G] \wedge \Gamma$ consists of ground clauses so we can reduce reasoning in $\mathcal{T}_1$ to reasoning in an extension of $\mathcal{T}_0$ with free function symbols; an SMT procedure can be used. If Γ_0 contains quantifiers or $\mathcal{K} * [G]$ contains free variables it is not possible to use SMT provers.

2.2 Recognizing generalized locality

Theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1$ satisfying condition (ELoc) and (ESLoc) can be recognized by showing that certain partial models of $\mathcal{T}_1$ can be "completed" to total models of $\mathcal{T}_1$ with elementarily equivalent (or isomorphic) Π_0-reducts. These conditions can be expressed by completability axioms:

(Comp) Every partial model A of $\mathcal{T}_1$ with totally defined Σ_0-functions and extension functions with a finite domain of definition weakly embeds into a total model B of $\mathcal{T}_1$ s.t. $A_{|\Pi_0}$ and $B_{|\Pi_0}$ are isomorphic.

($\mathsf{Comp_w}$) Every weak partial model A of $\mathcal{T}_1$ with totally defined Σ_0-functions and extension functions with a finite domain of definition weakly embeds into a total model B of $\mathcal{T}_1$ s.t. $A_{|\Pi_0}$ and $B_{|\Pi_0}$ are isomorphic.

Theorem 3 ([5]) *(1) If all terms of $\mathcal{K}$ starting with a Σ_1-function are flat and linear and the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ satisfies* ($\mathsf{Comp_w}$) *then it satisfies* (ELoc).
(2) If $\mathcal{T}_0$ is a universal theory and $\mathcal{T}_0 \subseteq \mathcal{T}_1$ satisfies (Comp) *then it satisfies* (ESLoc).

Let Ψ be a function associating with a set $\mathcal{K}$ of axioms and a set of ground terms T_0 a set $\Psi(\mathcal{K}, T_0)$ of ground terms with (i) All ground subterms in $\mathcal{K}$ and T_0 are also in $\Psi(\mathcal{K}, T_0)$; (ii) Ψ is monotone, i.e. for all sets of ground terms T_0 and T_0' if $T_0 \subseteq T_0'$ then $\Psi(\mathcal{K}, T_0) \subseteq \Psi(\mathcal{K}, T_0')$; (iii) Ψ defines a closure operation, i.e. for all sets of ground terms T_0, $\Psi(\mathcal{K}, \Psi(\mathcal{K}, T_0)) \subseteq \Psi(\mathcal{K}, T_0)$.

Theorem 4 *Assume that $\mathcal{K}$ is a family of Σ_1-flat clauses in the signature Π.*

(1) Assume that $\mathcal{T}_0$ is a first-order theory. If every weak partial model A of $\mathcal{T}_1$ such that $\{f(a_1, \ldots, a_n) \mid a_i \in A, f \in \Sigma_1, f_A(a_1, \ldots, a_n)$ defined$\}$ is closed under $\Psi(\mathcal{K}, \_\_)$ weakly embeds into a total model of $\mathcal{T}_1$ then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1 := \mathcal{T}_0 \cup \mathcal{K}$ satisfies (Loc^{Ψ}).
(2) To guarantee (ELoc^{Ψ}) *we need to additionally require that each partial model as in (1) weakly embeds into a total model B such that the reducts to Π_0 of A and B are elementarily equivalent.*

A combination of extensions of a theory $\mathcal{T}_0$ satisfying condition Comp ($\mathsf{Comp_w}$) also satisfies condition Comp ($\mathsf{Comp_w}$) and hence also condition ESLoc (ELoc).

Theorem 5 ([8]) *Let $\mathcal{T}_0$ be a first order theory with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$ and (for $i \in \{1, 2\}$) $\mathcal{T}_i = \mathcal{T}_0 \cup \mathcal{K}_i$ be an extension of $\mathcal{T}_0$ with signature $\Pi_i = (\Sigma_0 \cup \Sigma_i, \mathsf{Pred})$. Assume that both extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1$ and $\mathcal{T}_0 \subseteq \mathcal{T}_2$ satisfy condition* ($\mathsf{Comp_w}$), *and that $\Sigma_1 \cap \Sigma_2 = \emptyset$. Then the extension $\mathcal{T}_0 \subseteq \mathcal{T} = \mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ satisfies condition* ($\mathsf{Comp_w}$). *If, additionally, all terms in $\mathcal{K}_i$ which start with a function symbol in Σ_i are flat and linear, for $i = 1, 2$, then the extension is local.*

3 Examples of local theory extensions

Theorem 3 allows us to identify many examples of extensions satisfying $\mathsf{E(S)Loc}$.

Free functions and monotone functions. Any extension $\mathcal{T}_0 \cup \mathsf{Free}(\Sigma)$ of a theory $\mathcal{T}_0$ with a set Σ of *free function symbols* satisfies condition ($\mathsf{Comp_w}$). Consider the following monotonicity conditions:

$$(\mathsf{Mon}_f^{\sigma}) \qquad \bigwedge_{i \in I} x_i \leq_i^{\sigma_i} y_i \wedge \bigwedge_{i \notin I} x_i = y_i \rightarrow f(x_1, .., x_n) \leq f(y_1, .., y_n),$$

where for $i \in I$, $\sigma_i \in \{-, +\}$, and for $i \notin I$, $\sigma(i)=0$, and $\leq^+ = \leq$ and $\leq^- = \geq$. The extensions of any (possibly many-sorted) theory whose models are posets with functions satisfying the axioms Mon_f^{σ} satisfy condition ($\mathsf{Comp_w}$) if the codomains of the functions have a bounded semilattice reduct or are totally ordered [5,9,10].

Strict monotonicity. Consider the strict monotonicity condition:

$$(\mathsf{SMon}(f)) \qquad \forall i, j (i < j \rightarrow f(i) < f(j)).$$

Let $\mathcal{T}_0$ be a theory (containing a sort i) and $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathsf{SMon}(f)$ be the extension of $\mathcal{T}_0$ with a function f of sort $\mathsf{i} \to \mathsf{e}$. Assume that in all models of $\mathcal{T}_0$ the support of sort i has an underlying strict total order $<$, and in all models of $\mathcal{T}_1$ the support of sort e has an underlying strict total order $<$. If in all models of $\mathcal{T}_1$ there exist injective order-preserving maps from any interval of the support of sort i to any interval of the support e then the extension is local.

Example 1. Let $\mathcal{T}_0$ be the combination of $\mathcal{T}_0^{\mathsf{i}}$ (the theory of linear integer arithmetic, sort i) and $\mathcal{T}_0^{\mathsf{num}}$ (the theory of real numbers, sort num). The extension $\mathcal{T}_1$ of $\mathcal{T}_0$ with a function f of arity $\mathsf{i} \to \mathsf{num}$ satisfying $\mathsf{SMon}(f)$ is local.

Note that it is not possible to prove the locality of the extension of the theory of integers with a function satisfying $\mathsf{SMon}(f)$. We need to choose another axiomatization instead:

$$(\mathsf{SMon}_{\mathbb{Z}}(f)) \qquad \forall i, j, k (i < j \wedge j - i = k \to f(i) < f(j) - k).$$

It can be proved that the extension $\mathcal{T}_1$ of the theory of integers with a function f satisfying $(\mathsf{SMon}_{\mathbb{Z}}(f))$ satisfies condition $(\mathsf{Comp}_{\mathsf{w}})$ and hence is local.

A related axiom is the following (c below is a constant):

$$(\mathsf{SdMon}(f)) \qquad \forall i, j, k (i < j \wedge j - i = k \to f(i) < f(j) - k * c).$$

Example 2. Let $\mathcal{T}_0$ be the combination of $\mathcal{T}_0^{\mathsf{i}}$ (the theory of linear integer arithmetic, sort i) and $\mathcal{T}_0^{\mathsf{num}}$ (the theory of real numbers, sort num). The extension $\mathcal{T}_1$ of $\mathcal{T}_0$ with a function f of arity $\mathsf{i} \to \mathsf{num}$ satisfying $\mathsf{SdMon}(f)$ (where c can be a constant of sort i or of sort num) is local.

Injectivity. Let $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathsf{Inj}(f)$, be the extension of the theory $\mathcal{T}_0$ with a unary function f of sort $\mathsf{i} \to \mathsf{e}$ subject to the injectivity condition:

$$(\mathsf{Inj}(f)) \qquad \forall i, j (i \neq j \to f(i) \neq f(j)).$$

If in all models of $\mathcal{T}_1$ the support of sort i has cardinality lower or equal to the cardinality of the support of sort e then the extension is local.

Extensions with definitions. We considered extensions of a Π_0-theory $\mathcal{T}_0$, where $\Pi_0 = (\Sigma_0, \mathsf{Pred})$, with operators in a set Σ which are defined in terms of the operations in Σ_0 in [9,10]. Let $\mathcal{T}_0$ be a Π_0-theory and Σ_1 be a set of operation symbols. Let $\mathcal{K} = \{\mathsf{Def}_f \mid f \in \Sigma_1\}$, where for every $f \in \Sigma_1$, Def_f is a conjunction of formulae (which can be seen as a definition of f) of the form:

$$\bigwedge_{i=1}^{k} \forall \overline{x} (\phi_i(x_1, \ldots, x_n) \to f(x_1, \ldots, x_n) = t_i(x_1, \ldots, x_n))$$

where t_i are Σ_0-terms, and ϕ_i are Π_0-clauses such that for $i \neq j$, $\phi_i \wedge \phi_j$ is unsatisfiable w.r.t. $\mathcal{T}_0$. Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ is local.

Boundedness conditions. We now consider extensions with functions satisfying boundedness conditions and possibly also monotonicity [9,10]. Let $\mathcal{T}_0$ be a Π_0-theory with a reflexive binary predicate symbol $\leq$, and Σ_1 be a set of operation symbols. The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \{\mathsf{GBound}_f \mid f \in \Sigma_1\}$ is local, where (GBound_f) specifies piecewise boundedness of f:

$$(\mathsf{GBound}_f) \quad \bigwedge_{i=1}^{k} \forall \overline{x}(\phi_i(\overline{x}) \rightarrow t_i(\overline{x}) \leq f(\overline{x}) \leq t_i'(\overline{x}))$$

where t_i, t_i' are Σ_0-terms and ϕ_i are Π_0-clauses such that if $i \neq j$ then $\phi_i \wedge \phi_j$ is unsatisfiable w.r.t. $\mathcal{T}_0$.

Let $\mathcal{T}_0$ be a Σ_0-theory of bounded $\vee$-semilattice-ordered (possibly many-sorted) structures, and let f be a new function symbol. Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{Mon}_f^\sigma \cup \mathsf{Bound}_f^t$ is finitely local, where (Bound_f^t) is the boundedness condition:

$$(\mathsf{Bound}_f^t) \quad \forall x_1, \ldots, x_n (f(x_1, \ldots, x_n) \leq t(x_1, \ldots, x_n)),$$

where $t(x_1, \ldots, x_n)$ is a term in the base signature Π_0 with the same monotonicity as f, i.e. satisfying

$$\forall x (\bigwedge_{i=1}^{n} x_i \leq^{\sigma_i} y_i \rightarrow t(x_1, \ldots, x_n) \leq t(y_1, \ldots, y_n)).$$

Pointer data structures à la McPeak and Necula. In [7], McPeak and Necula investigate reasoning in pointer data structures. The language used has two sorts (a pointer sort p and a scalar sort s). Sets Σ_p and Σ_s of pointer resp. scalar fields are given. They can be modeled by functions of sort $\mathsf{p} \rightarrow \mathsf{p}$ and $\mathsf{p} \rightarrow \mathsf{s}$, respectively. A constant null of sort p exists. The only predicate of sort p is equality between pointers; predicates of scalar sort can have any arity. In this language one can define pointer (dis)equalities and arbitrary scalar constraints. The local axioms considered in [7] are of the form

$$\forall p \;\; \mathcal{E} \vee \mathcal{C} \tag{1}$$

where $\mathcal{E}$ contains disjunctions of pointer equalities and $\mathcal{C}$ contains scalar constraints (sets of both positive and negative literals). It is assumed that for all terms $f_1(f_2(\ldots f_n(p)))$ occurring in the body of an axiom, the axiom also contains the disjunction $p = \mathsf{null} \vee f_n(p) = \mathsf{null} \vee \cdots \vee f_2(\ldots f_n(p)) = \mathsf{null}$. This has the rôle of excluding null pointer errors.

Theorem 6 ([8]) *The two-sorted extension $\mathcal{T}_0 \cup \mathcal{K}$ of a Π_0-theory $\mathcal{T}_0$ (sort s), with signature $\Pi = (S, \Sigma, \mathsf{Pred})$ – where $S = \{\mathsf{p}, \mathsf{s}\}$, $\Sigma = \Sigma_p \cup \Sigma_s \cup \Sigma_0$ – axiomatized by a set $\mathcal{K}$ of axioms $\forall p(\mathcal{E} \vee \mathcal{C})$ of type (1) is a stably local extension of $\mathcal{T}_0$.*

The theory of arrays à la Bradley, Manna and Sipma. In [6] a decidable fragment of the theory of arrays that allows some quantification is studied, namely the *array property fragment*. The principal characteristics of this

fragment are the following: The index theory $\mathcal{T}_i$ is Presburger arithmetic. The element theory (in the case of many-dimensional arrays: the element theories) are parametric. The theory $\mathcal{T}_A$ of arrays is the extension of the combination $\mathcal{T}_0$ of the index and element theories with functions read, write and axioms:

$$\text{read}(\text{write}(a,i,e),i) = e \qquad j \neq i \rightarrow \text{read}(\text{write}(a,i,e),j) = \text{read}(a,j).$$

For simplicity, the considerations below are for arrays of dimension 1, the general case is similar. The array property fragment is defined as follows:

An *index guard* is a positive Boolean combination of atoms of the form $t \leq u$ or $t = u$ where t and u are either a variable of index sort or a ground term (of index sort) constructed from (Skolem) constants and integer numbers using addition and multiplication with integers. A formula of the form $(\forall i)(\varphi_I(i) \rightarrow \varphi_V(i))$ is an *array property* if φ_I is an index guard and if any universally quantified variable of index sort i only occurs in a direct array read $\mathsf{read}(a,x)$ in φ_V. Array reads may not be nested. The *array property fragment* consists of all existentially-closed Boolean combinations of array property formulae and quantifier-free formulae.

The decision procedure proposed in [6] decides satisfiability of formulae in negation normal form in the array property fragment in the following steps.

1. Replace all existentially quantified array variables with Skolem constants; replace $\mathsf{read}(a,i)$ with $a(i)$; eliminate all terms of the form $\mathsf{write}(a,i,e)$ by replacing them with fresh array names b, and replacing any formula $\phi(\mathsf{write}(a,i,e))$ with $\phi(b) \wedge (b(i){=}e) \wedge \forall j(j{\leq}i{-}1 \vee i{+}1{\leq}j{\rightarrow}b(j){=}a(j))$.[1]
2. Existentially quantified index variables are replaced with Skolem constants.
3. Universal quantification over index variables is replaced by conjunction of suitably chosen instances of the variables.

For determining the set of ground instances to be used in Step 5, the authors prove that certain partial "minimal" models can be completed to total ones.

Theorem 7 (cf. also [6]) *Let $\mathcal{I}$ be the set of index terms defined in [6]. Let $\mathcal{K}$ be the clause part and G the ground part (after the transformation steps (1)–(3)). Let $\Psi(\mathcal{K},G) = \{f(i_1,\ldots,i_n) \mid f$ array name $, i_1,\ldots,i_n \in \mathcal{I}\}$. Every partial model of $\mathcal{T}_0 \cup \mathcal{K}[\Psi(\mathcal{K},G)] \cup G$ in which all terms in $\Psi(\mathcal{K},G)$ are defined can be transformed into a (total) model of $\mathcal{T}_0 \cup \mathcal{K} \cup G$. This criterion entails ELoc^{Ψ}.*

4 A general framework for obtaining locality results

In Section 3 we identified a large number of theory extensions which can be proved to be local. Most of the updates in states of reactive and hybrid systems can be described by boundedness or definedness axioms guarded by formulae which describe a partition of the state space, and therefore can be expressed with axioms which define local theory extensions. In general however, programs handle complex data structures; in addition to reasoning about updates it is

[1] By the definition of array property formulae, if a term $\mathsf{write}(a,i,e)$ occurs in the array property fragment then i is an existentially quantified index variable.

usually necessary to also reason about various data types such as lists, arrays, records, etc. Theorem 5 allows us to identify the cases and the types of locality which are preserved when combining theories. The results in Sect. 2.2 and 3 provide a general framework for obtaining locality results.

4.1 Verification Problems

The problems we consider are related to the verification of parametric systems (parametric either w.r.t. the number of subsystems involved, or w.r.t. some data used to describe the states and their updates).

Models: Transition constraint systems. We model systems using transition constraint systems $T = (V, \Sigma, \mathsf{Init}, \mathsf{Update})$ which specify:

- the variables (V) and function symbols (Σ) whose values change in time,
- properties of initial states (formula Init), and
- a transition relation Update which specifies the relationship between the old values of variables x and function symbols f before the transition and the new values (x', f') after the transition. The transition relation is expressed by a formula containing variables in $V \cup V'$ and function symbols in $\Sigma \cup \Sigma'$, where V' is a copy of V and Σ' is a copy of Σ (denoting the variables resp. functions after the transition).

Such descriptions can be obtained from specifications (in [11] we used CSP-OZ-DC specifications – referring to processes, data and time). With every specification, a "background theory" $\mathcal{T}_S$ is associated, which describes the data types occurring in the specification and their properties ($\mathcal{T}_S$ usually is a combination of arithmetic and function symbols subject to axioms). The verification problems we consider are invariant checking and bounded model checking.

Invariant checking. We can check whether a formula Ψ is an inductive invariant of a transition constraint system $T{=}(V, \Sigma, \mathsf{Init}, \mathsf{Update})$ in two steps: (1) prove that $\mathcal{T}_S, \mathsf{Init} \models \Psi$; (2) prove that $\mathcal{T}_S, \Psi, \mathsf{Update} \models \Psi'$, where Ψ' results from Ψ by replacing each $x \in \mathcal{V}$ by x' and each f in Σ by f'. Failure to prove (2) means that Ψ is not an invariant, or Ψ is not inductive w.r.t. T.[2]

Bounded model checking. We check whether, for a fixed k, unsafe states are reachable by runs of length at most k. Formally, we check whether:

$$\mathcal{T}_S \wedge \mathsf{Init}_0 \wedge \bigwedge_{i=1}^{j} \mathsf{Update}_i \wedge \neg\Psi_j \models \bot \qquad \text{for all } 0 \leq j \leq k,$$

where Update_i is obtained from Update by replacing all variables $x \in \mathcal{V}$ by x_i and any $f \in \Sigma$ by f_i, and all $x' \in \mathcal{V}'$, $f' \in \Sigma'$ by x_{i+1}, f_{i+1}; Init_0 is Init with x_0 replacing $x \in \mathcal{V}$ and f_0 replacing $f \in \Sigma$; Ψ_i is obtained from Ψ similarly.

We can use deductive techniques to check whether safety (expressed by a suitable clause) is an invariant, or holds for paths of bounded length, *for given*

[2] Proving that Ψ is an invariant of the system in general requires to find a stronger formula Γ (i.e., $\mathcal{T}_S \models \Gamma \rightarrow \Psi$) and prove that Γ is an inductive invariant.

instances of the parameters, or *under given constraints on parameters*. We aim at identifying situations in which decision procedures exist. We will show that this is often the case, by investigating locality phenomena in verification. As a by-product, this will allow us to consider problems more general than usual tasks in verification, namely to *derive constraints between parameters* which guarantee safety. These constraints can be used to solve optimization problems (maximize/minimize some of the parameters) such that safety is guaranteed. [3]

4.2 A general framework for locality results

In invariant checking and bounded model checking, the paths to be verified can be used to identify the chains of extensions to be considered in the deduction process. These extensions are often (combinations) of various extensions with guarded boundedness conditions. Thus results in Sect. 3 and on preservation of locality under combinations of extensions of a base theory with disjoint extension functions (Sect. 2.2) allow us to extend the classes of theories occurring in verification for which instantiation-based complete decision procedures exist.

Extending the fragment of Necula and McPeak. The remarks above can be used for pointer structures which change during the execution of a program.

Theorem 8 *Assume that the update axioms* $\mathsf{Update}(\Sigma, \Sigma')$ *describe how the values of the* Σ*-functions change, depending on a set* $\{\phi_i \mid i \in I\}$ *of mutually exclusive conditions, expressed as formulae over the base signature and the* Σ*-functions (axioms of type (2) below represent precise ways of defining the updated functions, whereas axioms of type (3) represent boundedness properties on the updated scalar fields, assuming the scalar domains are partially ordered):*

$$\forall x(\phi_i(x) \rightarrow f_i'(x) = s_i) \quad i \in \{1, \ldots, m\}, \text{ where } \phi_i \wedge \phi_j \models_{\mathcal{T}_0} \bot \text{ if } i \neq j \quad (2)$$

$$\forall x(\phi_i(x) \rightarrow t_i \leq f_i'(x) \leq s_i) \quad i \in \{1, \ldots, m\}, \text{ where } \phi_i \wedge \phi_j \models_{\mathcal{T}_0} \bot \text{ if } i \neq j \quad (3)$$

where s_i, t_i *are terms over the signature* Σ*. They define a local theory extension.*

Deciding whether the formula ϕ holds for new data structure can now be considered a satisfiability problem over the extended theory. If ϕ is universally quantified, the problem can be efficiently decided using the method in [7].

Extending the array property fragment. There are several ways of extending the array property fragment:

Theorem 9 *(1) Extensions with new arrays satisfying guarded boundedness axioms, or defined using guarded definitions as in Sect. 3 are local.* [4]

[3] All the examples in this paper will address invariant checking only. Bounded model checking problems can be handled similarly.

[4] An example are definitions of new arrays by writing x at a (constant) index c, axiomatized by $\{\forall i(i = c \rightarrow a'(i) = a(i)), \quad \forall i(i \neq c \rightarrow a'(i) = x)\}$.

(2) Extensions with new arrays satisfying injectivity or (strict) monotonicity (and possibly boundedness axioms), under the assumptions about the element theory specified in the corresponding paragraphs of Sect. 3 are local.
(3) Any combination of extensions as those mentioned in (1), (2) (with disjoint sets of newly introduced array constants) leads to a proper extension of the fragment in [6] satisfying condition ELoc.

5 Examples of application domains

We illustrate the ideas above on several types of examples in verification. In Section 5.1 we describe an application of hierarchical reasoning to the verification of the safety of a train control system (cf. also [12]). In Section 5.2 we illustrate the applicability of our methods to the verification of algorithms for updating pointer structures. In Section 5.3 an application to the verification of a scheduling protocol is presented. (Note that the verification tasks in the examples mentioned above are outside the scope of known methods such as [7] and [6].)

5.1 A RBC Case Study

The case study we discuss here is taken from the specification of the European Train Control System (ETCS) standard [13] and presented in detail in [12]. We consider a radio block center (RBC), which communicates with all trains on a given track segment. Trains may enter and leave the area, given that a certain maximum number of trains on the track is not exceeded. Every train reports its position to the RBC in given time intervals and the RBC communicates to every train how far it can safely move, based on the position of the preceding train. It is then the responsibility of the trains to adjust their speed between given minimum and maximum speeds.

For a first try at verifying properties of this system, we have considerably simplified it: we abstract from the communication issues in that we always evaluate the system after a certain time $\Delta\mathsf{t}$, and at these evaluation points the positions of all trains are known. Depending on these positions, the possible speed of every train until the next evaluation is decided: if the distance to the preceding train is less than a certain limit $\mathsf{l}_{\mathsf{alarm}}$, the train may only move with minimum speed min (otherwise with any speed between min and the maximum speed max).

We present two formal system models. In the first one we have a fixed number of trains; in the second we allow for entering and leaving trains.

Model 1: Fixed Number of Trains. In this simpler model, any state of the system is characterized by the real-valued constants $\Delta\mathsf{t} > 0$ (the time between evaluations of the system), min and max (the minimum and maximum speed of trains), $\mathsf{l}_{\mathsf{alarm}}$ (the distance between trains which is deemed secure), the integer constant n (the number of trains) and the function pos (mapping integers between 0 and $n-1$ to real values representing the position of the corresponding train).

We use an additional function pos' to model the evolution of the system: $\mathsf{pos}'(i)$ denotes the position of i at the next evaluation point (after $\Delta\mathsf{t}$ time units). The way positions change (i.e. the relationship between pos and pos') is defined by the following set $\mathcal{K}_f = \{\mathsf{F1}, \mathsf{F2}, \mathsf{F3}, \mathsf{F4}\}$ of axioms[5]:

$$
\begin{array}{lll}
(\mathsf{F1}) & \forall i & (i = 0 \;\rightarrow\; \mathsf{pos}(i) + \Delta\mathsf{t}{*}\mathsf{min} \leq_{\mathbb{R}} \mathsf{pos}'(i) \leq_{\mathbb{R}} \mathsf{pos}(i) + \Delta\mathsf{t}{*}\mathsf{max}) \\
(\mathsf{F2}) & \forall i & (0 < i < \mathsf{n} \;\wedge\; \mathsf{pos}(i-1) >_{\mathbb{R}} 0 \;\wedge\; \mathsf{pos}(p(i)) - \mathsf{pos}(i) \geq_{\mathbb{R}} \mathsf{l}_{\mathsf{alarm}} \\
 & & \quad \rightarrow \mathsf{pos}(i) + \Delta\mathsf{t} * \mathsf{min} \leq_{\mathbb{R}} \mathsf{pos}'(i) \leq_{\mathbb{R}} \mathsf{pos}(i) + \Delta\mathsf{t}{*}\mathsf{max}) \\
(\mathsf{F3}) & \forall i & (0 < i < \mathsf{n} \;\wedge\; \mathsf{pos}(i-1) >_{\mathbb{R}} 0 \;\wedge\; \mathsf{pos}(p(i)) - \mathsf{pos}(i) <_{\mathbb{R}} \mathsf{l}_{\mathsf{alarm}} \\
 & & \quad \rightarrow \mathsf{pos}'(i) = \mathsf{pos}(i) + \Delta\mathsf{t}{*}\mathsf{min}) \\
(\mathsf{F4}) & \forall i & (0 < i < \mathsf{n} \;\wedge\; \mathsf{pos}(i-1) \leq_{\mathbb{R}} 0 \;\rightarrow\; \mathsf{pos}'(i) = \mathsf{pos}(i))
\end{array}
$$

Note that the train with number 0 is the train with the greatest position, i.e. we count trains from highest to lowest position.

Axiom F1 states that the first train may always move at any speed between min and max. F2 states that the other trains can do so if their predecessor has already started and the distance to it is larger than $\mathsf{l}_{\mathsf{alarm}}$. If the predecessor of a train has started, but is less than $\mathsf{l}_{\mathsf{alarm}}$ away, then the train may only move at speed min (axiom F3). F4 requires that a train may not move at all if its predecessor has not started.

Model 2: Incoming and leaving trains. If we allow incoming and leaving trains, we additionally need a measure for the number of trains on the track. This is given by additional constants first and last, which at any time give the number of the first and last train on the track (again, the first train is supposed to be the train with the highest position). Furthermore, the maximum number of trains that is allowed to be on the track simultaneously is given by a constant maxTrains. These three values replace the number of trains n in the simpler model, the rest of it remains the same except that the function pos is now defined for values between first and last, where before it was defined between 0 and $\mathsf{n} - 1$. The behavior of this extended system is described by the following set $\mathcal{K}_v$ consisting of axioms (V1) – (V9):

$$
\begin{array}{lll}
(\mathsf{V1}) & \forall i & (i = \mathsf{first} \rightarrow \mathsf{pos}(i) + \Delta\mathsf{t} * \mathsf{min} \leq_{\mathbb{R}} \mathsf{pos}'(i) \leq_{\mathbb{R}} \mathsf{pos}(i) + \Delta\mathsf{t} * \mathsf{max}) \\
(\mathsf{V2}) & \forall i & (\mathsf{first} < i \leq \mathsf{last} \wedge \mathsf{pos}(i-1) >_{\mathbb{R}} 0 \wedge \mathsf{pos}(i-1) - \mathsf{pos}(i) \geq_{\mathbb{R}} \mathsf{l}_{\mathsf{alarm}} \\
 & & \quad \rightarrow \mathsf{pos}(i) + \Delta\mathsf{t} * \mathsf{min} \leq_{\mathbb{R}} \mathsf{pos}'(i) \leq_{\mathbb{R}} \mathsf{pos}(i) + \Delta\mathsf{t} * \mathsf{max}) \\
(\mathsf{V3}) & \forall i & (\mathsf{first} < i \leq \mathsf{last} \wedge \mathsf{pos}(i-1) >_{\mathbb{R}} 0 \wedge \mathsf{pos}(i-1) - \mathsf{pos}(i) <_{\mathbb{R}} \mathsf{l}_{\mathsf{alarm}} \\
 & & \quad \rightarrow \mathsf{pos}'(i) = \mathsf{pos}(i) + \Delta\mathsf{t} * \mathsf{min}) \\
(\mathsf{V4}) & \forall i & (\mathsf{first} < i \leq \mathsf{last} \wedge \mathsf{pos}(i-1) \leq_{\mathbb{R}} 0 \rightarrow \mathsf{pos}'(i) = \mathsf{pos}(i)) \\
(\mathsf{V5}) & & \mathsf{last} - \mathsf{first} + 1 < \mathsf{maxTrains} \rightarrow \mathsf{last}' = \mathsf{last} \vee \mathsf{last}' = \mathsf{last} + 1 \\
(\mathsf{V6}) & & \mathsf{last} - \mathsf{first} + 1 = \mathsf{maxTrains} \rightarrow \mathsf{last}' = \mathsf{last} \\
(\mathsf{V7}) & & \mathsf{last} - \mathsf{first} + 1 > 0 \rightarrow \mathsf{first}' = \mathsf{first} \vee \mathsf{first}' = \mathsf{first} + 1
\end{array}
$$

[5] Inequality over integers is displayed without subscript, inequality over reals is marked with an $\mathbb{R}$

$$\text{(V8)} \qquad \mathsf{last} - \mathsf{first} + 1 = 0 \rightarrow \mathsf{first}' = \mathsf{first}$$
$$\text{(V9)} \qquad \mathsf{last}' = \mathsf{last} + 1 \quad \rightarrow \quad \mathsf{pos}'(\mathsf{last}') <_{\mathbb{R}} \mathsf{pos}'(\mathsf{last})$$

where primed symbols denote the state of the system at the next evaluation.

Here, axioms $\mathsf{V1}-\mathsf{V4}$ are similar to $\mathsf{F1}-\mathsf{F4}$, except that the fixed bounds are replaced by the constants first and last. V5 states that if the number of trains is less than maxTrains, then a new train may enter or not. V6 says that no train may enter if maxTrains is already reached. V7 and V8 are similar conditions for leaving trains. Finally, V9 states that if a train enters, its position must be behind the train that was last before.

The safety condition which is important for this type of systems is collision freeness. In [12] we use a very simplified model of the system of trains, where collision freeness is modeled by a 'strict monotonicity' property for the function pos which stores the positions of the trains. We now consider a more realistic axiomatization – assuming the maximum length of the trains is known – expressed by the following axiom:

$$\mathsf{SdMon}(\mathsf{pos}) \quad \forall\, i, j, k \;\; (0 \leq j < i < \mathsf{n} \wedge i - j = k \rightarrow \mathsf{pos}(j) - \mathsf{pos}(i) \geq_{\mathbb{R}} k * \mathsf{LengthTrain}),$$

where LengthTrain is the standard (resp. maximal) length of a train.

As base theory we consider the combination $\mathcal{T}_0$ of the theory of integers and reals with a multiplication operation $*$ of arity $\mathsf{i} \times \mathsf{num} \rightarrow \mathsf{num}$ (multiplication of k with the constant LengthTrain in the formula above)[6]. Let $\mathcal{T}_1$ be the theory obtained by extending $\mathcal{T}_0$ with a function pos satisfying the axiom above. By the results presented in Sect. 3 on locality of extensions with functions satisfying strict monotonicity and related properties, the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ is local.

We now extend the resulting theory $\mathcal{T}_1$ in two different ways, with the axiom sets for one of the two system models, respectively. Both extensions are extensions with guarded boundedness axioms. By the remarks in Sect. 3 both $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup \mathcal{K}_f$ and $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup \mathcal{K}_v$ are local extensions. The method for hierarchical reasoning described above allows us to reduce the problem of checking whether system properties such as collision freeness are inductive invariants to deciding satisfiability of corresponding constraints in $\mathcal{T}_0$. As a side effect, after the reduction of the problem to a satisfiability problem in the base theory, one can automatically determine constraints on the parameters (e.g. $\Delta\mathsf{t}, \mathsf{min}, \mathsf{max}, \ldots$) which guarantee that the property is an inductive invariant, and are sufficient for this. (This can be achieved for instance using quantifier elimination.) These constraints can be used, for instance, for optimization problems (e.g. maximize speed and Δt while guaranteeing safety).

[6] In the light of locality properties of such extensions (cf. Sect. 3), k will always be instantiated by values in a finite set of *concrete* integers, all within a given, *concrete* range; thus the introduction of this many-sorted multiplication does not affect decidability.

Illustration: We indicate how to apply hierarchical reasoning on the case study given in Section 5.1, Model 1[7]. We follow the steps given in Section 2.1 and show how the sets of formulas are obtained that can finally be handed to a prover of the base theory.

To check whether $\mathcal{T}_1 \cup \mathcal{K}_f \models \mathsf{ColFree}(\mathsf{pos}')$, where

$$\mathsf{ColFree}(\mathsf{pos}') \quad \forall i \quad (0 \leq i < \mathsf{n} - 1 \quad \rightarrow \quad \mathsf{pos}'(i) - \mathsf{pos}'(i+1) >_{\mathbb{R}} \mathsf{LengthTrain}),$$

we check whether $\mathcal{T}_1 \cup \mathcal{K}_f \cup G \models \bot$, where $G = \{0 \leq k < \mathsf{n} - 1, \quad k' = k + 1, \quad \mathsf{pos}'(k) - \mathsf{pos}'(k') \leq_{\mathbb{R}} \mathsf{LengthTrain}\}$ is the (skolemized) negation of $\mathsf{ColFree}(\mathsf{pos}')$, flattened by introducing a new constant k'.

Reduction from $\mathcal{T}_1 \cup \mathcal{K}_f$ to $\mathcal{T}_1$. This problem is reduced to a satisfiability problem over $\mathcal{T}_1$ as follows:

Step 1: Use locality. We construct the set $\mathcal{K}_f[G]$: There are no ground subterms with pos' at the root in $\mathcal{K}_f$, and only two ground terms with pos' in G, $\mathsf{pos}'(k)$ and $\mathsf{pos}'(k')$. This means that $\mathcal{K}_f[G]$ consists of two instances of $\mathcal{K}_f$: one with i instantiated to k, the other instantiated to k'. E.g., the two instances of F2 are:

$$\begin{aligned}(\mathsf{F2[G]}) \quad & (0 < k < \mathsf{n} \quad \wedge \quad \mathsf{pos}(k-1) >_{\mathbb{R}} 0 \quad \wedge \quad \mathsf{pos}(k-1) - \mathsf{pos}(k) \geq_{\mathbb{R}} \mathsf{l_{alarm}} \\ & \rightarrow \mathsf{pos}(k) + \Delta\mathsf{t} * \mathsf{min} \leq_{\mathbb{R}} \mathsf{pos}'(k) \leq_{\mathbb{R}} \mathsf{pos}(k) + \Delta\mathsf{t} * \mathsf{max}) \\ & (0 < k' < \mathsf{n} \quad \wedge \quad \mathsf{pos}(k'-1) >_{\mathbb{R}} 0 \quad \wedge \quad \mathsf{pos}(k'-1) - \mathsf{pos}(k') \geq_{\mathbb{R}} \mathsf{l_{alarm}} \\ & \rightarrow \mathsf{pos}(k') + \Delta\mathsf{t} * \mathsf{min} \leq_{\mathbb{R}} \mathsf{pos}'(k') \leq_{\mathbb{R}} \mathsf{pos}(k') + \Delta\mathsf{t} * \mathsf{max})\end{aligned}$$

The construction of (F1[G]), (F3[G]) and (F4[G]) is similar. In addition, we specify the known relationships between the constants of the system:
(Dom) $\Delta\mathsf{t} > 0 \quad \wedge \quad 0 \leq \mathsf{min} \quad \wedge \quad \mathsf{min} \leq \mathsf{max}$

Step 2: Flattening and purification. $\mathcal{K}_f[G] \wedge G$ is already flat w.r.t. pos'. We replace all ground terms with pos' at the root with new constants: we replace $\mathsf{pos}'(k)$ by c_1 and $\mathsf{pos}'(k')$ by c_2. We obtain a set of definitions $\mathsf{Def} = \{\mathsf{pos}'(k) = c_1, \mathsf{pos}'(k') = c_2\}$ and a set $(\mathsf{Dom}) \cup \mathsf{G_0} \cup \mathcal{K}_{f_0}$ of clauses which do not contain occurrences of pos', consisting of (Dom) together with:

$$\begin{aligned}(\mathsf{G_0}) \quad & 0 \leq k < \mathsf{n} - 1 \quad \wedge \quad k' = k + 1 \quad \wedge \quad c_1 - c_2 \leq_{\mathbb{R}} \mathsf{LengthTrain} \\ (\mathsf{F2_0}) \quad & (0 < k < \mathsf{n} \quad \wedge \quad \mathsf{pos}(k-1) >_{\mathbb{R}} 0 \quad \wedge \quad \mathsf{pos}(k-1) - \mathsf{pos}(k) \geq_{\mathbb{R}} \mathsf{l_{alarm}} \\ & \rightarrow \mathsf{pos}(k) + \Delta\mathsf{t} * \mathsf{min} \leq_{\mathbb{R}} c_1 \leq_{\mathbb{R}} \mathsf{pos}(k) + \Delta\mathsf{t} * \mathsf{max}) \\ & (0 < k' < \mathsf{n} \quad \wedge \quad \mathsf{pos}(k'-1) >_{\mathbb{R}} 0 \quad \wedge \quad \mathsf{pos}(k'-1) - \mathsf{pos}(k') \geq_{\mathbb{R}} \mathsf{l_{alarm}} \\ & \rightarrow \mathsf{pos}(k') + \Delta\mathsf{t} * \mathsf{min} \leq_{\mathbb{R}} c_2 \leq_{\mathbb{R}} \mathsf{pos}(k') + \Delta\mathsf{t} * \mathsf{max})\end{aligned}$$

The construction can be continued similarly for F1, F3 and F4.

Step 3: Reduction to satisfiability in $\mathcal{T}_1$. We add the functionality clause $N_0 = \{k = k' \rightarrow c_1 = c_2\}$ and obtain a satisfiability problem in $\mathcal{T}_1$: $\mathcal{K}_{f_0} \wedge G_0 \wedge N_0$.

[7] We illustrate our approach for the simplest model. For a variable number of trains the approach is the same.

The reduction is schematically represented in the following diagram:

Def	$\mathsf{Dom} \cup \mathsf{G}_0 \cup \mathcal{K}_{f\,0} \cup N_0$
$\mathsf{pos}'(k) = c_1$	$(\mathsf{Dom}) \cup (\mathsf{G}_0) \cup (\mathsf{F}_{1\,0}) \cup (\mathsf{F2}_0) \cup (\mathsf{F3}_0) \cup (\mathsf{F4}_0)$
$\mathsf{pos}'(k') = c_2$	$N_0 : k = k' \rightarrow c_1 = c_2$

Reduction from $\mathcal{T}_1$ to $\mathcal{T}_0$. To decide satisfiability of $\mathcal{T}_1 \wedge \mathcal{K}_{f_0} \wedge G_0 \wedge N_0$, we have to perform another transformation w.r.t. the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$. The resulting set of ground clauses can directly be handed to a decision procedure for the combination of the theory of indices and that of reals. We flatten and purify the set $\mathcal{K}_{f_0} \wedge G_0 \wedge N_0$ of ground clauses w.r.t. pos by introducing new constants denoting $k-1$ and $k'-1$, together with their definitions $k'' = k-1, k''' = k'-1$; as well as constants d_1 for $\mathsf{pos}(k)$, d_2 for $\mathsf{pos}(k')$, d_3 for $\mathsf{pos}(k'')$ and d_4 for $\mathsf{pos}(k''')$ together with their definitions. Taking into account only the corresponding instances of the collision freeness axiom for pos we obtain a set of clauses consisting of (Dom) together with:

(G_0') $\quad k'' = k-1 \;\wedge\; k''' = k'-1$

(G_0) $\quad 0 \le k < n-1 \;\wedge\; k' = k+1 \;\wedge\; c_1 - c_2 \le_{\mathbb{R}} \mathsf{LengthTrain}$

$(\mathsf{GF1}_0)$ $\quad k = 0 \;\rightarrow\; d_1 + \Delta t{*}\mathsf{min} \le_{\mathbb{R}} c_1 \le_{\mathbb{R}} d_1 + \Delta t{*}\mathsf{max}$
$\qquad\qquad k' = 0 \;\rightarrow\; d_2 + \Delta t{*}\mathsf{min} \le_{\mathbb{R}} c_2 \le_{\mathbb{R}} d_2 + \Delta t{*}\mathsf{max}$

$(\mathsf{GF2}_0)$ $\quad 0{<}k{<}n \;\wedge\; d_3{>}_{\mathbb{R}}0 \;\wedge\; d_3{-}d_1 \ge_{\mathbb{R}} \mathsf{l_{alarm}} \rightarrow d_1{+}\Delta\mathsf{t}{*}\mathsf{min} \le_{\mathbb{R}} c_1 \le_{\mathbb{R}} d_1{+}\Delta\mathsf{t}{*}\mathsf{max}$
$\qquad\qquad 0{<}k'{<}n \;\wedge\; d_4{>}_{\mathbb{R}}0 \;\wedge\; d_4{-}d_2 \ge_{\mathbb{R}} \mathsf{l_{alarm}} \rightarrow d_2{+}\Delta\mathsf{t}{*}\mathsf{min} \le_{\mathbb{R}} c_2 \le_{\mathbb{R}} d_2{+}\Delta\mathsf{t}{*}\mathsf{max}$

$(\mathsf{GF3}_0)$ $\quad 0{<}k{<}n \;\wedge\; d_3{>}_{\mathbb{R}}0 \;\wedge\; d_3{-}d_1 <_{\mathbb{R}} l_{\mathsf{alarm}} \rightarrow c_1 = d_1{+}\Delta t{*}\mathsf{min}$
$\qquad\qquad 0{<}k'{<}n \;\wedge\; d_4{>}_{\mathbb{R}}0 \;\wedge\; d_4{-}d_2 <_{\mathbb{R}} l_{\mathsf{alarm}} \rightarrow c_2 = d_2{+}\Delta t{*}\mathsf{min}$

$(\mathsf{GF4}_0)$ $\quad (0{<}k{<}n \;\wedge\; d_3 \le_{\mathbb{R}} 0 \;\rightarrow\; c_1{=}d_1) \wedge (0{<}k'{<}n \;\wedge\; d_4 \le_{\mathbb{R}} 0 \;\rightarrow\; c_2{=}d_2)$

$\mathsf{SdMon(pos)[G']}$
$$
\begin{array}{rcl}
0 \le k < k' < \mathsf{n} & \rightarrow & d_1 - d_2 >_{\mathbb{R}} (k'-k) * \mathsf{LengthTrain} \\
0 \le k' < k < \mathsf{n} & \rightarrow & d_2 - d_1 >_{\mathbb{R}} (k-k') * \mathsf{LengthTrain} \\
0 \le k < k'' < \mathsf{n} & \rightarrow & d_1 - d_3 >_{\mathbb{R}} (k''-k) * \mathsf{LengthTrain} \\
0 \le k'' < k < \mathsf{n} & \rightarrow & d_3 - d_1 >_{\mathbb{R}} (k-k'') * \mathsf{LengthTrain} \\
0 \le k < k''' < \mathsf{n} & \rightarrow & d_1 - d_4 >_{\mathbb{R}} (k'''-k) * \mathsf{LengthTrain} \\
0 \le k''' < k < \mathsf{n} & \rightarrow & d_4 - d_1 >_{\mathbb{R}} (k-k''') * \mathsf{LengthTrain} \\
0 \le k' < k'' < \mathsf{n} & \rightarrow & d_2 - d_3 >_{\mathbb{R}} (k''-k') * \mathsf{LengthTrain} \\
0 \le k'' < k' < \mathsf{n} & \rightarrow & d_3 - d_2 >_{\mathbb{R}} (k'-k'') * \mathsf{LengthTrain} \\
0 \le k' < k''' < \mathsf{n} & \rightarrow & d_2 - d_4 >_{\mathbb{R}} (k'''-k') * \mathsf{LengthTrain} \\
0 \le k''' < k' < n & \rightarrow & d_4 - d_2 >_{\mathbb{R}} (k'-k''') * \mathsf{LengthTrain} \\
0 \le k'' < k''' < n & \rightarrow & d_3 - d_4 >_{\mathbb{R}} (k'''-k'') * \mathsf{LengthTrain} \\
0 \le k''' < k'' < n & \rightarrow & d_4 - d_3 >_{\mathbb{R}} (k''-k''') * \mathsf{LengthTrain}
\end{array}
$$

$N_0(\mathsf{pos}')$ $\quad k = k' \rightarrow c_1 = c_2$
$N_0(\mathsf{pos})$ $\quad k = k' \rightarrow d_1 = d_2 \;\wedge\; k = k'' \rightarrow d_1 = d_3 \;\wedge\; k' = k''' \rightarrow d_2 = d_4$
$\qquad\qquad k = k''' \rightarrow d_1 = d_4 \wedge k' = k'' \rightarrow d_2 = d_3 \wedge k'' = k''' \rightarrow d_3 = d_4$

In fact, the constraints on indices can help to further simplify the instances of monotonicity of $\mathsf{Mon}(\mathsf{pos})[G'] \wedge N_0(\mathsf{pos}) \wedge N_0(\mathsf{pos}')$: $k' > k, k'' < k, k'' < k', k''' < k', k''' = k$. The set of clauses equivalent to $\mathsf{SdMon}(\mathsf{pos})[G'] \wedge N_0(\mathsf{pos}) \wedge N_0(\mathsf{pos}')$ is given below. (Here we do these simplifications by hand; this can be done as well by a pre-simplification program which detects obviously true relationships between the premises of these rules.) After making these simplifications we obtain the following set of (many-sorted) constraints:

$C_{\mathsf{Definitions}}$	C_{Indices} (sort i)	C_{Reals}	C_{Mixed}
$\mathsf{pos}'(k) = c_1$	$k' = k + 1$	$d_1 - d_2 >_{\mathbb{R}} \mathsf{LengthTrain}$	$0 \leq k'' \rightarrow d_3 - d_2 >_{\mathbb{R}} 2 * \mathsf{LengthTrain}$
$\mathsf{pos}(k') = d_2$	$k'' = k - 1$	$d_3 - d_4 >_{\mathbb{R}} \mathsf{LengthTrain}$	$0 \leq k'' \rightarrow d_3 - d_1 >_{\mathbb{R}} \mathsf{LengthTrain}$
$\mathsf{pos}'(k') = c_2$	$k''' = k' - 1$	$d_3 - d_2 >_{\mathbb{R}} \mathsf{LengthTrain}$	$(\mathsf{GF1}_0)$
$\mathsf{pos}(k'') = d_3$		$d_4 - d_2 >_{\mathbb{R}} \mathsf{LengthTrain}$	$(\mathsf{GF2}_0)$
$\mathsf{pos}(k) = d_1$	$0 \leq k < n - 1$	$d_1 = d_4 \wedge c_1 - c_2 \leq_{\mathbb{R}}$	$(\mathsf{GF3}_0)$
$\mathsf{pos}(k''') = d_4$	$0 \leq k' < n - 1$	(Dom)	$(\mathsf{GF4}_0)$

For checking the satisfiability of $C_{\mathsf{Indices}} \wedge C_{\mathsf{Reals}} \wedge C_{\mathsf{Mixed}}$ we can use a prover for the two-sorted combination of the theory of integers and the theory of reals, possibly combined with a DPLL methodology for dealing with full clauses. An alternative method is discussed in [12].

5.2 Verification of a program changing pointer structures

Consider the following algorithm for inserting an element c with priority field x into a doubly-linked list sorted according to the priority fields.

$c.\mathsf{priority} = x, c.\mathsf{next} = \mathsf{null}$
for all $p \neq c$ **do**
  if $p.\mathsf{priority} \leq x$ **then if** $p.\mathsf{prev} = \mathsf{null}$ then $c.\mathsf{next}' = p$, **endif**; $p.\mathsf{next}' = p.\mathsf{next}$
    $p.\mathsf{priority} > x$ **then** **case** $p.\mathsf{next} = \mathsf{null}$ **then** $p.\mathsf{next}' := c, c.\mathsf{next}' = \mathsf{null}$
      $p.\mathsf{next} \neq \mathsf{null}$ and $p.\mathsf{next} > x$ **then** $p.\mathsf{next}' = p.\mathsf{next}$
      $p.\mathsf{next} \neq \mathsf{null}$ and $p.\mathsf{next} \leq x$ **then** $p.\mathsf{next}' = c$,
        $c.\mathsf{next}' = p.\mathsf{next}$

The update rules $\mathsf{Update}(\mathsf{next}, \mathsf{next}')$ can be read from the program above:

$\forall p(p{\neq}\mathsf{null} \wedge p{\neq}c \wedge \mathsf{priority}(p){\leq}x \wedge (\mathsf{prev}(p) = \mathsf{null}) \rightarrow \mathsf{next}'(c){=}p \wedge \mathsf{next}'(p){=}\mathsf{next}(p))$
$\forall p(p{\neq}\mathsf{null} \wedge p{\neq}c \wedge \mathsf{priority}(p){\leq}x \wedge (\mathsf{prev}(p) \neq \mathsf{null}) \rightarrow \mathsf{next}'(p){=}\mathsf{next}(p))$
$\forall p(p{\neq}\mathsf{null} \wedge p{\neq}c \wedge \mathsf{priority}(p){>}x \wedge \mathsf{next}(p){=}\mathsf{null} \rightarrow \mathsf{next}'(p){=}c \wedge \mathsf{next}'(c){=}\mathsf{null})$
$\forall p(p{\neq}\mathsf{null} \wedge p{\neq}c \wedge \mathsf{priority}(p){>}x \wedge \mathsf{next}(p){\neq}\mathsf{null} \wedge \mathsf{priority}(\mathsf{next}(p)){>}x \rightarrow \mathsf{next}'(p){=}\mathsf{next}(p))$
$\forall p(p{\neq}\mathsf{null} \wedge p{\neq}c \wedge \mathsf{priority}(p){>}x \wedge \mathsf{next}(p){\neq}\mathsf{null} \wedge \mathsf{priority}(\mathsf{next}(p)){\leq}x \rightarrow \mathsf{next}'(p){=}c \wedge \mathsf{next}'(c){=}\mathsf{next}(p))$

We prove that if the list is sorted, it remains so after insertion, i.e. the formula:

$$\forall p(p \neq \mathsf{null} \wedge \mathsf{next}(p) \neq \mathsf{null} \rightarrow \mathsf{priority}(p) \geq \mathsf{priority}(\mathsf{next}(p))) \wedge \mathsf{Update}(\mathsf{next}, \mathsf{next}') \wedge (d \neq \mathsf{null} \wedge \mathsf{next}'(d) \neq \mathsf{null} \wedge \neg \mathsf{priority}(d) \geq \mathsf{priority}(\mathsf{next}'(d)))$$

is unsatisfiable in the extension $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathsf{Update}(\mathsf{next}, \mathsf{next}')$ of the theory $\mathcal{T}_0$ of doubly linked lists with a monotone priority field. The update rules are guarded

boundedness axioms, so the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ is local. Hence, the satisfiability task above w.r.t. $\mathcal{T}_1$ can be reduced to a satisfiability task w.r.t. $\mathcal{T}_0$ as follows:

Def	$\mathsf{Update}_0 \wedge G_0 \wedge N_0$
$\mathsf{next}'(d)=d_1$	$d\neq\mathsf{null} \wedge d\neq c \wedge \mathsf{priority}(d)\leq x \wedge \mathsf{prev}(d)=\mathsf{null} \rightarrow c_1=d$
	$d\neq\mathsf{null} \wedge d\neq c \wedge \mathsf{priority}(d)\leq x \wedge \mathsf{prev}(d)=\mathsf{null} \rightarrow d_1=\mathsf{next}(d)$
$\mathsf{next}'(c)=c_1$	$d\neq\mathsf{null} \wedge d\neq c \wedge \mathsf{priority}(d)\leq x \wedge \mathsf{prev}(d)\neq\mathsf{null} \rightarrow d_1=\mathsf{next}(d)$
	$d\neq\mathsf{null} \wedge d\neq c \wedge \mathsf{priority}(d)>x \wedge \mathsf{next}(d)=\mathsf{null} \rightarrow d_1=c \wedge c_1=\mathsf{null}$
	$d\neq\mathsf{null} \wedge d\neq c \wedge \mathsf{priority}(d)>x \wedge \mathsf{next}(d)\neq\mathsf{null} \wedge \mathsf{priority}(\mathsf{next}(d))<x \rightarrow d_1=\mathsf{next}(d)$
	$d\neq\mathsf{null} \wedge d\neq c \wedge \mathsf{priority}(d)>x \wedge \mathsf{next}(d)\neq\mathsf{null} \wedge \mathsf{priority}(\mathsf{next}(d))\leq x \rightarrow d_1=c$
	$d\neq\mathsf{null} \wedge d\neq c \wedge \mathsf{priority}(d)>x \wedge \mathsf{next}(d)\neq\mathsf{null} \wedge \mathsf{priority}(\mathsf{next}(d))\leq x \rightarrow c_1=\mathsf{next}(d)$
	$(d \neq \mathsf{null} \wedge \mathsf{next}'(d) \neq \mathsf{null} \wedge \neg\mathsf{priority}(d) \geq \mathsf{priority}(\mathsf{next}'(d)))$
	$d=c \rightarrow d_1=c_1$

The stable locality of $\mathcal{T}_0$ can be used to check the satisfiability of $G' = \mathsf{Update}_0 \wedge G_0 \wedge N_0$ w.r.t. $\mathcal{T}_0$; for this the instantiation method in [7] can be used.

5.3 Verification of a program handling arrays

Consider a parametric number m of processes. The priorities associated with the processes (non-negative real numbers) are stored in an array p. The states of the process – enabled (1) or disabled (0) – are stored in an array a. At each step only the process with maximal priority is enabled, its priority is set to x and the priorities of the waiting processes are increased by y. This can be expressed with the following set of axioms which we denote by $\mathsf{Update}(p, p', a, a')$

$$\begin{array}{l}
\forall i(1 \leq i \leq m \wedge \quad (\forall j(1 \leq j \leq m \wedge j\neq i \rightarrow p(i)>p(j))) \rightarrow a'(i) = 1) \\
\forall i(1 \leq i \leq m \wedge \quad (\forall j(1 \leq j \leq m \wedge j\neq i \rightarrow p(i)>p(j))) \rightarrow p'(i)=x) \\
\forall i(1 \leq i \leq m \wedge \neg(\forall j(1 \leq j \leq m \wedge j\neq i \rightarrow p(i)>p(j))) \rightarrow a'(i)=0) \\
\forall i(1 \leq i \leq m \wedge \neg(\forall j(1 \leq j \leq m \wedge j\neq i \rightarrow p(i)>p(j))) \rightarrow p'(i)=p(i)+y)
\end{array}$$

where x and y are considered to be parameters. We may be interested in checking whether the priority list remains injective after the update provided it was injective at the beginning, i.e. check the satisfiability of the following formula:

$$\forall i, j(1 \leq i \leq m \wedge 1 \leq j \leq m \wedge i \neq j \rightarrow p(i) \neq p(j)) \wedge \mathsf{Update}(p, p', a, a') \wedge \\ (1 \leq c \leq m \wedge 1 \leq d \leq m \wedge c \neq d \wedge p'(c) = p'(d)) \models \bot.$$

There are several proof tasks related to the proof of the property above:

- check satisfiability of the formula above for *given, fixed values* of x and y
- check satisfiability of the formula above assuming that x and y satisfy certain constraints, or
- determine constraints on x and y which guarantee unsatisfiability of the formula (i.e. guarantee that the priority list remains injective after the update).

The problem can be formalized as follows. Let $\mathcal{T}_0$ be a combination of the theory $\mathbb{Z}$ of integers (for indices), $\mathbb{R}$ of real numbers (priorities), and (the theory of) $\{0, 1\}$ (enabled/disabled). We consider:

(i) The extension $\mathcal{T}_1$ of $\mathcal{T}_0$ with the functions $a : \mathbb{Z} \to \{0,1\}$ (a free function) and $p : \mathbb{Z} \to \mathbb{R}$ (which satisfies the injectivity axiom $\mathsf{Inj}(p)$);
(ii) The extension $\mathcal{T}_2$ of $\mathcal{T}_1$ with the functions $a' : \mathbb{Z} \to \{0,1\}$, $p' : \mathbb{Z} \to \mathbb{R}$ satisfying the update axioms $\mathsf{Update}(p, p'a, a')$.

The second extension, $\mathcal{T}_1 \subseteq \mathcal{T}_2$, is an extension with guarded boundedness axioms, thus by the results in Sect. 3 it is local. Using locality we obtain the following reduction of the task of proving $\mathcal{T}_2 \wedge G \models \bot$ to a satisfiability problem w.r.t. $\mathcal{T}_1$. We take into account only the instances of $\mathsf{Update}(p, p', a, a')$ which contain ground terms occurring in G. This means that the axioms containing a' do not need to be considered. After purification and skolemization of the existentially quantified variables we obtain:

Def	$\mathsf{Update}_0 \wedge G_0 \wedge N_0$
$p'(c) = c_1$	$1 \leq c \leq m \wedge (1 \leq k_c \leq m \wedge k_c \neq c \rightarrow p(c) > p(k_c)) \rightarrow c_1 = x)$
$p'(d) = d_1$	$1 \leq d \leq m \wedge (1 \leq k_d \leq m \wedge k_d \neq d \rightarrow p(d) > p(k_d)) \rightarrow d_1 = x)$
	$\forall j (1 \leq j \leq m \wedge j \neq c \rightarrow p(c) > p(j)) \vee (1 \leq c \leq m \rightarrow c_1 = p(c) + y)$
	$\forall j (1 \leq j \leq m \wedge j \neq d \rightarrow p(d) > p(j)) \vee (1 \leq d \leq m \rightarrow d_1 = p(d) + y)$
	$1 \leq c \leq m \wedge 1 \leq d \leq m \wedge c \neq d \wedge c_1 = d_1$
	$c = d \rightarrow c_1 = d_1$

We reduced the problem to checking satisfiability w.r.t. $\mathcal{T}_1$ of the formula $G_1 = \mathsf{Update}_0 \wedge G_0 \wedge N_0$, which contains universal quantifiers. Let $G_1 = G_g \wedge G_\forall$, where G_g is the ground part of G and $G_\forall$ the part of G containing universally quantified variables. We need to check whether $\mathcal{T}_0 \wedge \mathsf{Inj} \wedge G_\forall \wedge G_g \models \bot$. The idea is now to note that extensions of injectivity axioms and boundedness are local, and thus $\mathcal{T}_0 \subseteq \mathcal{T}_0 \wedge \mathsf{Inj} \wedge G_\forall$. This makes the following reduction possible:

Def_1	$\mathsf{Inj}_0 \wedge G_{\forall 0} \wedge G_g \wedge N_0'$
$p(c) = c_2$	$(1 \leq j \leq m \wedge j \neq c \rightarrow c_2 > p(j)) \vee (1 \leq c \leq m \rightarrow c_1 = c_2 + y)$
$p(d) = d_2$	$(1 \leq j \leq m \wedge j \neq d \rightarrow d_2 > p(j)) \vee (1 \leq d \leq m \rightarrow d_1 = d_2 + y)$
$p(k_c) = c_3$	$1 \leq i \neq j \leq m \rightarrow p(i) \neq p(j)$
$p(k_d) = d_3$	where i, j are instantiated with c, d, k_c, k_d and definitions are applied
	$1 \leq c \leq m \wedge (1 \leq k_c \leq m \wedge k_c \neq c \rightarrow c_2 > c_3) \rightarrow c_1 = x$
	$1 \leq d \leq m \wedge (1 \leq k_d \leq m \wedge k_d \neq d \rightarrow d_2 > d_3) \rightarrow d_1 = x$
	$1 \leq c \leq m \wedge 1 \leq d \leq m \wedge c \neq d \wedge c_1 = d_1$
	$c = d \rightarrow c_1 = d_1$
	$c = d \rightarrow c_2 = d_2, \quad c = k_c \rightarrow c_2 = c_3, \quad c = k_d \rightarrow c_2 = d_3,$
	$d = k_c \rightarrow d_2 = c_3, \quad d = k_d \rightarrow d_2 = d_3, \quad k_c = k_d \rightarrow c_3 = d_3$

We can use a prover for a combination of integers and reals to determine whether this formula is satisfiable or symbolic computation packages performing quantifier elimination over the combined theory to derive constraints between x and y which guarantee injectivity after update.

6 Implementation and experiments

We have implemented the approach for hierarchical reasoning. Our tool (LoRe) allows us to reduce satisfiability problems in an extended theory to a base theory for which we can then use existing solvers. It takes as input the axioms of the theory extension, the ground goal and the list of extension function symbols. Chains of extensions are handled by having a list of axiom sets, and correspondingly a list of lists of extension function symbols. We follow the steps in Sect. 2.1:

- The input is analyzed for ground terms with extension functions at the root.
- After instantiating the axioms w.r.t. these terms, the extension symbols are hierarchically removed.
- The resulting formula is either given to a prover for a base theory, or taken as goal for another reduction (if we have a chain of extensions).

The program has two modes of execution. The first mode automatically produces inputs for a class of theorem provers for the base theory. Currently, we can produce base theory output for Yices, Mathsat and Redlog, but other solvers can be integrated easily. In the second mode, the interaction with the theorem prover for the base theory is integrated in the prover itself (the choice of the prover is settled by command line switch).

We run tests on various examples, including different versions of the train controller example [12,11] (cf. Sect. 5.1), an array version of the insertion algorithm, and reasoning in theories of arrays and lists.

In Fig. 1 we list running times for 8 different versions of the train example, each of the following four versions in a satisfiable and an unsatisfiable variant:

- simple (fixed number of trains, no length measure),
- length (fixed number of trains with length),
- variable (variable number of trains, no length measure),
- full (variable number of trains with length).

We set the timeout to 300 seconds. In Fig. 1, *not supported* means that we would have to hand a non-linear arithmetic problem to Yices. We can instead use Redlog. We have successfully used it for these problems.

While Yices can be used successfully also directly for unsatisfiable formulae, this does not hold if we change the input problem to a formula which is satisfiable w.r.t. the extended theory. In this case, Yices returns "unknown" after a 300 second timeout. After the reduction with our tool, Yices (applied to the problem for the base theory) returns "satisfiable" in a fraction of a second, and even supplies a model for this problem that can be lifted to a model in the extended theory for the initial set of clauses[8]. Even more information can be obtained using the quantifier elimination facilities offered e.g. by Redlog for determining *constraints between the parameters* of the problems which guarantee safety.

In Fig. 2 we list running times for two examples involving reasoning about arrays: an array insertion algorithm and an example considered in [6].

[8] The lifting is straightforward, given the output of our tool, but is not automated at the moment.

	$\|\mathcal{K}_1\|$	$\|\mathcal{K}_1[G]\|$	$\|\mathcal{K}_2\|$	$\|\mathcal{K}_2[G]\|$	Runtimes (sec)		
					LoRe	LoRe+Yices	Yices
simple $_{\mathsf{SAT}}$	5	20	4	8	0.028	0.049	timeout
simple $_{\mathsf{UNSAT}}$	6	21	4	8	0.027	0.047	0.032
length $_{\mathsf{SAT}}$	6	21	4	8	0.031	0.052*	not supported
length $_{\mathsf{UNSAT}}$	7	22	4	8	0.031	0.053*	not supported
variable $_{\mathsf{SAT}}$	8	71	9	21	0.128	0.183	timeout
variable $_{\mathsf{UNSAT}}$	9	72	9	21	0.135	0.181	0.052
full $_{\mathsf{SAT}}$	9	72	9	21	0.156	not supported	not supported
full $_{\mathsf{UNSAT}}$	10	73	9	21	0.167	not supported	not supported

The times for benchmark 3 and 4 are marked with '*' in order to point out that the replacement of the differences $(u - v) * \mathsf{LengthTrain}$, where $u, v \in \{k, k', k'', k'''\}$ has been done manually. This was necessary because the input for Yices cannot contain non-linear arithmetical expressions. We used the following abbreviations:

- $|\mathcal{K}_i|$: the number of clauses which define extension i in the chain of local extensions.
- $|\mathcal{K}_i[G]|$: the number of instances which we generate.
- **LoRe time:** the time needed by LoRe for the hierarchical reduction.
- **LoRe+Yices time:** the total time for reduction and use of Yices for SAT checking.
- **Yices time:** the time needed by Yices for solving the original problem.

Fig. 1. Tests with the RBC case study (various versions)

	$\|\mathcal{K}\|$	$\|G\|$	$\|\ \mathcal{K}[G]$	Runtimes (sec) (Mode 2; line switch -Yices)
array-insert	9	4	64	0.080
array-example [6]	10	11	640	0.400

We used the following abbreviations:
- $|\mathcal{K}|$: the number of clauses which define the extension.
- $|G|$: the size of the set of ground clauses to be proved/disproved.
- $|\mathcal{K}[G]|$: the number of instances which we generate.
- **Runtimes:** times in mode 2, using a direct call of Yices.

Fig. 2. Reasoning about arrays

We are working towards extending the tool support to stable locality, and for extensions with clauses containing proper first-order formulae.

7 Conclusions

We presented a general framework (based on locality) for identifying complex theories important in verification for which efficient (hierarchical and modular) reasoning methods exist. We showed that many theories of data structures studied in the verification literature are local extensions of a base theory. The general framework we use allows us to identify situations in which some of the syntactical restrictions imposed in previous papers can be relaxed. The list of theories we considered is not exhaustive (we do not mention, for instance, the theory of arrays with dimension studied in [14] which also fits within our framework).

The next step would be to integrate these methods into verification tools based on abstraction/refinement. Our work on hierarchical interpolation in local extensions [15] can be extended to many of the theories of data structures described in this paper. This is the topic of a separate paper.

Acknowledgements. This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS). See `www.avacs.org` for more information.

References

1. Givan, R., McAllester, D.: New results on local inference relations. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92), Morgan Kaufmann Press (1992) 403–412
2. McAllester, D.: Automatic recognition of tractability in inference relations. Journal of the Association for Computing Machinery **40** (1993) 284–303
3. Ganzinger, H.: Relating semantic and proof-theoretic concepts for polynomial time decidability of uniform word problems. In: Proc. 16th IEEE Symposium on Logic in Computer Science (LICS'01), IEEE Computer Society Press (2001) 81–92
4. Ganzinger, H., Sofronie-Stokkermans, V., Waldmann, U.: Modular proof systems for partial functions with Evans equality. Information and Computation **204** (2006) 1453–1492
5. Sofronie-Stokkermans, V.: Hierarchic reasoning in local theory extensions. In: 20th Int. Conf. on Automated Deduction (CADE-20), LNAI 3632, Springer (2005) 219–234
6. Bradley, A., Manna, Z., Sipma, H.: What's decidable about arrays? In: Verification, Model-Checking, and Abstract-Interpretation, 7th Int. Conf. (VMCAI 2006). LNCS 3855, Springer (2006) 427–442
7. McPeak, S., Necula, G.: Data structure specifications via local equality axioms. In: Computer Aided Verification, 17th International Conference, CAV 2005. LNCS 3576 (2005) 476–490
8. Sofronie-Stokkermans, V.: Hierarchical and modular reasoning in complex theories: The case of local theory extensions. In: Proc. 6th Int. Symp. Frontiers of Combining Systems *(FroCos 2007)* (Invited paper), LNCS 4720, Springer (2007) 47–71

9. Sofronie-Stokkermans, V., Ihlemann, C.: Automated reasoning in some local extensions of ordered structures. In: Proc. of ISMVL-2007, IEEE Computer Society (2007) http://dx.doi.org/10.1109/ISMVL.2007.10.
10. Sofronie-Stokkermans, V., Ihlemann, C.: Automated reasoning in some local extensions of ordered structures. Journal of Multiple-Valued Logic and Soft Computing **13** (2007) 397–414
11. Faber, J., Jacobs, S., Sofronie-Stokkermans, V.: Verifying CSP-OZ-DC specifications with complex data types and timing parameters. In: Integrated Formal Methods (IFM 2007), LNCS 4591, Springer (2007) 233–252
12. Jacobs, S., Sofronie-Stokkermans, V.: Applications of hierarchical reasoning in the verification of complex systems. Electronic Notes in Theoretical Computer Science **174** (2007) 39–54
13. Faber, J.: Verifying real-time aspects of the European Train Control System. In: Proceedings of the 17th Nordic Workshop on Programming Theory, University of Copenhagen, Denmark (2005) 67–70
14. Ghilardi, S., Nicolini, E., Ranise, S., Zucchelli, D.: Deciding extensions of the theory of arrays by integrating decision procedures and instantiation strategies. In: Proc. of the 10th European Conference on Logics in Artificial Intelligence (Jelia'06), LNCS 4160, Springer (2006) 177–189
15. Sofronie-Stokkermans, V.: Interpolation in local theory extensions. In: Automated Reasoning: 3rd International Joint Conference, IJCAR'2006. LNCS 4130, Springer (2006) 235–250