Local reasoning in verification

Viorica Sofronie-Stokkermans

Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, Saarbrücken, Germany
e-mail: sofronie@mpi-sb.mpg.de

Abstract. The goal of this paper is to illustrate the wide applicability in verification of results on local reasoning, and especially on hierarchical reasoning in local theory extensions. The paper contains a survey of our results on reasoning in local theory extensions, ranging from characterizations of locality to interpolation. In addition, several examples are provided, emphasizing theories occurring in a natural way in verification. We give several examples – some already existing in the literature, others obtained during the work in the AVACS project – of application domains where such theories occur in a natural way.

1 Introduction

Many problems in mathematics and computer science (and, especially, many problems occurring in the verification of complex systems) can be reduced to proving the satisfiability of conjunctions of literals in a background theory. This theory can be a concrete theory (e.g. the theory of real or rational numbers), the extension of a base theory with additional functions (free, monotone, or recursively defined) or a combination of theories. It is therefore very important to identify situations when reasoning in extensions and combinations of theories can be done efficiently.

In [6] and [15] we identified classes of theory extensions, which we called *local extensions*, for which efficient reasoning is possible. In a local extension of a theory $\mathcal{T}_0$ by a set $\mathcal{K}$ of clauses, unsatisfiability of any set G of ground clauses depends only of $\mathcal{T}_0$ and those instances of the extension clauses $\mathcal{K}$ which are 'similar' to the clauses in G, i.e. in which the extension terms are instantiated with ground terms already occurring in G. This is very helpful in several ways:

- First, it helps to limit search, since not all instances of $\mathcal{K}$ are needed to derive a contradiction.
- Second, locality often allows to reduce satisfiability checking for clauses with variables to testing satisfiability of ground clauses.
- In addition, in [6,15] we showed, as a by-product, that in local theory extensions *hierarchical reasoning* is possible (that is, for reasoning in a theory extension a prover for the base theory can be used as a black-box).

Many theories important for computer science or mathematics are local extensions of a base theory: several examples are presented in Sections 4–8 cf. also [15,16]. Results which allow to identify situations in which a combination of local

extensions of a base theory is guaranteed to be itself a local extension of the base theory [17] are discussed in Section 9.

These methods for reasoning in extensions and combinations of theories turn out to be very useful e.g. in invariant checking or bounded model checking. However, in the verification of complex systems we may need to automatically generate invariants, or to prove safety for all runs. For this we may need to use a goal-directed overapproximation for achieving faster termination, or to use abstraction (and apply methods from abstraction refinement). An idea which proved very useful is to use ground interpolants for abstraction refinement [10,11,12,18]. McMillan recently used interpolation also for invariant generation. As the results in [10,11,12] were restricted to a special class of data types, in [16] we investigated possibilities of obtaining simple interpolants in more expressive theory extensions. We identified situations in which it is possible to do this in a hierarchical manner, by using a prover and a procedure for generating interpolants in the base theory as "black-boxes".

The main goal of this paper is to show how widely applicable these results are in verification. The contribution of the paper has three aspects:

- First, we present a survey of our most recent results on reasoning in local theory extensions [6,15,17,16].
- Second, we give some new examples of classes of theories which occur in a natural way in verification which turn out to be local extensions of a simple base theory.
- Finally, we identify several examples from verification – some already existing in the literature, and others which we obtained during the work in the AVACS project – where these theories occur in a natural way.

We show that hierarchical, local reasoning can be used for proving certain invariants and in bounded model checking, and offers general theoretical arguments for possibilities limiting the search to neighborhoods of counterexamples without losing completeness, or for systematic slicing. We also present implications of efficient interpolation to abstraction refinement based verification.

The paper is structured as follows: Section 2 contains the definitions needed in the paper. Section 3 surveys our results on local theory extensions obtained in [15]. Sections 4–8 contain several examples of theories in this class, some already presented in [15,17,16], and some new. The theoretical results are illustrated with examples from verification where such theories occur in a natural way. Section 9 surveys our results on combinations of local theory extebsions in [17]. Section 10 contains some results on interpolation in local theory extensions [16].

2 Preliminaries

Partial algebras. A *partial Σ-algebra* is a structure $(A, \{f_A\}_{f\in\Sigma})$, where A is a non-empty set and for every $f \in \Sigma$ with arity n, f_A is a partial function from A^n to A. The structure is a *(total) algebra* if all functions f_A are total. (In

what follows we usually denote both an algebra and its support with the same symbol.) Details on partial algebras can be found in [3].

The notion of evaluating a term t with respect to a variable assignment $\beta : X \to A$ for its variables in a partial algebra A is the same as for total algebras, except that this evaluation is undefined if $t = f(t_1, \ldots, t_n)$ and either one of $\beta(t_i)$ is undefined, or else $(\beta(t_1), \ldots, \beta(t_n))$ is not in the domain of f_A.

Weak homomorphisms. A total map $h : A \to B$ between partial Σ-algebras A and B is called a *weak Σ-homomorphism* if whenever $f_A(a_1, \ldots, a_n)$ is defined (in A), then $f_B(h(a_1), \ldots, h(a_n))$ is defined (in B) and $h(f_A(a_1, \ldots, a_n)) = f_B(h(a_1), \ldots, h(a_n))$. Let Pred be a set of predicate symbols, and let $\Pi = (\Sigma, \mathsf{Pred})$. A *weak Π-embedding between two structures* whose restrictions to Σ are partial algebras, $(A, \{f_A\}_{f\in\Sigma}, \{P_A\}_{P\in\mathsf{Pred}})$ and $(B, \{f_B\}_{f\in\Sigma}, \{P_B\}_{P\in\mathsf{Pred}})$, is an injective weak Σ-homomorphism $i : A \to B$ which is an embedding w.r.t. Pred, i.e. for every $P \in \mathsf{Pred}$ with arity n and every $a_1, \ldots, a_n \in A$, $P_A(a_1, \ldots, a_n)$ if and only if $P_B(i(a_1), \ldots, i(a_n))$.

Weak validity. We define *weak validity* in structures $(A, \{f_A\}_{f\in\Sigma}, \{P_A\}_{P\in\mathsf{Pred}})$, where Pred is a set of predicate symbols and $(A, \{f_A\}_{f\in\Sigma})$ is a partial Σ-algebra. Let $\beta : X \to A$.

(1) $(A, \beta) \models_w t \approx s$ if and only if (a) $\beta(t)$ and $\beta(s)$ are both defined and equal; or (b) at least one of $\beta(s)$ and $\beta(t)$ is undefined.
(2) $(A, \beta) \models_w t \not\approx s$ if and only if (a) $\beta(t)$ and $\beta(s)$ are both defined and different; or (b) at least one of $\beta(s)$ and $\beta(t)$ is undefined.
(3) $(A, \beta) \models_w P(t_1, \ldots, t_n)$ if and only if (a) $\beta(t_1), \ldots, \beta(t_n)$ are all defined and $(\beta(t_1), \ldots, \beta(t_n)) \in P_A$; or (b) at least one of $\beta(t_1)$, $\ldots$, $\beta(t_n)$ is undefined.
(4) $(A, \beta) \models_w \neg P(t_1, \ldots, t_n)$ if and only if (a) $\beta(t_1), \ldots, \beta(t_n)$ are all defined and $(\beta(t_1), \ldots, \beta(t_n)) \notin P_A$; or (b) at least one of $\beta(t_1), \ldots, \beta(t_n)$ is undefined.

(A, β) *weakly satisfies a clause* C (notation: $(A, \beta) \models_w C$) if $(A, \beta) \models_w L$ for at least one literal L in C. *A weakly satisfies C* (notation: $A \models_w C$) if $(A, \beta) \models_w C$ for all assignments β. *A weakly satisfies a set of clauses $\mathcal{K}$* (notation: $A \models_w \mathcal{K}$) if $A \models_w C$ for all $C \in \mathcal{K}$.

Example 1 *Let A be a partial Σ-algebra, where $\Sigma = \{\mathsf{car}/1, \mathsf{nil}/0\}$. Assume that nil_A is defined and $\mathsf{car}_A(\mathsf{nil}_A)$ is not defined. Then $A \models_w \mathsf{car}(\mathsf{nil}) \approx \mathsf{nil}$ and $A \models_w \mathsf{car}(\mathsf{nil}) \not\approx \mathsf{nil}$ (because one term is not defined in A).*

Theories and extensions of theories. In what follows we will consider extensions of theories, in which the signature is extended by new *function symbols*. For the sake of simplicity we assume that the set of predicate symbols remains unchanged in the extension. A theory can be regarded as a set of formulae. Then extension with a set of formulae is set union. Alternatively, a theory $\mathcal{T}_0$ can be regarded as a collection of models. Then its extension with a set $\mathcal{K}$ of formulae consists of all structures (in the extended signature) which are models of $\mathcal{K}$ and whose reduct to the signature of $\mathcal{T}_0$ is in $\mathcal{T}_0$.

In what follows we regard theories as sets of formulae, if not otherwise specified. All the results of this paper can easily be reformulated to a setting in which $\mathcal{T}_0$

is a collection of models. (In a few cases in this paper we will explicitly refer to collections of models only; this will simplify the presentation.)

Let $\mathcal{T}_0$ be an arbitrary theory with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$, where the set of function symbols is Σ_0. We consider extensions $\mathcal{T}_1$ of $\mathcal{T}_0$ with signature $\Pi = (\Sigma, \mathsf{Pred})$, where the set of function symbols is $\Sigma = \Sigma_0 \cup \Sigma_1$. We assume that $\mathcal{T}_1$ is obtained from $\mathcal{T}_0$ by adding a set $\mathcal{K}$ of (universally quantified) clauses.

Weak partial model of a theory. A partial Π-algebra A is a *weak partial model* of $\mathcal{T}_1$ with totally defined Σ_0-function symbols if (i) $A_{|\Pi_0}$ is a model of $\mathcal{T}_0$ and (ii) A weakly satisfies all clauses in $\mathcal{K}$.

In what follows, if the base theory $\mathcal{T}_0$ and its signature are clear from the context, we will refer to *weak partial models* of $\mathcal{T}_1$. We will use the following notation:

- $\mathsf{PMod_w}(\Sigma_1, \mathcal{T}_1)$ denotes the class of all weak partial models of $\mathcal{T}_1$ in which the Σ_1-functions are partial and all other function symbols are total;
- $\mathsf{Mod}(\mathcal{T}_1)$ denotes the class of all models of $\mathcal{T}_1$ in which all functions in $\Sigma_0 \cup \Sigma_1$ are totally defined.

Embeddability. For theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$, where $\mathcal{K}$ is a set of clauses, and for the classes of partial algebras mentioned above, we consider the following condition:

($\mathsf{Emb_w}$) Every $A \in \mathsf{PMod_w}(\Sigma_1, \mathcal{T}_1)$ weakly embeds into a total model of $\mathcal{T}_1$.

We also define a stronger notion of embeddability, which we call *completability*:

($\mathsf{Comp_w}$) Every $A \in \mathsf{PMod_w}(\Sigma_1, \mathcal{T}_1)$ weakly embeds into a total model B of $\mathcal{T}_1$ such that $A_{|\Pi_0}$ and $B_{|\Pi_0}$ are isomorphic.

Corresponding weaker conditions, which only refer to embeddability of *finite* partial models, will be denoted by ($\mathsf{Emb^f_w}$), resp. ($\mathsf{Comp^f_w}$).

3 Local theory extensions

Local theories were introduced by McAllester and Givan in [7,9], and further studied by Ganzinger and Basin [1,5]. 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 already $\mathcal{K}[C] \models C$ (where $\mathcal{K}[C]$ is the set of instances of $\mathcal{K}$ in which all terms are subterms of ground terms in either $\mathcal{K}$ or C). The notion of locality in *equational* theories was studied by Ganzinger [5], who also related it to a semantical property, namely embeddability of partial algebras into total algebras. In [15] the notion of locality for Horn clauses is extended to the notion of *local extension* of a base theory.

Let $\mathcal{K}$ be a set of clauses in the signature $\Pi = (\Sigma_0 \cup \Sigma_1, \mathsf{Pred})$. In what follows, when we refer to sets G of ground clauses we assume that they are in the signature $\Pi^c = (\Sigma \cup \Sigma_c, \mathsf{Pred})$, where $\Sigma = \Sigma_0 \cup \Sigma_1$ and Σ_c is a set of new constants. If Ψ is a set of ground $\Sigma_0 \cup \Sigma_1 \cup \Sigma_c$-terms, we denote by $\mathcal{K}_\Psi$ the set of all instances of $\mathcal{K}$ in which all terms starting with a Σ_1-function symbol are ground terms in the set Ψ. If G is a set of ground clauses and $\Psi = \mathsf{st}(\mathcal{K}, G)$ is the set of ground subterms occurring in either $\mathcal{K}$ or G then we write $\mathcal{K}[G] := \mathcal{K}_\Psi$.

We will focus on the following type of locality of a theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$, where $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ with $\mathcal{K}$ a set of (universally quantified) clauses:

(Loc) For every set G of ground clauses $\mathcal{T}_1 \cup G \models \bot$ iff $\mathcal{T}_0 \cup \mathcal{K}[G] \cup G$ has no weak partial model in which all terms in $\mathsf{st}(\mathcal{K}, G)$ are defined.

A weaker notion (Loc^f) can be defined if we require that the respective conditions only hold for *finite* sets G of ground clauses. We say that an extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ is *local* if it satisfies condition (Loc^f). (A local theory [5] is a local extension of the empty theory.)

Let $\mathcal{T}_0 \subseteq \mathcal{T}_1{=}\mathcal{T}_0 \cup \mathcal{K}$ be a local theory extension. To check the satisfiability of a set G of ground clauses w.r.t. $\mathcal{T}_1$ we can proceed as follows (cf. [15]):

Step 1: Locality. By the locality condition, G is unsatisfiable w.r.t. $\mathcal{T}_1$ iff $\mathcal{K}[G] \wedge G$ has no weak partial model in which all the subterms of $\mathcal{K}[G] \wedge G$ are defined, and whose restriction to Π_0 is a total model of $\mathcal{T}_0$.

Step 2: Reduction to testing satisfiability in $\mathcal{T}_0$. We purify and flatten $\mathcal{K}[G] \wedge G$ by introducing new constants for the arguments of the extension functions as well as for the (sub)terms $t = f(g_1, \ldots, g_n)$ starting with extension functions $f \in \Sigma_1$, together with new corresponding definitions $c_t \approx t$. The set of clauses thus obtained has the form $\mathcal{K}_0 \wedge G_0 \wedge D$, where D is a set of ground unit clauses of the form $f(c_1, \ldots, c_n) \approx c$, where $f \in \Sigma_1$ and $c_1, \ldots, c_n, c$ are constants, and $\mathcal{K}_0, G_0$ are clause sets without function symbols in Σ_1. We reduce the problem to testing satisfiability in $\mathcal{T}_0$ by replacing D with the following set of clauses:

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

Theorem 1 ([15]) *With the notations above, the following are equivalent:*
(1) $\mathcal{T}_0 \wedge \mathcal{K} \wedge G$ has a model.
(2) $\mathcal{T}_0 \wedge \mathcal{K}[G] \wedge G$ has a w.p.model (where all terms in $\mathsf{st}(\mathcal{K}, \mathsf{G})$ are defined).
(3) $\mathcal{T}_0 \wedge \mathcal{K}_0 \wedge G_0 \wedge D$ has a w.p.model (with all terms in $\mathsf{st}(\mathcal{K}, \mathsf{G})$ defined).
(4) $\mathcal{T}_0 \wedge \mathcal{K}_0 \wedge G_0 \wedge N_0$ has a (total) Σ_0-model.

3.1 Example

Let $\mathcal{T}_0$ be a theory (with a binary predicate $\leq$), and $\mathcal{T}_1$ a local extension of $\mathcal{T}_0$ with two monotone functions f and g. Consider the following problem:

$$\mathcal{T}_0 \cup \mathsf{Mon}_f \cup \mathsf{Mon}_g \models \forall x, y, z, u, v (x \leq y \wedge f(y \vee z) \leq g(u \wedge v) \rightarrow f(x) \leq g(v))$$

The problem reduces to the problem of checking whether $\mathcal{T}_0 \cup \mathsf{Mon}_f \cup \mathsf{Mon}_g \cup G \models \bot$, where $G = c_0 \leq c_1 \wedge f(c_1 \vee c_2) \leq g(c_3 \wedge c_4) \wedge f(c_0) \not\leq g(c_4)$.

The locality of the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ means that, in order to test if $\mathcal{T}_0 \cup \mathsf{Mon}_f \cup \mathsf{Mon}_g \cup G \models \bot$, it is sufficient to test whether $\mathcal{T}_0 \cup \mathsf{Mon}_f[G] \cup \mathsf{Mon}_g[G] \cup G \models_w \bot$, where $\mathsf{Mon}_f[G], \mathsf{Mon}_g[G]$ consist of those instances of the monotonicity axioms for f and g in which the terms starting with the function symbols f and g occur already in G:

$\mathsf{Mon}_f[G] = c_0 \leq c_1 \vee c_2 \rightarrow f(c_0) \leq f(c_1 \vee c_2)$ $\quad$ $c_1 \vee c_2 \leq c_0 \rightarrow f(c_1 \vee c_2) \leq f(c_0)$

$\mathsf{Mon}_g[G] = c_4 \leq c_3 \wedge c_4 \rightarrow g(c_4) \leq g(c_3 \wedge c_4)$ $\quad$ $c_3 \wedge c_4 \leq c_4 \rightarrow g(c_3 \wedge c_4) \leq g(c_4)$

In order to check the satisfiability of the latter formula, we purify it, introducing definitions for the terms below the extension functions $d_1 = c_1 \vee c_2, d_2 = c_3 \wedge c_4$ as well as for the terms starting with the extension functions themeslves: $f(d_1) = e_1$, $f(c_0) = e_3$, $g(c_4) = e_4$, $g(d_2) = e_2$. As totality of the extension functions is not required, we can use a relational notation, and add the functionality axioms: $d_1 = c_0 \rightarrow e_1 = e_3$ and $c_4 = d_2 \rightarrow e_4 = e_2$. We obtain the following set of clauses:

$r_f(d_1, e_1)$	$d_1 = c_1 \vee c_2$	$c_0 \leq c_1$	$d_1 = c_0 \rightarrow e_1 = e_3$	$d_1 \leq c_0 \rightarrow e_1 \leq e_3$
$r_f(c_0, e_3)$	$d_2 = c_3 \wedge c_4$	$e_1 \leq e_2$	$d_2 = c_4 \rightarrow e_2 = e_4$	$c_0 \leq d1 \rightarrow e_3 \leq e_1$
$r_g(c_4, e_4)$		$e_3 \not\leq e_4$		$d_2 \leq c_4 \rightarrow e_2 \leq e_4$
$r_g(d_2, e_2)$				$d_4 \leq d_2 \rightarrow e_4 \leq e_2$

The following extensions are proved to be local in Theorem 7. We illustrate the hierarchical reduction to testing satisfiability in the base theory in each case.

(1) Assume $\mathcal{T}_0$ is $\mathcal{DL}$, the theory of distributive lattices or $\mathcal{B}$, the theory of Boolean algebras. The universal clause theory of $\mathcal{DL}$ (resp. $\mathcal{B}$) is the theory of the two element lattice (resp. two element Boolean algebra), so testing Boolean satisfiability is sufficient. (This is in NP.) We proved unsatisfiability using SPASS (`http://spass.mpi-sb.mpg.de/`), but SAT solvers such as for instance CHAFF (`http://www.princeton.edu/~chaff/software.html`) can be used as well.

(2) If $\mathcal{T}_0 = \mathcal{L}$ is the theory of lattices, then we can reduce the problem above to the problem of checking the satisfiability of a set of ground Horn clauses [14]. This can be checked in polynomial time.

(3) If $\mathcal{T}_0 = \mathbb{R}$ we first need to explain what $\vee$ and $\wedge$ are. For this, we replace $d_1 = c_1 \vee c_2$ with $(c_1 \leq c_2 \rightarrow d_1 = c_2) \wedge (c_2 < c_1 \rightarrow d_1 = c_1)$ and similarly for $d_2 = c_3 \wedge c_4$. We proved unsatisfiability using the REDLOG demo (`http://www.fmi.uni-passau.de/~sturm/software/redlog/`).

We can therefore conclude that in all cases above:

$$\mathcal{T}_1 \models \forall x, y, z, u, v (x \leq y \wedge f(y \vee z) \leq g(u \wedge v) \rightarrow f(x) \leq g(v)).$$

3.2 Identifying local theory extensions

There is a strong link between locality of a theory extension and embeddability of partial models into total ones. In [15] we proved that embeddability implies locality of an extension. This allows to identify several local theory extensions.

In what follows we say that a non-ground clause is Σ_1-*flat* if function symbols (including constants) do not occur as arguments of function symbols in Σ_1. A Σ_1-flat non-ground clause is called Σ_1-*linear* if whenever a variable occurs in two terms in the clause which start with function symbols in Σ_1, the two terms are identical, and if no term which starts with a function in Σ_1 contains two occurrences of the same variable.

Theorem 2 ([15]) *Let $\mathcal{K}$ be a set of clauses in which all terms starting with a function symbol in Σ_1 are flat and linear. If the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ satisfies* (Emb_w) *then it satisfies* (Loc).

Let $\mathcal{T}_0 \subseteq \mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ be a local theory extension. We can reduce testing satisfiability of a set of ground clauses w.r.t. $\mathcal{T}_1$, hierarchically, to testing the satisfiability of a formula w.r.t. $\mathcal{T}_0$ as shown in Section 3. If all variables in $\mathcal{K}$ occur below extension functions then this formula is guaranteed to be again ground.

Note: All the results above can be extended without problems to a many-sorted framework. We here chose a sort-neutral presentation for the sake of simplicity.

We present several examples of theory extensions for which embedding conditions among those mentioned above hold. Some examples first appear in [15,16], other examples are new – they occurred in our work on verification in the frame of the AVACS project. In particular, we present new general boundedness conditions which define local theory extensions relevant in verification. As illustration for the types of theory extensions we prove to be local, we present some examples from verification where the respective local theory extensions occur in a natural way. We show that hierarchical, local reasoning can be used for proving certain invariants and in bounded model checking, and offers general theoretical arguments for possibilities of systematically limiting the search to neighborhoods of counterexamples without losing completeness, or for systematic slicing.

4 Extensions with free and bounded 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 (Comp_w) [6,15].

In applications however, one often needs to consider additional constraints on free functions, e.g. boundedness or definedness constraints. Let $\mathcal{T}_0$ be a theory with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$ endowed with a reflexive partial ordering $\leq$.

Theorem 3 (Boundedness for free functions [16]) *Any extension of $\mathcal{T}_0$ with a function $f \notin \Sigma_0$ satisfying boundedness* (Bound_f^t) *or guarded boundedness* (GBound_f^t) *conditions is local.*

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

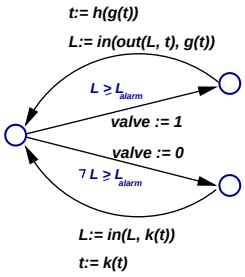
where $t(x_1, \ldots, x_n)$ is a term in the base signature Π_0 with variables among $x_1, \ldots, x_n$ and $\phi(x_1, \ldots, x_n)$ a conjunction of literals in signature Π_0 with variables among $x_1, \ldots, x_n$.

Theorem 4 (Piecewise boundedness for free functions) *For $i \in \{1, \ldots, m\}$, let $t_i(x_1, \ldots, x_n)$ and $s_i(x_1, \ldots, x_n)$ be terms in the signature Π_0 with variables among $x_1, \ldots, x_n$, and let $\phi_i(x_1, \ldots, x_n)$, $i \in \{1, \ldots, m\}$ be conjunctions of literals in the base signature Π_0, with variables among $x_1, \ldots, x_n$, i.e. such that for every $i \neq j$, $\phi_i \wedge \phi_j \models_{\mathcal{T}_0} \bot$. Any "piecewise-bounded" extension $\mathcal{T}_0 \wedge (\mathsf{GBound}_f)$, where $f \notin \Sigma_0$, is local. Here $(\mathsf{GBound}_f) = \bigwedge_{i=1}^m (\mathsf{GBound}_f^{[s_i, t_i], \phi_i})$;*

$$(\mathsf{GBound}_f^{[s_i, t_i], \phi_i}) \quad \forall \overline{x} (\phi_i(\overline{x}) \rightarrow s_i(\overline{x}) \leq f(\overline{x}) \leq t_i(\overline{x})).$$

4.1 Example: Water tank controller

Consider a water level controller modeled as follows: Assume that one can check, with the help of two sensors, whether the water level is above or below an alarm level L_{alarm}, and whether it is above or below an overflow level L_{overflow}, where $L_{\mathsf{alarm}} < L_{\mathsf{overflow}}$. Changes in the water level by inflow/outflow are represented as functions $\mathsf{in}, \mathsf{out}$, depending on current time t and water level L.



- If $L \geq L_{\mathsf{alarm}}$ then a valve is opened until time $g(t)$, the water level changes by $L' := \mathsf{in}(\mathsf{out}(L,t), g(t))$ and time by $t' := h(g(t))$.
- If $L < L_{\mathsf{alarm}}$ then the valve is closed; the water level changes by $L' := \mathsf{in}(L,t)$ and time by $t' := k(t)$.

We want to show that if initially $L < L_{\mathsf{alarm}}$ then the water level always remains below L_{overflow}.

We impose the following restrictions on the functions in and out:

$$\forall T(k(T) > 0) \qquad \forall T(g(T) > 0) \qquad \forall T(h(T) > 0) \tag{1}$$

$$\forall L, T(\mathsf{in}(L,T) > L) \qquad \forall L, T(\mathsf{out}(L,T) < L) \tag{2}$$

$$\forall L, T(L < L_{\mathsf{alarm}} \rightarrow \mathsf{in}(L,T) < L_{\mathsf{overflow}}) \qquad \forall L, T(L < L_{\mathsf{overflow}} \rightarrow \mathsf{out}(L,T) < L_{\mathsf{alarm}}) \tag{3}$$

By results in Section 4, the set of axioms (1)–(3) defines a local extension of linear real arithmetic. We prove that if initially $L < L_{\mathsf{alarm}}$ then

(1) Inv $L < L_{\mathsf{overflow}}$ is an invariant of the system.
(2) $\mathsf{Inv}(2)$ In state 2, always $L_{\mathsf{alarm}} < L < L_{\mathsf{overflow}}$.
(3) $\mathsf{Inv}(3)$ In state 3, always $L \leq L_{\mathsf{alarm}}$.

This can be done automatically as follows: As $L_{\mathsf{alarm}} < L_{\mathsf{overflow}}$, obviously, in the initial state $L < L_{\mathsf{alarm}} < L_{\mathsf{overflow}}$, hence the invariant is fulfilled. Assume that the invariant is fulfilled at state 1, i.e. $L < L_{\mathsf{overflow}}$. There are two possible transitions from state 1, according to the current water level.

- If $L > L_{\mathsf{alarm}}$ then the system passes to state 2, with $\mathsf{valve} := 1$. It is easy to check that the invariants are also fulfilled in state 2.
- If $\neg(L > L_{\mathsf{alarm}})$ then the system passes to state 3, with $\mathsf{valve} := 0$. It is easy to check that the invariants are also fulfilled in state 3.

Assume that the invariants are fulfilled at state 2. Then $L_{\mathsf{alarm}} < L < L_{\mathsf{overflow}}$. The only possible transition is from state 2 to state 1; the change in water level is given by $\mathsf{Tr}(21)$: $L' = \mathsf{in}(\mathsf{out}(L,t), g(t))$. We show, using the hierarchical method proposed in [15] that $\mathbb{R} \cup \mathcal{K} \cup \mathsf{Inv}(2) \cup \mathsf{Tr}(21) \cup \neg\mathsf{Inv}(1)$ is unsatisfiable, where:

$$\mathcal{K}: \quad \forall L, T(L < L_{\mathsf{alarm}} \rightarrow \mathsf{in}(L,T) < L_{\mathsf{overflow}}) \quad \forall L, T(L < L_{\mathsf{overflow}} \rightarrow \mathsf{out}(L,T) < L_{\mathsf{alarm}})$$

$$\mathsf{Inv}(2): \quad L_{\mathsf{alarm}} < L < L < L_{\mathsf{overflow}}$$
$$\mathsf{Tr}(21): \quad L' := \mathsf{in}(\mathsf{out}(L,t), g(t))$$
$$\neg\mathsf{Inv}(1): \quad \neg L' < L_{\mathsf{overflow}}.$$

Assume that the invariants are fulfilled at state 3. Then $L < L_{\mathsf{alarm}}$. The only possible transition is to state 1; the change in water level is given by $\mathsf{Tr}(31)$: $L' = \mathsf{in}(L, k(t))$. We show, using the hierarchical method in [15], that $\mathbb{R} \cup \mathcal{K} \cup \mathsf{Inv}(3) \cup \mathsf{Tr}(31) \cup \neg\mathsf{Inv}(1)$ is unsatisfiable, where:

$$\mathcal{K}: \quad \begin{array}{l} \forall L, T(L{<}L_{\mathsf{alarm}} \rightarrow \mathsf{in}(L,T){<}L_{\mathsf{overflow}}) \\ \forall L, T(L{<}L_{\mathsf{overflow}} \rightarrow \mathsf{out}(L,T){<}L_{\mathsf{alarm}}) \end{array} \qquad \begin{array}{rl} \mathsf{Inv}(3): & L \leq L_{\mathsf{alarm}} \\ \mathsf{Tr}(31): & L' = \mathsf{in}(L, k(t)) \\ \neg\mathsf{Inv}(1): & \neg L' < L_{\mathsf{overflow}}. \end{array}$$

5 Definedness restrictions

We now consider extensions with sets of clauses in which additionally definedness restrictions are taken into account. Let $\mathcal{K}$ be a set of clauses of the form

$$\bigwedge_{f(\overline{t}) \in \mathsf{Subterm}(\bigvee_{i=1}^{m} t_i = s_i)} \mathsf{Def}(f(\overline{t})) \rightarrow \bigvee_{i=1}^{m} t_i = s_i.$$

Let $\mathcal{K}_f$ be the set of clauses obtained from $\mathcal{K}$ by flattening, with the remark that the terms under Def are not completely replaced with variables, but only terms of the form $f(x_1, \ldots, x_n)$ can occur below Def.

Example 2 *Assume that $\mathcal{K}$ consists of the clause C:*

$\mathsf{Def}(\mathsf{car}(x)) \wedge \mathsf{Def}(\mathsf{cdr}(x)) \wedge \mathsf{Def}(\mathsf{cons}(\mathsf{car}(x), \mathsf{cdr}(x))) \rightarrow \mathsf{cons}(\mathsf{car}(x), \mathsf{cdr}(x)){=}x.$

Then $\mathcal{K}_f$ consists of the clause C_f:

$\mathsf{car}(x){=}y \wedge \mathsf{cdr}(x){=}z \wedge \mathsf{Def}(\mathsf{car}(x)) \wedge \mathsf{Def}(\mathsf{cdr}(x)) \wedge \mathsf{Def}(\mathsf{cons}(y,z)) \rightarrow \mathsf{cons}(y,z){=}x.$

Theorem 5 *With the notations above, $\mathcal{K}_f$ is a local theory.*

Proof: Let P be a partial model of $\mathcal{K}_f$. We embed P into a total model A of $\mathcal{K}_f$ with support $P \cup \{\bot\}$ as follows: for $f \in \Sigma$ and $p \in P$, if $f_P(p)$ is undefined we define $f_A(p) := \bot$, and $f_A(\bot) = \bot$. We define $\mathsf{Def}(\bot) = \mathsf{false}$. Let $C = \mathcal{K}_f$, and let $\beta : X \rightarrow A$. If $\beta(t)$ is defined in P for all terms occurring in C then as $(P, \beta) \models C$ also $(A, \beta) \models C$. If $\beta(f(x))$ is undefined in P for some subterm $f(x)$ in C then $\beta(f(x)) = f_A(\beta(x)) = \bot$, As $\neg\mathsf{Def}(f(x))$ is in C, $(A, \beta) \models C$.

5.1 Verification of pointer programs: Local data structures

In [13], McPeak and Necula investigate local reasoning in pointer data structures, with the goal of efficiently proving invariants in programs dealing with pointers. The logic 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. 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 [13] are of the form

$$\forall p \;\; \mathcal{E} \wedge \mathcal{C} \tag{4}$$

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

pointer terms $f_1(f_2(\dots f_n(p)))$ occurring in the body of an axiom, the axiom also contains the disjunction $p = \mathsf{null} \vee f_n(p) = \mathsf{null} \vee \dots \vee f_2(\dots f_n(p))) = \mathsf{null}$. This disjunction has the rôle of excluding null pointer errors. In addition, we assume that the disjunction contains $f_1(f_2(\dots f_n(p))) = \mathsf{null}$ if $f_1 \in \Sigma_p$ resp. $f_1(f_2(\dots f_n(p))) = \mathsf{null}_s$ if $f_1 \in \Sigma_s$.

Examples of axioms (for doubly linked data structures with state and priorities) which are considered there are:

$$\begin{array}{ll}
\forall p \;\; p \neq \mathsf{null} \wedge \mathsf{next}(p) \neq \mathsf{null} & \rightarrow \mathsf{prev}(\mathsf{next}(p)) = p \\
\forall p \;\; p \neq \mathsf{null} \wedge \mathsf{next}(p) \neq \mathsf{null} & \rightarrow \mathsf{state}(p) = \mathsf{state}(\mathsf{next}(p)) \\
\forall p \;\; p \neq \mathsf{null} \wedge \mathsf{next}(p) \neq \mathsf{null} \wedge \mathsf{state}(p) = \mathsf{RUN} & \rightarrow \mathsf{priority}(p) \geq \mathsf{priority}(\mathsf{next}(p))
\end{array}$$

(the first axiom states that prev is a left inverse for next, the second axiom tells how a state is updated; the third axiom is a monotonicity condition on the function priority with values in a partially ordered domain).
The special form (4) of the axioms ensures that all partial models can be embedded into total models.

Theorem 6 *Let $\mathcal{T}_0$ be the union of a two-sorted combination of a pointer theory without any function symbols and a given scalar theory. Any set of axioms of type (4) defines a local extension $\mathcal{T}_1$ of $\mathcal{T}_0$.*

Proof: Let $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ be the extension of $\mathcal{T}_0$ with clauses of the form $\forall p \;\; \mathcal{E} \wedge \mathcal{C}$. Let $P \in \mathsf{PMod}_w(\Sigma_p \cup \Sigma_s, \mathcal{T}_1)$. We embed P into a total model A of $\mathcal{T}_1$ with the same support as follows: for $f \in \Sigma_p$, if $f_P(p)$ is undefined we define $f_A(p) := \mathsf{null}$, for every $f \in \Sigma_s$, if $f_P(p)$ is undefined we define $f_A(p) = \mathsf{null}_s$. Let now $C = \mathcal{E} \vee \mathcal{C} \in \mathcal{K}$, and let $\beta : X \rightarrow A$. If $\beta(t)$ is defined in P for all terms occurring in C then as $(P, \beta) \models C$ also $(A, \beta) \models C$. If $\beta(t)$ is undefined in P for some term in C then $\beta(t) = \mathsf{null}$, hence the literal $t = \mathsf{null}$ in C is true, so $(A, \beta) \models C$. □

The results on local theory extensions can thus be used as well in this context.

6 Extensions with constructors and selectors

Let $\mathcal{T}_0$ be a theory with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$, let $c \in \Sigma_0$ with arity n, and let $\Sigma_1 = \{s_1, \dots, s_n\}$ consist of n unary function symbols. Let $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathsf{Sel}_c$ (a theory with signature $\Pi = (\Sigma_0 \cup \Sigma_1, \mathsf{Pred})$) be the extension of $\mathcal{T}_0$ with the set Sel_c of clauses below. Assume that $\mathcal{T}_0$ satisfies the (universally quantified) formula $\mathsf{Inj_c}$ (i.e. c is injective in $\mathcal{T}_0$), then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ satisfies condition ($\mathsf{Comp_w}$) [15].

$$\begin{array}{ll}
(\mathsf{Sel}_c) & s_1(c(x_1, \dots, x_n)) \approx x_1 \\
& \dots \\
& s_n(c(x_1, \dots, x_n)) \approx x_n \\
& x \approx c(x_1, \dots, x_n) \rightarrow c(s_1(x), \dots, s_n(x)) \approx x \\
(\mathsf{Inj_c}) & c(x_1, \dots, x_n) \approx c(y_1, \dots, y_n) \rightarrow (\bigwedge_{i=1}^{n} x_i \approx y_i)
\end{array}$$

Such extensions are used e.g. in programming, but also in cryptography (for encoding/decoding with given keys).

7 Extensions with monotone functions

Extensions with monotonicity axioms often are local as well.

Theorem 7 (Monotone functions [15]) *Let $\mathcal{T}_0$ be one of the theories in the class* Ord *consisting of: (1) $\mathcal{P}$ (posets), (2) $\mathcal{T}$ (totally-ordered sets), (3) $\mathcal{DO}$ (dense totally-ordered sets), (4) $\mathcal{S}$ (semilattices), (5) $\mathcal{L}$ (lattices), (6) $\mathcal{DL}$ (distributive lattices), (7) $\mathcal{B}$ (Boolean algebras), (8) $\mathbb{R}$ (theory of reals). Let Mon_f be the monotonicity axiom of an n-ary function f:*

$$(\mathsf{Mon}_f) \quad \bigwedge_{i=1}^{n} x_i \leq y_i \rightarrow f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n).$$

The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{Mon}_f$ satisfies condition $(\mathsf{Emb}_\mathsf{w})$ *in cases (1)–(5); condition* $(\mathsf{Comp}^\mathsf{f}_\mathsf{w})$ *in cases (6) and (7); and condition* $(\mathsf{Comp}_\mathsf{w})$ *in case (8).*

Similar results can be obtained if we assume that the class of models of $\mathcal{T}_0$ is a class of complete lattices (possibly with additional operators). Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{Mon}_f$ satisfies condition $(\mathsf{Comp}_\mathsf{w})$. (This is an easy consequence of the fact that the Dedekind-MacNeille completion of any complete lattice is isomorphic to the lattice.)

Adding boundedness does not destroy locality, provided that the bounding terms have the same monotonicity as the extension function.

Theorem 8 (Bounded monotone functions [16]) *Any extension of a theory in* Ord *with $\mathsf{Mon}_f \wedge \mathsf{Bound}^t_f$ is local, provided that $t(x_1, \ldots, x_n)$ is monotone in the variables $x_1, \ldots, x_n$.*

Blockwise and piecewise monotonicity also define local theory extensions if the base theory (for indices) $\mathcal{T}_0$ is the theory of integers or natural numbers. The extensions below are among the examples considered in [2].

Theorem 9 (Piecewise monotonicity) *Let $\mathcal{T}_0$ be the theory of integers or natural numbers. Assume that $l_1, \ldots, l_m, u_1, \ldots, u_m$ are given constants such that $l_1 \leq u_1 < l_2 \leq u_2 < \cdots < l_m \leq u_m$. Let f be a unary function. Any* piecewise-monotone *extension $\mathcal{T}_0 \wedge \mathsf{GMon}_f$) is local. Here $(\mathsf{GMon}_f) = (\mathsf{GMon}^{[l_1,u_1]}_f) \wedge \cdots \wedge \mathsf{GMon}^{[l_m,u_m]}_f)$, where:*

$$(\mathsf{GMon}^{[l_i,u_i]}_f) \quad \forall x, y(l_i \leq x \leq y \leq u_i \quad \rightarrow \quad f(x) \leq f(y)).$$

Theorem 10 (Blockwise monotonicity) *Let $\mathcal{T}_0$ be the theory of integers or natural numbers. Assume that $l_1, \ldots, l_m, u_1, \ldots, u_m$ are given constants such that $l_1 \leq u_1 < l_2 \leq u_2 < \cdots < l_m \leq u_m$. Let f be a unary function. Any* blockwise-monotone *extension $\mathcal{T}_0 \wedge (\mathsf{BMon}_f)$ is local. Here $(\mathsf{BMon}_f) = \bigwedge_{i=1}^{m-1}(\mathsf{BMon}^{[l_i,u_i],[l_{i+1},u_{i+1}]}_f)$, where:*

$$(\mathsf{BMon}^{[l_i,u_i],[l_{i+1},u_{i+1}]}_f) \quad \forall x, y(l_i \leq x \leq u_i < l_{i+1} \leq y \leq u_{i+1} \ \rightarrow \ f(x) \leq f(y)).$$

Similar conditions can be defined for n-ary functions with integer arguments. Strict inequality in the conclusion of boundedness rules (both in the case of free and in the case of monotone functions) and strict monotonicity can be handled too, under the assumption of density of the codomain of the functions.

Limits of decidability in reasoning about sorted arrays were explored in [2]. The decidability of satisfiability of ground clauses in the fragment of the theory of sorted arrays which we consider above is an easy consequence of the locality of such extensions.

7.1 Case study: Train controller

We present a simple example in the verification of a train controller (for details and more realistic rules we refer to [8]). We consider a controller which communicates with all the trains on a given linear track. Every train reports its position to the controller in given time intervals and the controller communicates to every train how far it can safely move, based on the position of the preceding train. The trains adjust their speed accordingly – the speed is supposed to range between given minimum and maximum speeds. A simple way in which this can be done can be described by the following rules[1], where the positions of trains are stored in two arrays a (for the current moment of time) and a' for their positions at the next evaluation point (after Δt seconds).

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

where the following constants are considered given: $\Delta t > 0$ (the time between evaluations of the system), min and max (the minimum and maximum speed of trains; we assume that $0 \leq \mathsf{min} \leq \mathsf{max}$), l_{alarm} (the distance between trains which is deemed secure), n (the number of trains).

An example of an invariant to be checked is collision-freeness. At a very abstract level, this can be expressed as a monotonicity axiom,

$$\mathsf{CF}(a) \quad \forall i, j \ \ (0 \leq i < j \leq n \rightarrow a(i) >_{\mathbb{R}} a(j)),$$

where $<$ is an ordering which expresses train precedence and $>_{\mathbb{R}}$ is the usual ordering on the real numbers. $\mathsf{CF}(a)$ expresses the condition that for all trains i, j on the track, if i precedes j then i should be positioned strictly ahead of j.

[1] 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 l_{alarm}. If the predecessor of a train has started, but is less than l_{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.

A more precise model (for the case when the length of trains is given) can be obtained by replacing the monotonicity axiom for a with the following axiom:

$$\forall i, j, k \quad (\mathsf{first} \leq j \leq i \leq \mathsf{last} \wedge i - j = k \rightarrow a(j) - a(i) \geq k * \mathsf{LengthTrain}),$$

where $\mathsf{LengthTrain}$ is the standard (resp. maximal) length of a train.

To check that collision-freeness is an invariant of the system, we check that the initial state is collision free and that collision freeness is preserved by the updating rules $\mathcal{K} = \{\mathsf{F}_1, ..., \mathsf{F}_4\}$, i.e. that in the extension $\mathcal{T}$ of a the base theory $\mathcal{T}_0$ (a many-sorted combination of real arithmetic – for reasoning about positions, sort num – with an index theory – for describing precedence between trains, sort i) with the two functions a and a' the following hold:

$$\mathcal{T} \models \mathcal{K} \wedge \mathsf{CF}(a) \rightarrow \mathsf{CF}(a'), \quad \text{i.e.} \quad \mathcal{T} \wedge \mathcal{K} \wedge \mathsf{CF}(a) \wedge \neg \mathsf{CF}(a') \models \bot .$$

For this, in [8] we considered two successive extensions of the base theory $\mathcal{T}_0$:

- the extension $\mathcal{T}_1$ of $\mathcal{T}_0$ with a monotone function a, of sort $\mathsf{i} \rightarrow \mathsf{num}$,
- the extension $\mathcal{T}_2$ of $\mathcal{T}_1$ with a function a' satisfying the update axioms $\mathcal{K}$.

The results in Sections 4 and 7 show that both these extensions are local. This allows us to choose only appropriate instances of the axioms in $\mathcal{K}$ and $\mathsf{CF}(a)$ representing the behavior of trains which are in a close neighborhood of the trains which would violate the safety condition. Similar methods can also be used for bounded model checking.

8 Comparisons between functions

Also theory extensions with clauses which specify relationships between extension functions often turn out to be local:

Theorem 11 (Comparisons between functions [16])

(1) Any theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ with functions f, g (possibly with output of a different sort, in a theory endowed with a reflexive predicate symbol $\leq$) satisfying condition $(\mathsf{Leq}(f, g))$ is local.
$(\mathsf{Leq}(f, g)) \quad \forall x_1, \ldots, x_n (f(x_1, \ldots, x_n) \leq g(x_1, \ldots, x_n))$

(2) Extensions of any theory in Ord with functions satisfying $\mathsf{Leq}(f, g) \wedge \mathsf{Mon}_f$ are local.
$(\mathsf{Leq}(f, g)) \quad \forall x_1, \ldots, x_n (\bigwedge_{i=1}^{n} x_i \leq y_i \rightarrow f(x_1, \ldots, x_n) \leq g(y_1, \ldots, y_n))$

Theorem 12 (Comparisons of monotone functions) *Extensions of any subtheory of a theory in Ord whose models are complete lattices w.r.t. the ordering $\leq$, with functions satisfying $\mathsf{Leq}(f, g) \wedge \mathsf{Mon}_f \wedge \mathsf{Mon}_g$ are local.*

Theorem 13 (Semi-Galois connections [16]) *The following are local:*

(1) Extensions of any totally-ordered theory in Ord with functions satisfying the axioms $\mathsf{SGc}(f, g_1, \ldots, g_n) \wedge \mathsf{Mon}(f, g_1, \ldots, g_n)$.
$(\mathsf{SGc}(f, g_1, \ldots, g_n)) \quad \forall x_1, \ldots, x_n, x (\bigwedge_{i=1}^{n} x_i \leq g_i(x) \rightarrow f(x_1, \ldots, x_n) \leq x)$

(2) Extensions of theories in Ord with functions satisfying $\mathsf{SGc}(f, g_1) \wedge \mathsf{Mon}(f, g_1)$.

8.1 Example: Water tank controller (variant 2)

An alternative axiomatization of the desired relationship between the input and the output function (for the sake of simplicity, we here ignore time) is presented below: We assume that $\mathsf{in}, \mathsf{out}$ are monotone, satisfy $\forall L(\mathsf{in}(L) > L)$ and $\forall L(\mathsf{out}(L) < L)$, and

$$\forall L, L' \;\; (L' \leq \mathsf{out}(L) \rightarrow \mathsf{in}(L') \leq L),$$

By results in Theorem 13, these axioms again define a local theory extension.

9 Local combinations of theories

In [17] we proved the following results on combinations of local theory extensions:

Theorem 14 (Combinations of local theory extensions [17]) *Let $\mathcal{T}_0$ be a first-order theory with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$ and $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}_1$ and $\mathcal{T}_2 = \mathcal{T}_0 \cup \mathcal{K}_2$ two extensions of $\mathcal{T}_0$ with signatures $\Pi_1 = (\Sigma_0 \cup \Sigma_1, \mathsf{Pred})$ and $\Pi_2 = (\Sigma_0 \cup \Sigma_2, \mathsf{Pred})$, respectively, where $\Sigma_1 \cap \Sigma_2 = \emptyset$.*

(1) Assume that both extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1$ and $\mathcal{T}_0 \subseteq \mathcal{T}_2$ satisfy condition $(\mathsf{Comp_w})$. 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, in $\mathcal{K}_i$ all terms starting with a function symbol in Σ_i are flat and linear, for $i = 1, 2$, then the extension is local.

(2) Assume that (i) $\mathcal{T}_0 \subseteq \mathcal{T}_1$ satisfies condition $(\mathsf{Comp_w})$; (ii) $\mathcal{T}_0 \subseteq \mathcal{T}_2$ satisfies $(\mathsf{Emb_w})$; (iii) $\mathcal{K}_1$ is a set of Σ_1-flat clauses in which all variables occur below a function symbol in Σ_1. Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ satisfies condition $(\mathsf{Emb_w})$. If, additionally, in $\mathcal{K}_i$ all terms starting with a function symbol in Σ_i are flat and linear, for $i = 1, 2$, then the extension is local.

(3) Assume that (i) both extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1$ and $\mathcal{T}_0 \subseteq \mathcal{T}_2$ satisfy condition $(\mathsf{Emb_w})$; (ii) $\mathcal{K}_1$ and $\mathcal{K}_2$ are sets of Σ_1-flat clauses in which every variable occurs below some extension function; (iii) the class of models of $\mathcal{T}_0$ is closed under directed limits (or, equivalently, $\mathcal{T}_0$ is a $\forall\exists$ theory). Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ satisfies condition $(\mathsf{Emb_w})$. If, additionally, in $\mathcal{K}_i$ all terms starting with a function symbol in Σ_i are flat and linear, for $i = 1, 2$, then the extension is local.

Theorem 14 allows us to combine several of the extensions above and still obtain local extensions of the base theory. For instance, we can directly infer that the following combinations of theories (seen as extensions of a first-order theory $\mathcal{T}_0$) satisfy condition $(\mathsf{Comp_w})$:

Example 3 *The combination of $\mathcal{T}_0 \cup \mathsf{Free}(\Sigma_1)$ and $\mathcal{T}_0 \cup \mathsf{Sel}_c$ if $\mathcal{T}_0$ is a theory and $c \in \Sigma_0$ is injective in $\mathcal{T}_0$; the combination of $\mathbb{R} \cup \mathsf{Free}(\Sigma_1)$ and $\mathbb{R} \cup \mathsf{Mon}(f)$, where $f \notin \Sigma_1$.*

The theory extension $\mathcal{T}_0 \subseteq (\mathcal{L} \cup \mathsf{Free}(\Sigma_1)) \cup (\mathcal{T}_0 \cup \mathsf{Mon}(\Sigma_2))$ – where $\Sigma_1 \cap \Sigma_2 = \emptyset$, and $\mathcal{T}_0$ is a theory in Ord (e.g., the theory of lattices or of semilattices) – satisfies condition $(\mathsf{Emb_w})$, and is therefore local.

Similar results hold for combinations of extensions of a base theory with (piece-wise) boundedness conditions for various function symbols in a set Σ_1 and of extensions of the same base theory with piece-wise monotone functions.

10 Interpolation and abstraction refinement

In verification it is often necessary to automatically generate invariants, or to prove safety for all runs – for this one may need to use a goal-directed overapproximation for achieving faster termination, or to use abstraction (and possibly refine it for ruling out spurious counterexamples). An idea which proved very useful is to use ground interpolants for abstraction refinement [10,11,12,18].

> A theory $\mathcal{T}$ has *interpolation* if, for all formulae ϕ and ψ in the signature of $\mathcal{T}$, if $\phi \models_{\mathcal{T}} \psi$ then there exists a formula I containing only symbols which occur in both ϕ and ψ such that $\phi \models_{\mathcal{T}} I$ and $I \models_{\mathcal{T}} \psi$.

First order logic has interpolation [4], but the interpolants may contain (alternations of) quantifiers even for very simple formulae ϕ and ψ. It often is important to identify situations in which ground clauses have ground interpolants.

> A theory $\mathcal{T}$ has the *ground interpolation property* if for all ground clauses $A(\overline{c}, \overline{d})$ and $B(\overline{c}, \overline{e})$, if $A(\overline{c}, \overline{d}) \wedge B(\overline{c}, \overline{e}) \models_{\mathcal{T}} \bot$ then there exists a ground formula $I(\overline{c})$, containing only the constants $\overline{c}$ occurring both in A and B, such that $A(\overline{c}, \overline{d}) \models_{\mathcal{T}} I(\overline{c})$ and $B(\overline{c}, \overline{e}) \wedge I(\overline{c}) \models_{\mathcal{T}} \bot$.

It is sufficient to know how to compute interpolants for sets of unit clauses, i.e. conjunctions of ground literals: if interpolants for conjunctions of ground literals can be obtained, then also interpolants for conjunctions of arbitrary clauses can be constructed by using standard methods[2], discussed e.g. in [11] or [18].
In [16] we identify a class of theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1$ for which interpolants can be computed hierarchically using a procedure for generating interpolants in the base theory $\mathcal{T}_0$. This allows to exploit specific properties of $\mathcal{T}_0$ for obtaining simple interpolants in $\mathcal{T}_1$. We make the following assumptions[3] about $\mathcal{T}_0$ and $\mathcal{T}_1$:

Assumption 1: $\mathcal{T}_0$ is *convex* w.r.t. the set Pred of all predicates (including equality $\approx$), i.e., for all conjunctions Γ of ground atoms, relations $R_1, \ldots, R_m \in \mathsf{Pred}$ and ground tuples of corresponding arity $\overline{t}_1, \ldots, \overline{t}_n$, if $\Gamma \models_{\mathcal{T}_0} \bigvee_{i=1}^{m} R_i(\overline{t}_i)$ then there exists $j \in \{1, \ldots, m\}$ such that $\Gamma \models_{\mathcal{T}_0} R_j(\overline{t}_j)$.

Assumption 2: $\mathcal{T}_0$ is *P-interpolating*, i.e. for all conjunctions A and B of ground literals, all binary predicates $R \in P$ and all constants a and b such that a occurs in A and b occurs in B (or vice-versa), if $A \wedge B \models_{\mathcal{T}_0} aRb$ then there exists a term t containing only constants common to A and B with $A \wedge B \models_{\mathcal{T}_0} aRt \wedge tRb$.

Assumption 3: $\mathcal{T}_0$ has ground interpolation.

[2] E.g. in a DPLL-style procedure partial interpolants are generated for the unsatisfiable branches and then recombined using ideas of Pudlák.

[3] Examples of theories which have these properties are provided in [16].

Assumption 4: $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$, where $\mathcal{K}$ consists of the combinations of clauses:

$$\begin{cases} x_1\, R_1\, s_1 \wedge \cdots \wedge x_n\, R_n\, s_n \rightarrow f(x_1, \ldots, x_n)\, R\, g(y_1, \ldots, y_n) \\ x_1\, R_1\, y_1 \wedge \cdots \wedge x_n\, R_n\, y_n \rightarrow f(x_1, \ldots, x_n)\, R\, f(y_1, \ldots, y_n) \end{cases} \tag{5}$$

where $n \geq 1$, $x_1, \ldots, x_n$ are variables, $R_1, \ldots, R_n, R$ are binary relations with $R_1, \ldots, R_n \in P$ and R transitive, and each s_i is either a variable among the arguments of g, or a term of the form $f_i(z_1, \ldots, z_k)$, where $f_i \in \Sigma_1$ and all the arguments of f_i are variables occurring among the arguments of g.

Theorem 15 *[16] Assume that the theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ satisfies the assumptions above. Then ground interpolants for $\mathcal{T}_1$ exist and can be computed hierarchically.*

As an immediate consequence of the results in [16], the following theory extensions have ground interpolation, and interpolants can be computed hierarchically.

(a) Extensions with free function symbols of any of the base theories: $\mathcal{E}q$ (pure equality), $\mathcal{P}$ (posets), $\mathsf{LI}(\mathbb{Q}), \mathsf{LI}(\mathbb{R})$ (linear rational, resp. real arithmetic), $\mathcal{S}$ (semilattices), $\mathcal{DL}$ (lattices), $\mathcal{B}$ Boolean algebras.
(b) Extensions with monotone functions of any of the base theories: $\mathcal{P}$ (posets), $\mathcal{S}$ (semilattices), $\mathcal{DL}$ (lattices), $\mathcal{B}$ Boolean algebras.
(c) Extensions of any of the base theories in (b) with $\mathsf{Leq}(f, g) \wedge \mathsf{Mon}_f$.
(d) Extensions of any of the base theories in (b) with $\mathsf{SGc}(\mathsf{f}, \mathsf{g_1}) \wedge \mathsf{Mon}(f, g_1)$.
(e) Extensions of any of the base theories in (a) with Bound^t_f or GBound^t_f (where t is a term and ϕ a set of literals in the base theory).
(f) Extensions of any base theory in (b) with $\mathsf{Mon}_f \wedge \mathsf{Bound}^t_f$, if t is monotone.

Example: Consider the first variant of the water tank example presented in Section 4.1. Since the extension of linear real arithmetic with functions satisfying axioms (1)–(3) satisfies Assumptions 1–4, we can generate interpolants in a hierarchic way for such extensions. This opens possibilities for abstraction refinement based model checking in this particular example and in similar situations.

Limitations. Because of Assumption 1 and of the fact that neither real linear arithmetic, nor integer arithmetic, nor the theory of total orderings is convex w.r.t. $\leq$, we cannot apply the method above for obtaining interpolants in a hierarchical way in the presence of monotone functions over these domains. The method can be applied in the presence of monotone functions in the theory of partial orders, semilattices, (distributive) lattices, or Boolean algebras.

11 Conclusions

We identified several examples from verification where local theory extensions occur in a natural way and where, therefore, hierarchical reasoning in local theory extensions can be used successfully for proving certain invariants and in bounded model checking. In addition, the notion of locality allows to use general theoretical arguments for possibilities of limiting the search to neighborhoods of

counterexamples without losing completeness. We also briefly discussed implications of efficient interpolation to verification by abstraction refinement.

In all the examples considered in this paper, locality allows to pass whithout loss of completeness, from testing satisfiability of *quantified formulae* to testing satisfiability for suitably constructed *ground formulae*, problem which is much easier. We illustrated this reduction by examples in verification. The applicability of the method is however general: the challenge is, at the moment, to recognize classes of local theories occurring in various areas of application.

The method can, in fact, be applied for arbitrary theories, even if locality was not proved. When applied for arbitrary theory extensions, the method is sound (we test satisfiability for a smaller subset of the instances of the extension axioms; if unsatisfiability is proved this way then the initial set of clauses was unsatisfiable w.r.t. the extended theory) but may be incomplete. In such situations, it can be seen as a type of 'abstraction'; ideas from instantiation-based theorem proving may be used in addition to add more instances of the extension clauses and thus refine the abstraction. (The method is guaranteed to be sound and complete and to terminate for local extensions, provided decision procedures for the base theory exist.)

Acknowledgements. I thank Uwe Waldmann for bringing the contents of [13] to my attention, and to Andreas Podelski for bringing [2] to my attention. I thank Swen Jacobs, together with whom I analyzed the train example described in Section 7.1 (cf. also [8]), and Andrey Rybalchenko for interesting discussions on abstraction refinement model checking. I thank the other participants in the AVACS project for providing examples and feedback.

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. D.A. Basin and H. Ganzinger. Complexity analysis based on ordered resolution. In *Proc. 11th IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 456–465. IEEE Computer Society Press, 1996.
2. A.R. Bradley, Z. Manna, and H.B. Sipma. What's decidable about arrays? In E.A. Emerson and K.S. Namjoshi, editors, *Verification, Model-Checking, and Abstract-Interpretation, 7th Int. Conf. (VMCAI 2006)*, LNCS 3855, pages 427–442. Springer, 2006.
3. P. Burmeister. *A Model Theoretic Oriented Approach to Partial Algebras: Introduction to Theory and Application of Partial Algebras, Part I*, volume 31 of *Mathematical Research*. Akademie-Verlag, Berlin, 1986.
4. W. Craig. Linear reasoning. A new form of the Herbrand-Gentzen theorem. *J. Symb. Log.*, 22(3):250–268, 1957.
5. H. Ganzinger. 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)*, pages 81–92. IEEE Computer Society Press, 2001.

6. H. Ganzinger, V. Sofronie-Stokkermans, and U. Waldmann. Modular proof systems for partial functions with Evans equality. *Information and Computation*. To appear.
7. R. Givan and D. McAllester. New results on local inference relations. In *Principles of Knowledge Representation and reasoning: Proceedings of the Third International Conference (KR'92)*, pages 403–412. Morgan Kaufmann Press, 1992.
8. S. Jacobs and V. Sofronie-Stokkermans. Applications of hierarchical reasoning in the verification of complex systems. Proceedings of the Fourth International Workshop on Pragmatics of Decision Procedures in Automated Reasoning (PDPAR'06), To appear., 2006.
9. D. McAllester. Automatic recognition of tractability in inference relations. *Journal of the Association for Computing Machinery*, 40(2):284–303, 1993.
10. K.L. McMillan. Interpolation and SAT-based model checking. In *CAV'2003: Computer Aided Verification, LNCS 2725*, pages 1–13. Springer, 2003.
11. K.L. McMillan. An interpolating theorem prover. In *TACAS'2004: Tools and Algorithms for the Construction and Analysis of Systems, LNCS 2988*, pages 16–30. Springer, 2004.
12. K.L. McMillan. Applications of Craig interpolants in model checking. In *TACAS'2005: Tools and Algorithms for the Construction and Analysis of Systems, LNCS 3440*, pages 1–12. Springer, 2005.
13. S. McPeak and G.C. Necula. Data structure specifications via local equality axioms. In K. Etessami and S.K. Rajamani, editors, *Computer Aided Verification, 17th International Conference, CAV 2005*, LNCS 3576, pages 476–490, 2005.
14. T. Skolem. Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit und Beweisbarkeit mathematischen Sätze nebst einem Theoreme über dichte Mengen. *Skrifter utgit av Videnskabsselskapet i Kristiania, I. Matematisk-naturvidenskabelig klasse, 4*, pages 1–36, 1920. Reprinted as 1920a in Skolem 1970, pp. 103-136.
15. V. Sofronie-Stokkermans. Hierarchic reasoning in local theory extensions. In R. Nieuwenhuis, editor, *20th International Conference on Automated Deduction (CADE-20), LNAI 3632*, pages 219–234. Springer, 2005.
16. V. Sofronie-Stokkermans. Interpolation in local theory extensions. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006)*, LNAI 4130, pages 235–250. Springer, 2006.
17. V. Sofronie-Stokkermans. On combinations of local theory extensions. Submitted, 2006.
18. G. Yorsh and M. Musuvathi. A combination method for generating interpolants. In R. Nieuwenhuis, editor, *20th International Conference on Automated Deduction (CADE-20), LNAI 3632*, pages 353–368. Springer, 2005.