Incremental Instance Generation in Local Reasoning

Swen Jacobs

Max-Planck-Institut für Informatik, Campus E1.4, Saarbrücken, Germany
e-mail: sjacobs@mpi-inf.mpg.de

Abstract. Many verification approaches use SMT solvers in some form, and are limited by their incomplete handling of quantified formulas. Local reasoning allows to handle SMT problems involving a certain class of universally quantified formulas in a complete way by instantiation to a finite set of ground formulas. We present a method to generate these instances incrementally, in order to provide a more efficient way of solving these satisfiability problems. The incremental instantiation is guided semantically, inspired by the instance generation approach to first-order theorem proving. Our method is sound and complete, and terminates on both satisfiable and unsatisfiable input after generating a subset of the instances needed in standard local reasoning. Experimental results show that for a large class of examples the incremental approach is substantially more efficient than eager generation of all instances.

1 Introduction

SMT solvers are widely used tools in software verification today. One of the big challenges for current SMT solvers [11] is to handle quantified formulas, e.g. in order to make statements over potentially unbounded data structures. Current approaches to solve problems involving quantifiers [6] are usually refinements of the heuristical instantiation introduced with the Simplify prover [3]. A drawback of these approaches is that completeness is sacrificed: even for unsatisfiable problems, termination cannot be guaranteed.

Local reasoning [12] provides a decision procedure for SMT problems containing quantified formulas, provided they satisfy a locality property with respect to the base theory. For example, axioms specifying monotonicity or injectivity fall into this class, allowing us to specify sortedness or uniqueness of elements in an unbounded array. In local reasoning, these axioms are instantiated to a finite set of ground instances, which grows with the number of ground terms appearing in the given problem. For large problems, generating all instances at once may become inefficient, especially in verification problems where unsatisfiability often only depends on a relatively small part of the formula.

We introduce a method to generate the needed instances incrementally, in a systematical way that allows termination without generating the full set of instances in both the satisfiable and unsatisfiable case, while preserving completeness. The idea is based on the instantiation-based theorem proving methods introduced by Ganzinger and Korovin [5]: we keep a candidate model which satisfies the ground part of the current problem, and only add instances that evolve this candidate model, either strengthening or refuting it.

This paper is organized as follows: in Section 2 we introduce the logical terminology used throughout this paper, as well as the basics of local reasoning. In Section 3, we introduce our approach, together with proofs of correctness. This approach is refined in Section 4, resulting in broader applicability. In Section 5, we evaluate experimental results, comparing different approaches to local reasoning. We conclude the paper with remarks on related and ongoing work.

2 Local Reasoning and Terminology

This section introduces necessary terminology and gives a short introduction to local reasoning. For a more detailed description, including theoretical background, we refer to Sofronie-Stokkermans et al. [12, 9, 8].

2.1 Logical Prerequisites

We use the usual symbols, notation and terminology of sorted first-order logic with standard definitions.

Formulas and instances. We use x, y to denote variables, a, b, c, d for constants and f as non-constant function symbol. L will denote literals, C and D clauses, and F formulas. We consider all formulas to be in clausal normal form, represented as a set of clauses, where all variables in a clause are implicitly universally quantified. Clauses are called *axioms* if they contain variables, *ground clauses* otherwise. We denote sets of axioms by $\mathcal{K}$, sets of ground clauses by G.

If F is a formula and σ a substitution, then $F\sigma$ is an *instance* of F. If F_2 is an instance of F_1, then F_1 is a *generalization* of F_2. A *variable renaming* is an injective substitution mapping variables to variables. Two formulas F_1 and F_2 are *variants* of each other if there is a variable renaming σ such that $F_1\sigma = F_2$. For a formula F, let $\mathsf{st}(F)$ be the set of ground subterms appearing in F.

Theories and models. Consider a signature $\Pi = (S, \Sigma, \mathsf{Pred})$, where S is a set of sorts, Σ is a set of function symbols and Pred a set of predicate symbols (with given arities). A *Π-structure* assigns concrete, non-empty sets of elements to all sorts $s \in S$ and concrete valuations to every $f \in \Sigma$ and $P \in \mathsf{Pred}$, according to their arity. A *theory* $\mathcal{T}$ can be defined by a set of axioms or a set of models. A given Π-structure M is a *model* of a theory $\mathcal{T}$ iff every axiom of $\mathcal{T}$ is satisfied by M. If a formula F is true in all models of $\mathcal{T}$, we write $\models_{\mathcal{T}} F$. A formula F_2 is a *consequence* of F_1 (modulo $\mathcal{T}$), written $F_1 \models_{\mathcal{T}} F_2$, if F_2 is true in every model of $\mathcal{T}$ which also satisfies F_1. If no model of $\mathcal{T}$ satisfies F, we write $F \models_{\mathcal{T}} \Box$, where $\Box$ represents the empty clause. Validity (modulo $\mathcal{T}$) of a formula F in a Π-structure M is depicted by $M \models_{\mathcal{T}} F$.

The *Herbrand base* $\mathcal{H}$ of Π is the set of all ground atoms over the signature. A *Herbrand interpretation* of Π is a set of literals $\mathcal{I}$ which contains for every atom $A \in \mathcal{H}$ either A or $\neg A$. A set of literals $\mathcal{L} \subseteq \mathcal{I}$ is a *partial Herbrand interpretation* of Π. A (partial) Herbrand interpretation $\mathcal{L}$ is a (partial) *Herbrand model* of a set of clauses F if $\mathcal{L}$ contains at least one of the literals from each clause in F.

2.2 Reasoning in Local Theory Extensions

Hierarchic reasoning in local theory extensions, or in short: *local reasoning*, has been introduced by Sofronie-Stokkermans [12]. The approach is based on earlier results by Givan and McAllester [7] and Ganzinger [4]. In this section we present the method, which allows us to solve satisfiability problems in local extensions of decidable background theories.

Local theory extensions. Consider a background theory $\mathcal{T}$ with signature $\Pi_0 = (S, \Sigma_0, \mathsf{Pred})$. In the following, we will allow formulas which are not restricted to this signature, but may contain additional function (and constant) symbols given by a set Σ_1. If F and G are formulas in the signature $\Pi = (S, \Sigma_0 \cup \Sigma_1, \mathsf{Pred})$, we will reason in $\mathcal{T}^{\Sigma_1}$, the extension of $\mathcal{T}$ with free function symbols from Σ_1. Whenever Σ_1 is clear from the context, we will write $F \models_{\mathcal{T}} G$ instead of $F \models_{\mathcal{T}^{\Sigma_1}} G$.

A *theory extension* of a base theory $\mathcal{T}$ is given by a set $\mathcal{K}$ of axioms with a set of additional function symbols Σ_1, and denoted by $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}$. We will use $\mathcal{T} \cup \mathcal{K}$ to denote the *extended theory*, which treats symbols from Σ_1 according to the axioms in $\mathcal{K}$. For non-constant function symbols $f \in \Sigma_1$, f is called an *extension symbol*, terms $f(t)$ are *extension terms*. Note that $F \models_{\mathcal{T}\cup\mathcal{K}} G$ iff $\mathcal{K} \cup F \models_{\mathcal{T}} G$.

In the following, we only consider theory extensions $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}$ such that every $C \in \mathcal{K}$ is Σ_1*-linear*: all extension terms in C which contain the same variable are syntactically equal, and no extension term in C contains two occurrences of the same variable.[1] Furthermore, we require that all variables in a clause $C \in \mathcal{K}$ have at least one occurrence in an extension term.

For a set G of ground clauses (in signature Π, but possibly with additional constant symbols), and a Σ_1-linear set of axioms $\mathcal{K}$, let

$$\mathcal{K}[G] = \{C\sigma \mid C \in \mathcal{K} \text{ and } f(t)\sigma \in \mathsf{st}(G) \text{ for each extension subterm } f(t) \text{ of } C\}.$$

For the fragment we consider, an extension $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}$ is *local* if it satisfies condition (Loc):

(Loc) For every set G of ground clauses, $G \models_{\mathcal{T}\cup\mathcal{K}} \square \Leftrightarrow \mathcal{K}[G] \cup G \models_{\mathcal{T}} \square$,

where by our convention from above $\models_{\mathcal{T}}$ means $\models_{\mathcal{T}^{\Sigma_1}}$, with Σ_1 containing the additional function and constant symbols from $\mathcal{K}$ and G.

Examples. In previous papers [12, 9, 8], several theory extensions have been proved to be local.[2] We want to mention a few of them which satisfy our additional restriction and are interesting in a verification context:

- *monotone functions* over several theories with a partial or total ordering,
- *strictly monotone functions*, if additionally the codomain is densely ordered,
- *injective functions*, if domain and codomain allow them,

[1] This restriction is needed to prove locality of an extension and is also used throughout previous papers [12, 8].

[2] The proof is done by hand and requires to show that certain partial models can be embedded into total models.

- sets of *guarded boundedness* conditions of the form

  $$\phi_i(\overline{x}) \to s_i(\overline{x}) \leq f(\overline{x}) \leq t_i(\overline{x}),$$

  where the ϕ_i are mutually exclusive formulas in the base theory and the s_i, t_i are terms in the base theory such that $\phi_i(\overline{x}) \to s_i(\overline{x}) \leq t_i(\overline{x})$ for all $i, \overline{x}$ (where $\overline{x}$ is the vector of all variables occurring in the given clause).

Hierarchic reasoning in local theory extensions. Let $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}$ be a local theory extension. Satisfiability of a set G of ground clauses modulo the extended theory $\mathcal{T} \cup \mathcal{K}$ can be checked in the following way (for details and the general approach cf. [12]):

By the locality condition, $G \models_{\mathcal{T}\cup\mathcal{K}} \Box$ iff $\mathcal{K}[G] \cup G \models_{\mathcal{T}} \Box$. Note that because in every clause in $\mathcal{K}$, all variables occur in extension terms, $\mathcal{K}[G]$ is a set of ground clauses. As everything is ground, extension functions can now be treated as free functions. Thus, we can check $\mathcal{K}[G] \cup G \models_{\mathcal{T}} \Box$ with any SMT solver which supports the base theory $\mathcal{T}$ plus uninterpreted functions (EUF).

Example 1. Suppose $\mathcal{T}$ is any theory with a partial ordering (like real or integer arithmetic) and we consider its extension with a monotone function f, i.e.

$$\mathcal{K} = \{x \leq y \to f(x) \leq f(y)\}.$$

We want to check whether the following ground goal G is satisfiable with respect to the extended theory $\mathcal{T} \cup \mathcal{K}$:

$$G = \{\ a \leq b,\ \neg(f(a) \leq f(b))\ \}.$$

By locality of the extension $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}$, we know that

$$G \models_{\mathcal{T}\cup\mathcal{K}} \Box \ \Leftrightarrow\ \mathcal{K}[G] \cup G \models_{\mathcal{T}} \Box,$$

$$\begin{aligned}\text{where } \mathcal{K}[G] = \{\ & a \leq a \to f(a) \leq f(a), \quad a \leq b \to f(a) \leq f(b)\\ & b \leq a \to f(b) \leq f(a), \quad b \leq b \to f(b) \leq f(b)\ \}.\end{aligned}$$

$\mathcal{K}[G] \cup G \models_{\mathcal{T}} \Box$ can be checked with any prover for SMT(EUF $\cup\, \mathcal{T}$).

Chains of extensions. If we have an extension $\mathcal{K}$ which is not local wrt. the base theory, we may still be able to treat it within the local reasoning framework:

Suppose we have the case that $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}_1 \cup \mathcal{K}_2$ is not a local extension, but both $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}_1$ and $\mathcal{T} \cup \mathcal{K}_1 \subseteq \mathcal{T} \cup \mathcal{K}_1 \cup \mathcal{K}_2$ are local extensions. This means we can extend the base theory $\mathcal{T}$ to the extended theory $\mathcal{T} \cup \mathcal{K}_1 \cup \mathcal{K}_2$ in two steps, and use the local reasoning procedure repeatedly to reduce a ground goal from $\mathcal{T} \cup \mathcal{K}_1 \cup \mathcal{K}_2$ to $\mathcal{T} \cup \mathcal{K}_1$ and then to $\mathcal{T}$. This approach can be generalized to an arbitrary number of steps. We call such a repeated extension a *chain of extensions*, and will have a more detailed look at reasoning in chains of extensions in Section 4.

3 An Incremental Approach to Local Reasoning

In the last section we have introduced *local reasoning*, which provides a decision procedure for satisfiability problems in local extensions of theories. Given unlimited resources, the procedure is guaranteed to terminate. As in practice neither time nor space are unlimited, we are looking for more efficient methods for local reasoning. In Section 3.1, we introduce *local instance generation*, a method which generates instances of local axioms incrementally, in a semantically guided way inspired by the instance generation approach of Ganzinger and Korovin [5]. In Section 3.2, we give examples how local reasoning can benefit from this approach, and in Section 3.3 we prove correctness of the method.

3.1 Local Instance Generation (LIG)

We present the procedure *LIG*, which solves the satisfiability problem of a ground formula G in an extended theory $\mathcal{T} \cup \mathcal{K}$ by interleaving ground satisfiability checks with instantiation of axioms from $\mathcal{K}$. For ground reasoning we use a blackbox SMT solver, which should allow incremental addition of constraints and must be able to generate models for satisfiable input.

Proof procedure. We keep working sets $\mathcal{K}$ of axioms and G of ground clauses. Define a selection function sel on $\mathcal{K}$ so that for every $C \in \mathcal{K}$, $\mathsf{sel}(C)$ is the literal $L \in C$ which contains the highest number of variables, preferably so that variables occur below extension symbols. This selection is chosen in order to enforce fast instantiation to the ground level, as explained below. Then, the following two steps are repeated until termination:

(1) Ground satisfiability check. We give G to an SMT solver. If $G \models_{\mathcal{T}} \square$, then the procedure terminates and states unsatisfiability of the input. Otherwise, obtain a model M such that $M \models_{\mathcal{T}} G$. From M, compute a partial Herbrand model $\mathcal{M}$ of G.

(2) Instance generation. Instances are generated according to the following inference rule:

$$\mathit{LIG} \qquad \frac{(L_1 \vee D_1) \ \cdots \ (L_n \vee D_n) \ \mid \mathcal{M}}{(L_1 \vee D_1)\sigma \ \cdots \ (L_n \vee D_n)\sigma}$$

where:

(i) the $L_i \vee D_i$ are (variants of) clauses from $\mathcal{K}$ with $\mathsf{sel}(L_i \vee D_i) = L_i$,
(ii) σ is a substitution such that all $L_i\sigma$ are ground literals,
(iii) all clauses $(L_i \vee D_i)\sigma$ are generalizations of clauses in $\mathcal{K}[\mathcal{M}]$, and
(iv) $\{L_i\sigma\}_{1 \leq i \leq n} \cup \mathcal{M} \models_{\mathcal{T}} \square$

Note that several variants of a clause $C \in \mathcal{K}$ may be instantiated in one inference. Ground clauses $(L_i \vee D_i)\sigma$ are added to G, axioms (if any) are added to $\mathcal{K}$. If all inferences based on a given selection sel and model $\mathcal{M}$ only produce variants of clauses in $\mathcal{K}$ and G, we call the pair $(\mathcal{K}, G)$ *saturated* under *LIG*. If $(\mathcal{K}, G)$ is saturated under *LIG*, the procedure terminates and states satisfiability of the input. Otherwise, define sel on new axioms as described above and return to **(1)**.

Finding substitutions. Side conditions (ii) and (iii) allow to restrict the search for substitutions to a finite set. The following lemma shows how side condition (iii) can be ensured.

Lemma 1. *Let* $\{(L_1 \vee D_1), \dots, (L_n \vee D_n)\}$ *and* $\mathcal{M}$ *be premises of an LIG-inference. For* $1 \leq i \leq n$*, let* $\mathsf{sel}(L_i \vee D_i) = L_i$*, and let* $\mathcal{L}_i$ *be a set of literals such that* $\mathcal{L}_i \subseteq (L_i \vee D_i)$*, and all variables from* $(L_i \vee D_i)$ *appear in extension terms in* $\mathcal{L}_i$*. Then, for any substitution* σ *which satisfies condition* (ii) *of the LIG inference rule, the following two are equivalent:*

(1) σ *satisfies condition* (iii) *of the LIG inference rule.*
(2) σ *is such that all ground extension terms in* $\mathcal{L}_i\sigma$ *are in* $\mathsf{st}(\mathcal{M})$*.*

Proof: Suppose (1) holds. Then by (iii), all $(L_i \vee D_i)\sigma$ are generalizations of clauses in $\mathcal{K}[\mathcal{M}]$, which means they are either in $\mathcal{K}[\mathcal{M}]$ or they can be instantiated to a clause in $\mathcal{K}[\mathcal{M}]$. In any way, by definition of $\mathcal{K}[G]$, they can only contain ground extension terms that are in $\mathsf{st}(\mathcal{M})$. Thus, this holds also for every subset of every $(L_i \vee D_i)\sigma$, implying (2).

Now suppose (2) holds. Because $\mathcal{K}$ is Σ_1-linear, for every $(L_i \vee D_i)$, all extension terms containing the same variable are syntactically equal. As $\mathcal{L}_i$ contains all variables in extension terms, all non-ground extension terms in $(L_i \vee D_i) \setminus \mathcal{L}_i$ must be equal to some term in $\mathcal{L}_i$. As all ground extension terms in $\mathcal{L}_i\sigma$ are in $\mathsf{st}(\mathcal{M})$, so must be all ground extension terms in $(L_i \vee D_i)\sigma$. By definition of $\mathcal{K}[\mathcal{M}]$, every $(L_i \vee D_i)\sigma$ is a generalization of a clause in $\mathcal{K}[\mathcal{M}]$, satisfying (iii).□

The lemma suggests that in order to ensure condition (iii) of the inference rule, we may watch for every clause $C \in \mathcal{K}$ a set of literals $\mathcal{L} \subseteq C$ such that in $\mathcal{L}$, all variables from C occur in extension terms. Together with condition (ii), we then only need to consider substitutions σ mapping variables of $\mathsf{sel}(C)$ to ground terms such that all ground extension terms in $\mathcal{L}\sigma$ are in $\mathsf{st}(\mathcal{M})$.

Special cases. The following special cases make the search for a substitution easier, and justify our definition of sel on $\mathcal{K}$. For any $C \in \mathcal{K}$:

- if all variables from C occur in $\mathsf{sel}(C)$, then any conclusion $C\sigma$ of an *LIG*-inference will be ground.
- if furthermore all variables appear below extension terms in $\mathsf{sel}(C)$, then it is sufficient to consider $\mathcal{L} = \{\mathsf{sel}(C)\}$.

Unsatisfiable cores. In order to minimize the number of generated instances, one can eliminate premises (and conclusions) of an inference which are not needed to satisfy condition (iv) of the *LIG* rule. If $\{L_i\sigma\}_{1 \leq i \leq n} \cup \mathcal{M} \models_{\mathcal{T}} \Box$, then a decision procedure for $\mathcal{T}$ can be used to compute an *unsatisfiable core* of $\{L_i\sigma\}_{1 \leq i \leq n}$ with respect to $\mathcal{M}$, i.e. a small subset $\mathcal{L} \subseteq \{L_i\sigma\}_{1 \leq i \leq n}$ such that $\mathcal{L} \cup \mathcal{M} \models_{\mathcal{T}} \Box$. If we drop premises and conclusions which do not contain a literal $L_i \in \mathcal{L}$, we still have a valid *LIG* inference.

3.2 Examples: Behaviour of *LIG*

In the following examples, we compare the different approaches to local reasoning. We will call the standard method from Section 2.2 *eager instantiation*, the incremental method without unsatisfiable cores LIG, and with unsatisfiable cores LIG_{uc}. For all examples, consider the same background theory and extension as in Example 1, i.e. $\mathcal{T}$ has a partial ordering and $\mathcal{K} = \{\ x \leq y \rightarrow f(x) \leq f(y)\ \}$.

Example 2. Let $G = \{\ a \leq b,\ \neg(f(a) \leq f(b))\ \}$. Then, as seen in Example 1, eager instantiation generates a set of clauses $\mathcal{K}[G]$, with $|\mathcal{K}[G]| = 4$.

When using LIG, we have $\mathsf{sel}(\mathcal{K}) = \{\ f(x) \leq f(y)\ \}$ and $\mathcal{M} = G$ (as we only have unit clauses in G). To satisfy the side conditions of the LIG rule, we search for a substitution instantiating one or several variants of $f(x) \leq f(y)$ such that the resulting ground instances are unsatisfiable together with $\mathcal{M}$, and the resulting extension terms (starting with f) already appear in $\mathcal{M}$.

In the worst case LIG considers the whole set $\mathcal{L} = \mathsf{sel}(\mathcal{K})[G]$, containing 4 instances of the literal $f(x) \leq f(y)$. As $\mathcal{L} \cup \mathcal{M}$ is unsatisfiable, LIG produces exactly the same instances as before.

LIG_{uc} searches for an unsatisfiable core in $\mathcal{L}$ and will probably find out that already $\{\ f(a) \leq f(b)\ \} \cup \mathcal{M}$ is unsatisfiable. Thus, it will produce $a \leq b \rightarrow f(a) \leq f(b)$ with its first inference and add it to G, which makes G unsatisfiable.

We can see that because of the missing boolean structure in G, the LIG rule does not reduce the number of instances. However, the additional effort of computing an unsatisfiable core pays off: we produce exactly the one instance which is needed to show unsatisfiability of $\mathcal{K}[G] \cup G$.

The following example shows that the effect of the incremental methods grows with the amount of boolean structure in G and of information that does not contribute to the proof of unsatisfiability.

Example 3. Let G consist of the ground clauses

$$\begin{array}{rcl}
(C_i) & a_i \leq b_i \ \vee\ c_i \leq d_i, & \text{for } 1 \leq i \leq n \\
(D_i) & f(a_i) \leq f(b_i) \ \vee\ f(c_i) \leq f(d_i), & \text{for } 1 \leq i \leq n \\
(E) & \neg(a_1 \leq b_1) \ \vee\ \neg(f(a_1) \leq f(b_1)) & \\
(F) & \neg(c_1 \leq d_1) \ \vee\ \neg(f(c_1) \leq f(d_1)) &
\end{array}$$

For any n, this set of clauses is unsatisfiable, because the set $\{C_1, D_1, E, F\}$ is already unsatisfiable in $\mathcal{T} \cup \mathcal{K}$. The instances of the monotonicity axiom needed to prove this are $a_1 \leq b_1 \rightarrow f(a_1) \leq f(b_1)$ and $c_1 \leq d_1 \rightarrow f(c_1) \leq f(d_1)$. Let us compare how the different approaches to local reasoning treat this problem:

The eager approach generates $\mathcal{K}[G]$, which consists of all instances of the monotonicity axiom with all combinations of substituting x and y with the constants a_i, b_i, c_i, d_i, i.e. $|\mathcal{K}[G]| = (4n)^2$, and then checks satisfiability.

In LIG, $\mathsf{sel}(\mathcal{K})$ is as above, and $\mathcal{M}$ may contain an arbitrary literal out of each clause, except for C_1, D_1, E and F. The dependencies between these imply that either $f(a_1) \leq f(b_1)$ and $\neg(f(c_1) \leq f(d_1))$ are in $\mathcal{M}$, or $\neg(f(a_1) \leq f(b_1))$ and $f(c_1) \leq f(d_1)$. From the other D_i, we have $n-1$ different literals in $\mathcal{M}$, each with 2 extension terms that do not appear elsewhere. Thus, $\mathsf{st}(\mathcal{M})$ contains $2n+2$ extension terms and in search for a substitution we generate a set $\mathcal{L}$ containing $(2n+2)^2$ instances of the literal $f(x) \leq f(y)$. $\mathcal{L}$ is unsatisfiable and LIG may produce all $(2n+2)^2$ corresponding clause instances. Since $f(a_1), f(b_1), f(c_1), f(d_1)$ are all in $\mathsf{st}(\mathcal{M})$, among these instances are $a_1 \leq b_1 \rightarrow f(a_1) \leq f(b_1)$ and $c_1 \leq d_1 \rightarrow f(c_1) \leq f(d_1)$, which makes G unsatisfiable after the inference.

In LIG_{uc}, we compute an unsatisfiable core of $\mathcal{L}$ with respect to $\mathcal{M}$. As $\mathcal{M}$ must contain either $\neg(f(a_1) \leq f(b_1))$ or $\neg(f(c_1) \leq f(d_1))$, we can find that either $f(a_1) \leq f(b_1)$ or $f(c_1) \leq f(d_1)$ are unsatisfiable cores. After adding the corresponding instance of the axiom, the problem is still satisfiable, and the new model must contain exactly those literals from C_1, D_1, E and F which were not in $\mathcal{M}$ before. Assume that the other literals in $\mathcal{M}$ do not change unnecessarily. Then, $\mathsf{st}(\mathcal{M})$ and our set $\mathcal{L}$ of literals will be the same as in the first inference. Now, the unsatisfiable core consists again of one of the two literals $f(a_1) \leq f(b_1)$ or $f(c_1) \leq f(d_1)$, namely the one that was not chosen before. The generated instance is exactly the one which makes G unsatisfiable.

Thus, LIG proves $G \models_{\mathcal{T} \cup \mathcal{K}} \square$ by generating $(2n+2)^2$ instances in one step, and LIG_{uc} needs to generate only 2 instances in 2 steps. Recall that in contrast to this, eager instantiation generates $(4n)^2$ instances.

Finally, we give an example how LIG and LIG_{uc} can not only prove unsatisfiability, but also generate models for satisfiable problems incrementally.

Example 4. Consider a modified version of Example 3: let $\mathcal{K}$ be as before, and remove clause F from G, which makes the problem satisfiable for all n. We will show how the different approaches generate a model for $|\mathcal{K}[G]|$.

In the eager approach, $\mathcal{K}[G]$ is generated completely and the search for a model is left to the SMT solver.

In LIG, the SMT solver generates a model $\mathcal{M}$ for G. We can distinguish two cases: either (i) the second literal of E is not in $\mathcal{M}$, or (ii) it is. In case (i), LIG will not generate any instances, since $\mathcal{M}$ only contains positive literals with extension symbols, i.e. no set of instantiations of $\mathsf{sel}(\mathcal{K}) = \{f(x) \leq f(y)\}$ can be unsatisfiable with respect to $\mathcal{M}$. In case (ii), LIG will find out that $\mathsf{sel}(\mathcal{K})[\mathcal{M}]$ is unsatisfiable with respect to $\mathcal{M}$, and generate $(2n+1)^2$ axiom instances. After this, LIG calls the SMT solver for a new model of $G \cup \mathcal{K}[\mathcal{M}]$. The procedure above is repeated until we get back a model which does not include the second literal of E, or until the set of instances is saturated with respect to the model, i.e. all axiom instances that can be generated have already been generated.

In LIG_{uc}, we can make the same case distinction. In case (i) we detect satisfiability immediately, in case (ii) we will find out that $f(a_1) \leq f(b_1)$ is already unsatisfiable with respect to $\mathcal{M}$, and generate $a_1 \leq b_1 \rightarrow f(a_1) \leq f(b_1)$. After that, the SMT solver is called again, and we can apply the same case distinction for the new model: in case (i), we cannot generate any instances, and in case (ii) the only instance we can generate is $a_1 \leq b_1 \rightarrow f(a_1) \leq f(b_1)$, which we have already generated. Thus, the set of instances is saturated.

We see that in this case the benefit of *LIG* may become smaller than before, since termination depends very much on the models supplied by the SMT solver. For LIG_{uc} however, we can still guarantee termination after two iterations and generation of a single axiom instance.

3.3 Correctness of *LIG*

In the following, we show that *LIG* is sound, complete and terminating. An *LIG-derivation* is a sequence of tuples $(\mathcal{K}^0, G^0) \vdash (\mathcal{K}^1, G^1) \vdash \ldots \vdash (\mathcal{K}^n, G^n)$ such that G^0 is a set of ground clauses, $\mathcal{K}^0$ is a local extension of the background theory $\mathcal{T}$, and for $0 \leq i \leq n-1$, G^i is $\mathcal{T}$-satisfiable and $(\mathcal{K}^{i+1}, G^{i+1})$ is the result of an *LIG*-inference on $(\mathcal{K}^i, G^i)$.

Theorem 1 (Soundness). *Let $(\mathcal{K}^0, G^0) \vdash (\mathcal{K}^1, G^1) \vdash \ldots \vdash (\mathcal{K}^n, G^n)$ be an LIG-derivation. If $G^n \models_{\mathcal{T}} \Box$, then $\mathcal{K}^0 \cup G^0 \models_{\mathcal{T}} \Box$.*

Proof: All elements of G^n are instances of clauses in $\mathcal{K}^0 \cup G^0$. Thus, their unsatisfiability implies unsatisfiability of $\mathcal{K}^0 \cup G^0$. □

For the proof of completeness we need the following definition: Let $\mathcal{F}$ be a set of formulas. F_1 is a *most specific generalization of* F_2 *with respect to* $\mathcal{F}$ if F_1 is a generalization of F_2 and there is no $F_3 \in \mathcal{F}$ which is both a non-variant instance of F_1 and a generalization of F_2.

Theorem 2 (Completeness). *Let $(\mathcal{K}^0, G^0) \vdash (\mathcal{K}^1, G^1) \vdash \ldots \vdash (\mathcal{K}^n, G^n)$ be an LIG-derivation. If G^n is $\mathcal{T}$-satisfiable and $(\mathcal{K}^n, G^n)$ is saturated under LIG with respect to a given selection function* sel *and a partial Herbrand model $\mathcal{M}$, then G^0 is $(\mathcal{T} \cup \mathcal{K}^0)$-satisfiable.*

Proof: Let $\mathcal{K}^0, G^0, \mathcal{K}^n, G^n, \mathsf{sel}$ and $\mathcal{M}$ as defined above. $\mathcal{M}$ is a partial Herbrand model of G^n, where $G^n \subseteq \mathcal{K}^0[G^0] \cup G^0$. We extend $\mathcal{M}$ to a total Herbrand model of $\mathcal{K}^0[G^0] \cup G^0$ in the following way: for every $C_i \in (\mathcal{K}^0[G^0] \cup G^0) \setminus G^n$ we can find a non-ground clause $D_i \in \mathcal{K}^n$ s.t. $D_i\sigma_i = C_i$ and D_i is a most specific generalization of C_i wrt. $\mathcal{K}^n$. For every such C_i, we add $\mathsf{sel}(D_i)\sigma_i$ to $\mathcal{M}$, i.e. the resulting set of literals $\mathcal{M}'$ contains at least one literal from each clause in $\mathcal{K}^0[G^0] \cup G^0$. Furthermore, $\mathcal{M}'$ cannot be contradictory: if this was the case, we would have an *LIG* inference with the D_i and $\mathcal{M}$ as premises, producing the C_i or generalizations thereof. As the D_i were chosen to be most specific generalizations of the C_i wrt. $\mathcal{K}^n$, the produced instances are neither in G^n nor $\mathcal{K}^n$, thus contradicting our assumption that $(\mathcal{K}^n, G^n)$ is saturated under *LIG* wrt. sel and $\mathcal{M}$.[3] As $\mathcal{M}'$ is not contradictory and contains one literal out of each clause in $\mathcal{K}^0[G^0] \cup G^0$, $\mathcal{M}'$ is a partial Herbrand model of $\mathcal{K}^0[G^0] \cup G^0$. By locality, G^0 is $(\mathcal{T} \cup \mathcal{K}^0)$-satisfiable iff $\mathcal{K}^0[G^0] \cup G^0$ is $\mathcal{T}$-satisfiable. □

[3] If unsatisfiable cores are used, the argument is the same except that only a subset of the D_i are used as premises — but since $\mathcal{M}$ was not contradictory before, at least one of the D_i has to be instantiated to a new instance.

Theorem 3 (Termination). *For any input $(\mathcal{K}^0, G^0)$ (where $\mathcal{K}^0$ and G^0 are as defined above), LIG terminates after a finite number of inferences.*

Proof: *LIG*-inferences produce only clauses which are both instances of clauses in $\mathcal{K}$ and generalizations of clauses in $\mathcal{K}[\mathcal{M}]$, where for every sel, $\mathcal{K}[\mathcal{M}] \subseteq \mathcal{K}[G]$. For a given input $(\mathcal{K}, G)$, only finitely many clauses are both instances of clauses in $\mathcal{K}$ and generalizations of clauses in $\mathcal{K}[G]$, and thus there are only finitely many possible *LIG*-inferences. □

4 Chains of Local Theory Extensions

In Section 2.2, we mentioned that local extensions of theories can be serialized, i.e. an extended theory can be extended again. This approach is especially useful in the context of verification [9], but it requires a more sophisticated approach to incremental generation than one-time extensions. Section 4.1 introduces some terminology for reasoning in chains of extensions. In Section 4.2 we show how *LIG* can be extended to chains of extensions, followed by examples (Section 4.3) and a correctness argument (Section 4.4).

4.1 Hierarchic Reasoning in Chains of Extensions

Consider a base theory $\mathcal{T}_0$ and a number of clause sets $\mathcal{K}_1, \ldots, \mathcal{K}_m$. For $j = 0, \ldots, m-1$, let $\mathcal{T}_{j+1} = \mathcal{T}_j \cup \mathcal{K}_{j+1}$ and assume that $\mathcal{T}_j \subseteq \mathcal{T}_{j+1}$ is a local extension. Then, for every extension, we can use local reasoning to reduce a given ground goal G from theory $\mathcal{T}_j$ to theory $\mathcal{T}_{j-1}$:

$$G \models_{\mathcal{T}_j} \Box \Leftrightarrow \mathcal{K}_j[G] \cup G \models_{\mathcal{T}_{j-1}} \Box.$$

If we define

$$\begin{aligned} G_m &= G \\ G_{j-1} &= \mathcal{K}_j[G_j] \cup G_j \ \text{(for } 1 \leq j \leq m\text{)}, \end{aligned}$$

then $G \models_{\mathcal{T}_m} \Box \Leftrightarrow G_0 \models_{\mathcal{T}_0} \Box$.

For incremental generation we cannot use the given *LIG* procedure, as that would require for every reduction from $\mathcal{T}_j$ to $\mathcal{T}_{j-1}$ an SMT procedure for $\mathcal{T}_{j-1}$ to already exist. On the other hand, reasoning in chains of extensions can benefit very much from incremental generation of instances, since the ground goal G_j grows polynomially with every reduction. Therefore, we supply a more sophisticated method *LIG** that can treat several extensions at once.

4.2 Chain-Local Instance Generation (*LIG**)

We present the procedure *LIG**, which solves the satisfiability problem of a ground goal G in a repeatedly extended theory $\mathcal{T} \cup \mathcal{K}_1 \cup \ldots \cup \mathcal{K}_n$, and differs from *LIG* mainly in two points: First, for m successive extensions, axioms need to be kept in m different sets. Second, the search space is restricted to the sets of instances G_j defined above.

Proof procedure. For a given input $(\mathcal{K}_1, \ldots, \mathcal{K}_m, G)$, LIG^* keeps axioms from the different extensions and ground clauses in separate sets. The selection function sel is defined as in LIG, except that we have different extension symbols for each $\mathcal{K}_j$. For every j, $\mathsf{sel}(\mathcal{K}_j)$ consists of the literals with the highest number of variables, preferably below extension symbols of $\mathcal{K}_j$. Then, the following two steps are repeated until termination:

(1) Ground satisfiability check. We give G to an SMT solver. If $G \models_{\mathcal{T}_0} \Box$, then the procedure terminates and returns unsatisfiable. Otherwise, the solver returns a model M such that $M \models_{\mathcal{T}_0} G$. As before, generate from M a partial Herbrand model $\mathcal{M}$ of G.

(2) Instance generation. Define the following sets of literals:

$$\begin{aligned} \mathcal{L}_{m+1} &= \mathcal{M} \\ \mathcal{L}_j &= \mathsf{sel}(\mathcal{K}_j)[\textstyle\bigcup_{j<k\leq m+1} \mathcal{L}_k] \text{ (for } 1 \leq j \leq m) \end{aligned}$$

Then, instances are generated according to the following rule:

$$LIG^* \qquad \frac{(L_1 \vee D_1) \ \cdots \ (L_n \vee D_n) \ \mid \mathcal{M}}{(L_1 \vee D_1)\sigma \ \cdots \ (L_n \vee D_n)\sigma}$$

where:

(i) the $L_i \vee D_i$ are (variants of) clauses from $\mathcal{K}_1 \cup \ldots \cup \mathcal{K}_m$ with $\mathsf{sel}(L_i \vee D_i) = L_i$,

(ii) σ is a substitution such that all $L_i\sigma$ are ground literals,

(iii) if $(L_i \vee D_i) \in \mathcal{K}_j$, then $(L_i \vee D_i)\sigma$ is a generalization of a clause in $\mathcal{K}_j[\bigcup_{j<k\leq m+1} \mathcal{L}_k]$, and

(iv) $\{L_i\sigma\}_{1\leq i\leq n} \cup \mathcal{M} \models_{\mathcal{T}} \Box$

Again, several variants of an axiom may be instantiated in one inference. Ground clauses are added to G, non-ground instances $C\sigma$ with $C \in \mathcal{K}_j$ are added to $\mathcal{K}_j$. The notion of saturation applies as before. If $(\mathcal{K}_1, \ldots, \mathcal{K}_n, G)$ is saturated under LIG^*, the procedure terminates and states satisfiability of the input. Otherwise, define sel on new axioms as described above and return to **(1)**.

Search for substitutions. The remarks we made for LIG apply analogously: for every axiom C, we can watch a set of literals $\mathcal{L} \subseteq C$ to ensure conditions (ii) and (iii) of the inference rule. Then, we only need to consider those substitutions σ which map variables of $\mathsf{sel}(C)$ to ground terms such that resulting ground extension terms (in the extension $\mathcal{K}_j \supseteq C$) in $\mathcal{L}\sigma$ are in $\mathsf{st}(\mathcal{L}_j)$. We omit a proof of this property, which is a straightforward lifting of the proof of Lemma 1.

Special cases, unsatisfiable cores. The special cases apply as before, except that axioms $C\sigma$ have to be added to the set $\mathcal{K}_j$ which contains C. Also, unsatisfiable cores can be computed as before.

4.3 Examples: Applications of *LIG**

In previous work on verification of parametric systems [9] we introduced typical applications of LIG^*. Due to space restrictions, we can only sketch the idea here.

Example 5. Local axioms can be used to model parameterized systems by using sets of guarded boundedness axioms (see Section 2.2) as update rules. Consider a system were a function f is used to model the current state of the system (e.g., the contents of a data structure), and updates are defined by

$$\mathsf{update} = \{\phi_i(\overline{x}) \rightarrow s_i(\overline{x}) \leq f'(\overline{x}) \leq t_i(\overline{x}) \mid 1 \leq i \leq n\}\,,$$

i.e. f' is used to model the state of the system after the update. A desired invariant of such a system may be that f is sorted, i.e. if the monotonicity axiom for f ($\mathsf{Mon}(f)$) holds before an update, it should also hold for f' after the update. To prove this, one can show that $\mathsf{Mon}(f) \cup \mathsf{update} \cup \neg\mathsf{Mon}(f') \models_{\mathcal{T}_0} \square$ by using locality of the first extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{Mon}(f)$ and the second extension $\mathcal{T}_0 \cup \mathsf{Mon}(f) \subseteq \mathcal{T}_0 \cup \mathsf{Mon}(f) \cup \mathsf{update}$.

Example 6. If the safety property of a system is not inductive, we may resort to bounded model checking (BMC). Consider a system where an update can be expressed as a local theory extension, and this extension can be repeated (after proper renaming of extension symbols) in order to model successive updates. Then, LIG^* can be used to prove safety of the system for m steps by proving

$$\mathsf{init} \wedge \mathsf{update}_1 \wedge \ldots \wedge \mathsf{update}_m \wedge \neg\mathsf{safe} \models_{\mathcal{T}_0} \square,$$

where init is the initial condition of the system, update gives the update rules (with subscript i for the ith step), and safe gives the safety condition.

Since eager instantiation will blow up the ground goal polynomially in each extension, we should benefit a lot from an incremental approach.

4.4 Correctness of *LIG**

In the following we state that LIG^* is sound, complete, and terminating. Detailed proofs are omitted due to space restrictions. They are straightforward liftings of the corresponding proofs in Section 3.3, with $\mathcal{K}_1 \cup \ldots \cup \mathcal{K}_m$ replacing $\mathcal{K}$, G_0 replacing $\mathcal{K}[G]$, and with $\mathcal{L} = \bigcup_{1 \leq j \leq m+1} \mathcal{L}_j$.

Consider a base theory $\mathcal{T}_0$ and successively extended theories $\mathcal{T}_{j+1} = \mathcal{T}_j \cup \mathcal{K}^0_{j+1}$. An *$LIG^*$-derivation* is a sequence of tuples $(\mathcal{K}^0_1, \ldots, \mathcal{K}^0_m, G^0) \vdash \ldots \vdash (\mathcal{K}^n_1, \ldots, \mathcal{K}^n_m, G^n)$ such that for $0 \leq j \leq m-1$, $\mathcal{T}_j \subseteq \mathcal{T}_j \cup \mathcal{K}^0_{j+1}$ is a local theory extension, and for $0 \leq i \leq m-1$, G^i is $\mathcal{T}_0$-satisfiable and $(\mathcal{K}^{i+1}_1, \ldots, \mathcal{K}^{i+1}_m, G^{i+1})$ is the result of an LIG^*-inference on $(\mathcal{K}^i_1, \ldots, \mathcal{K}^i_m, G^i)$.

Theorem 4 (Soundness). *Let $(\mathcal{K}^0_1, \ldots, \mathcal{K}^0_m, G^0) \vdash \ldots \vdash (\mathcal{K}^n_1, \ldots, \mathcal{K}^n_m, G^n)$ be an LIG^*-derivation. If $G^n \models_{\mathcal{T}_0} \square$, then $G^0 \models_{\mathcal{T}_m} \square$.*

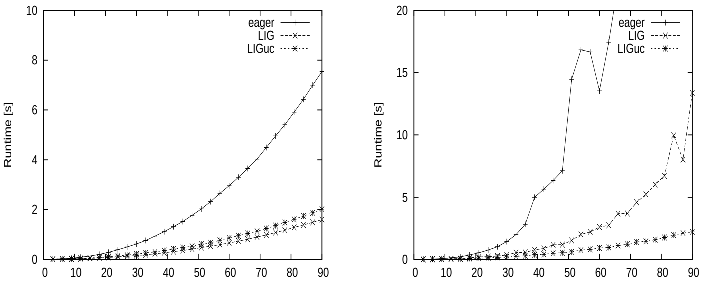
Theorem 5 (Completeness). *Let* $(\mathcal{K}_1^0, \dots, \mathcal{K}_m^0, G^0) \vdash \dots \vdash (\mathcal{K}_1^n, \dots, \mathcal{K}_m^n, G^n)$ *be an LIG***-derivation. If* G^n *is* $\mathcal{T}_0$*-satisfiable and* $(\mathcal{K}_1^n, \dots, \mathcal{K}_m^n, G^n)$ *is saturated under LIG** *with respect to a given selection function* sel *and a Herbrand model* $\mathcal{M}$ *of* G^n*, then* G^0 *is* $\mathcal{T}_m$*-satisfiable.*

Theorem 6 (Termination). *For any input* $(\mathcal{K}_1^0, \dots, \mathcal{K}_m^0, G^0)$ *(with the* $\mathcal{K}_i^0$ *and* G^0 *as defined above), LIG** *terminates after a finite number of inferences.*

5 Experimental Results

We implemented the incremental approaches on top of an existing OCaml implementation of the eager approach, with Z3 as our blackbox SMT solver. We tested our implementation on four sets of benchmarks:[4]

In Figure 1 (a) we see runtimes of all three approaches on the parameterized problem from Example 3. The incremental approaches outperform the eager instantiation substantially, while LIG_{uc} is a bit slower than LIG. Figure 1 (b) shows runtimes for Example 4: here, LIG_{uc} is significantly faster than LIG.



(a) Example 3

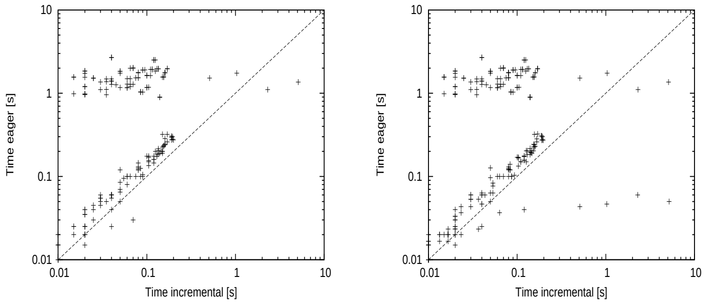
(b) Example 4

Fig. 1. Runtime comparison of the three approaches on parameterized example

In Figure 2, we see runtimes of LIG and the eager approach on a set of verification benchmarks taken from the QF_UFLIA/wisas section of SMT-LIB.[5] We checked satisfiability when the background theory assumes one of the functions to be either monotone or injective. All of the problems turned out to be unsatisfiable in both of the extended theories. Figure 2 (a) compares performance of the two approaches for the subset of originally satisfiable problems. In the diagram, a point above the diagonal means that LIG is faster than eager instantiation on a given problem, and vice versa for points below the diagonal. We can see that LIG is faster on allmost all of the problems.

[4] The implementation of LIG_{uc} is still preliminary, therefore it is only included in two of the four comparisons below.

[5] These were originally benchmarks of the Wisconsin Safety Analyzer.



(a) SMT-LIB:WiSA (sat) (b) SMT-LIB:WiSA (all)

Fig. 2. Runtime comparison of *LIG* and eager approach on verification problems

For the set of all benchmarks from the QF_UFLIA/wisas section of SMT-LIB, we had mixed results (see Figure 2 (b)): although *LIG* is faster on the vast majority, there are some problems for which it needs much longer than the eager method. It turned out that these problems are already unsatisfiable without the additional axioms, but proving unsatisfiability of the original problem was much harder than proving unsatisfiability after adding a few hundred axiom instances.

6 Conclusions and Future Work

We have introduced incremental procedures for reasoning in local theory extensions and chains of local extensions. We have shown that our methods are decision procedures and generate less instances than the standard approach. Experimental results indicate that already *LIG*, which always takes maximal inferences, is considerably more efficient than eager instantiation. LIG_{uc}, taking minimal inferences, can outperform *LIG* at least for satisfiable problems.

In part, our approach resembles quantifier handling in SMT procedures. This holds specifically for the matching of extension terms in watched literals to ground terms in the model. The main differences are our additional semantic condition for instantiation, and matching modulo equality in SMT solvers, in contrast to our syntactical matching. It would be interesting to check whether we could benefit somehow from using E-matching instead of syntactic matching.

A similar approach of generating instances incrementally, guided by a candidate model, has been followed by Veanes et al. [2] in context of bounded reachability analysis. Although not explained in much detail, they seem to use an even stronger semantic condition: only instances which contradict the current candidate model are generated. Furthermore, their search for substitutions is guided by all ground terms appearing in the problem, where our approach takes into account only the terms in the candidate model.

An extension of our work in this paper would be to consider more general notions of locality. In previous work, we have shown that both the array property fragment of Bradley et al. [1] and the pointer data structures of McPeak and Necula [10] can be expressed as Ψ-local extensions [8]. Thus, extending our approach to Ψ-locality would give us an incremental procedure for these fragments.

Acknowledgments. Thanks to the German Research Council (DFG) for financial support of this work 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). Many thanks also to Viorica Sofronie-Stokkermans for our fruitful discussions, and to Leonardo de Moura for supporting the integration of Z3 and answering many questions.

References

1. A.R. Bradley, Z. Manna, and H.B. Sipma. What's decidable about arrays? In *Verification, Model-Checking, and Abstract-Interpretation, 7th Int. Conf. (VMCAI 2006)*, LNCS 3855, pages 427–442. Springer, 2006.
2. M. Veanes, N. Bjørner and A. Raschke. An SMT Approach to Bounded Reachability Analysis of Model Programs. In *28th IFIP WG6.1 International Conference on Formal Techniques for Networked and Distributed Systems (FORTE 2008)*, LNCS 5048, pp. 53–68. Springer, 2008.
3. D. Detlefs, G. Nelson and J.B. Saxe. Simplify: a theorem prover for program checking. In *Journal of the ACM*, 52(3):365–473, 2005.
4. H. Ganzinger. Relating semantic and proof-theoretic concepts for polynomial time decidability of uniform word problems. In *16th IEEE Symposium on Logic in Computer Science (LICS'01)*, pp. 81–92. IEEE Computer Society Press, 2001.
5. H. Ganzinger and K. Korovin. Theory Instantiation. In *Logic for Programming, Artificial Intelligence, and Reasoning, 13th Int. Conf. (LPAR 2006)*, LNCS 4246, pp. 497–511. Springer, 2006.
6. Y. Ge, C. Barrett and C. Tinelli. Solving Quantified Verification Conditions Using Satisfiability Modulo Theories. In *21st International Conference on Automated Deduction (CADE-21)*, LNCS 4603, pp. 167–182. Springer, 2007.
7. R. Givan and D. McAllester. Polynomial-time computation via local inference relations. In *ACM Transactions on Computational Logic*, 3(4):521–541, 2002.
8. C. Ihlemann, S. Jacobs and V. Sofronie-Stokkermans. On local reasoning in verification. In *14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008)*, LNCS 4963, pp. 265–281. Springer, 2008.
9. S. Jacobs and V. Sofronie-Stokkermans. Applications of Hierarchical Reasoning in the Verification of Complex Systems. In *Fourth Workshop on Pragmatics of Decision Procedures in Automated Reasoning (PDPAR 2006)*, ENTCS 174(8), pp. 39–54. Elsevier, 2007.
10. S. McPeak and G.C. Necula. Data structure specifications via local equality axioms. In *Computer Aided Verification, 17th International Conference, CAV 2005*, LNCS 3576, pages 476–490, 2005.
11. R. Nieuwenhuis, A. Oliveras, E. Rodriguez-Carbonell and A. Rubio. Challenges in Satisfiability Modulo Theories. In *Term Rewriting and Applications, 18th Int. Conf. (RTA 2007)*, LNCS 4533, pp. 2–18. Springer, 2007.
12. V. Sofronie-Stokkermans. Hierarchic reasoning in local theory extensions. In *20th International Conference on Automated Deduction (CADE-20)*, LNAI 3632, pp. 219–234. Springer, 2005.