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. 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 this set 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.

1 Introduction

Handling quantified formulas is one of the big challenges in satisfiability modulo theories today [10]. Current approaches to solve problems involving quantifiers [7] 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 [11] gives a method to handle a certain class of quantified formulas in a complete way by instantiating them to a finite set of ground instances. As this set grows with the number of ground terms appearing in the given problem, generating all instances at once becomes inefficient or even intractable for large problems.

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 [4–6]: 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.

The remainder of the paper is organized as follows: in Section 2 we introduce the logical terminology used throughout this paper, as well as the underlying methods of instantiation-based theorem proving and local reasoning. In Section 3, we introduce our basic approach, together with proofs of correctness. This approach is refined in Section 4, resulting in broader applicability. We conclude the paper with remarks on ongoing work, including implementation.

2 Instance Generation and Local Reasoning

This section introduces necessary terminology and gives a short introduction to the methods that form the basis of our approach. For more detailed descriptions, including theoretical background, we refer to Ganzinger and Korovin [4–6] for instantiation-based theorem proving and to Sofronie-Stokkermans et al. [11, 12, 9, 8] for local reasoning.

2.1 Logical Prerequisites

We use the usual symbols, notation and terminology of (either sorted or unsorted) first-order logic with standard definitions.

Formulas and instances. We use x, y to denote variables, a, b, c, d for constants and f, g as non-constant function symbols. L and K 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* or *instantiation* of F. It is a *ground instance* if it is variable-free, otherwise it is a *partial instantiation*. 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 $F_1\sigma = F_2$. If F_2 is an instance of F_1, then F_1 is a *generalization* of F_2. 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 for every $F_3 \in \mathcal{F}$ such that F_3 is both an instance of F_1 and a generalization of F_2, F_3 is a variant of F_1. For a formula F, let $\mathsf{st}(F)$ be the set of ground subterms appearing in F.

Theories and models. Consider a (many-sorted) 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). In a many-sorted framework, every variable has a sort $s \in S$. 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 $P \in \mathsf{Pred}$. A *theory* $\mathcal{T}$ can be defined by a set of axioms or a set of models. If $\mathcal{T}$ is defined by axioms, then a given Π-structure M is a *model* of a theory $\mathcal{T}$ if 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 $F \models_{\mathcal{T}} \square$, where $\square$ represents the empty clause. Validity (modulo $\mathcal{T}$) of a formula F in a Π-structure M is depicted by $M \models_{\mathcal{T}} F$.

2.2 Instance Generation Methods

Resolution-based instance generation (*IG*) is an instantiation-based calculus for first-order logic, introduced by Ganzinger and Korovin [4]. Later, they presented extensions of the calculus to first-order logic with equality [5] and to instantiation modulo background theories [6]. The calculus we present in the following is close

to the last one, but subsumes all three versions. Like in the original papers, IG works in an unsorted framework.[1]

Proof Procedure. Given a set of clauses F, IG checks satisfiability of F modulo a background theory $\mathcal{T}$ (which may be the empty theory) by interleaving ground satisfiability checks with instantiation of clauses. Let $\bot$ denote both a distinguished constant and a substitution mapping all variables to this constant. Then, IG repeats the following two steps until termination:

(1) Ground satisfiability check. If $F\bot \models_{\mathcal{T}} \Box$, the procedure terminates and states unsatisfiability of the input. Otherwise, generate a model M such that $M \models_{\mathcal{T}} F\bot$.

(2) Instance generation. Consider a *selection function* sel which selects from every clause $C \in F$ a literal $L \in C$ such that $M \models_{\mathcal{T}} L\bot$. Then, instances are generated according to the following inference rule:

$$IG \qquad \frac{(L_1 \vee D_1) \quad \ldots \quad (L_n \vee D_n)}{(L_1 \vee D_1)\sigma \quad \ldots \quad (L_n \vee D_n)\sigma}$$

where each L_i is selected by sel, and σ is a substitution such that $L_1\sigma \wedge \ldots \wedge L_n\sigma \models_{\mathcal{T}} \Box$.

Note that (variants of) the same clause $C \in F$ may be used several times as an input, and thus can be instantiated several times. We consider a clause C *redundant with respect to* F if there is a clause $D \in F$ which is a variant of C.[2] If all inferences based on sel only produce instances which are redundant with respect to F, we call F *saturated* under IG (with respect to the selection function sel). If F is saturated under IG, the procedure terminates and states satisfiabliliy of the input. Otherwise, after an inference step which adds at least one non-redundant clause to F, we go back to **(1)**.

Example 1 (taken from [5]). Let $\mathcal{T}$ be the theory of equality, and

$$F = \{\underline{f(g(x)) = c} \vee g(g(x)) \neq a,\ \underline{g(y) = y},\ \underline{f(a) \neq c}\}.$$

The satisfiability check shows that $F\bot$ is satisfiable. Suppose the underlined literals are selected by sel. A substitution that makes the selected literals unsatisfiable is $\sigma = [a/x, a/y]$. The new set of clauses is obtained by applying σ to the old clauses and adding the resulting instances: $F' = F \cup \{f(g(a)) = c \vee g(g(a)) \neq a,\ g(a) = a,\ f(a) \neq c\}$. $F'\bot$ is unsatisfiable and the procedure terminates.

Finding substitutions. Thus far, we have not specified how the substitution σ for the inference is computed, or how to prove that such a σ does not exist.

[1] Even though we will work in a sorted framework later, we will see that we do not have to add explicit sort information to the instance generation part.

[2] Ganzinger and Korovin introduced a formal notion of redundancy, based on implication of ground instances of clauses by smaller clauses wrt. an ordering. For the moment, we only use this simpler definition.

For a given theory, a procedure that finds such a substitution if one exists is called *answer-complete*. E.g., unification is an answer-complete procedure for the empty theory, and paramodulation is answer-complete for the theory of equality. In general, answer computation is undecidable.

Correctness. An *IG-derivation* is a sequence of sets $F_0 \vdash F_1 \vdash \ldots \vdash F_n$ such that for $0 \leq i \leq n-1$, $F_i \bot$ is $\mathcal{T}$-satisfiable and F_{i+1} is the result of an IG-inference on F_i.

The following results have been proved by Ganzinger and Korovin [4–6]: Let $F_0 \vdash F_1 \vdash \ldots \vdash F_n$ be an IG-derivation. If $F_n \bot \models_{\mathcal{T}} \Box$, then $F_0 \models_{\mathcal{T}} \Box$. If F_n is $\mathcal{T}$-satifiable and saturated under IG, then F_0 is $\mathcal{T}$-satisfiable. IG is refutationally complete if every persisting inference is eventually taken, and if the background theory $\mathcal{T}$ has a universal axiomatization and an answer-complete procedure for finding unsatisfiable instances.

2.3 Reasoning in Local Theory Extensions

Hierarchic reasoning in local theory extensions, or in short: *local reasoning*, has been introduced by Sofronie-Stokkermans [11]. 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 sorted background theory $\mathcal{T}$ with signature $\Pi_0 = (S_0, \Sigma_0, \mathsf{Pred})$. 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, which may in turn introduce additional sorts, given by S_1. If F and G are formulas in the signature $\Pi = (S_0 \cup S_1, \Sigma_0 \cup \Sigma_1, \mathsf{Pred})$, we will reason in $\mathcal{T}^{\Sigma_1}$, the extension of $\mathcal{T}$ with free function symbols from Σ_1 (and the corresponding free sorts S_1). Whenever Σ_1 (and S_1) are 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 possibly new sorts S_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 $\mathcal{K}$. Non-constant function symbols $f \in \Sigma_1$ are called *extension symbols*, terms $f(t)$ are *extension terms*. Note that $F \models_{\mathcal{T} \cup \mathcal{K}} G$ is equivalent to $\mathcal{K} \cup F \models_{\mathcal{T}} G$.

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*: there are no constant or function symbols below an extension symbol in C, all extension terms which contain the same variable are (syntactically) equal, and no extension term contains two occurrences of the same variable.[3] 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

[3] This restriction is needed to prove locality of an extension and is also used throughout previous papers [11, 8]

$\mathcal{K}[G] = \{C\sigma \mid C \in \mathcal{K}$ and $f(t)\sigma \in \mathsf{st}(G)$ for each extension subterm $f(t)$ 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, $G \models_{\mathcal{T}\cup\mathcal{K}} \Box \Leftrightarrow \mathcal{K}[G] \cup G \models_{\mathcal{T}} \Box$,

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 [11, 12, 9, 8], several theory extensions have been proved to be local.[4] Most of them satisfy our additional restriction, including:

- *monotone functions* over several theories with a partial or total ordering,
- *strictly monotone functions* mapping from a partially or totally ordered set into a densely ordered one,
- *Lipschitz functions*, i.e. real-valued functions satisfying the Lipschitz condition at a given point,
- *injective functions*, if domain and codomain allow them,
- *guarded boundedness* conditions of the form

$$\bigwedge_{i=1}^{n} (\phi_i(\overline{x}) \rightarrow s_i(\overline{x}) \leq f(\overline{x}) \leq t_i(\overline{x})),$$

  where the ϕ_i are mutually exclusive and the s_i, t_i are terms in the base theory such that $\phi_i(\overline{x}) \rightarrow s_i(\overline{x}) \leq t_i(\overline{x})$ 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. We can check the satisfiability of a set G of ground clauses modulo the extended theory $\mathcal{T} \cup \mathcal{K}$ in two simple steps (for details and the general approach cf. [11]):

Step 1: Use locality. 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.

Step 2: Reduction to testing satisfiability in $\mathcal{T}$. There are two ways to check whether $\mathcal{K}[G] \cup G \models_{\mathcal{T}} \Box$ (where Σ_1-symbols are treated as free function symbols): i) removing the extension functions by Ackermann's reduction, and then solving a satisfiability problem in $\mathcal{T}$, or ii) treating the problem with a combined decision procedure for $\mathcal{T}$ and EUF. See [1] for a comparison of the approaches.

In the following we assume that a given SMT solver for ground satisfiability problems in the background theory $\mathcal{T}$ can also solve ground problems in EUF$\cup\mathcal{T}$.

Example 2. 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.

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

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

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

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

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

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

Combinations and chains of extensions. If we want to treat an extension $\mathcal{K}$ which is not known to be local wrt. the base theory, we have two possibilities which may still allow us to treat them within the local reasoning framework:

If we have $\mathcal{K} = \mathcal{K}_1 \cup \mathcal{K}_2$, and both $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}_1$ and $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}_2$ are local extensions, then $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}_1 \cup \mathcal{K}_2$ may again be a local extension itself. Sofronie-Stokkermans [12] identified circumstances where this is the case.

Otherwise, we may 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. 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 gives us a decision procedure for satisfiability problems in local extensions of theories. Theoretically, this means that the procedure is guaranteed to terminate on both satisfiable and unsatisfiable input, given unlimited resources. As in practice neither time nor space are unlimited, we are looking for methods to treat local theory extensions more efficiently. 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. In Section 3.2, we give some examples how local reasoning can benefit from this approach, and in Section 3.3 we show that the method is sound and complete.

3.1 Local Instance Generation (LIG)

We present a refinement of IG, which uses knowledge about locality in order to obtain a decision procedure. Furthermore, axioms and ground clauses are stored seperately, and only the ground part determines the candidate model — selection on axioms is fixed by a heuristic. For reasoning on ground clauses we

use an SMT solver, which should allow incremental addition of constraints and must be able to return a model for a set of satisfiable clauses.

Separation of axioms and ground goal. In LIG, we keep axioms $\mathcal{K}$ and ground clauses G in different sets, on which we define a selection function sel. For a set of clauses F, let $\mathsf{sel}(F) = \{\mathsf{sel}(C) \mid C \in F\}$. Then, as in IG, $\mathsf{sel}(G)$ is based on a $\mathcal{T}$-model of G. We define $\mathsf{sel}(\mathcal{K})$ 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. In contrast to $\mathsf{sel}(G)$, $\mathsf{sel}(\mathcal{K})$ remains fixed throughout the saturation process.

Proof Procedure. Given a set of axioms $\mathcal{K}$ and a set of ground clauses G, LIG checks satisfiability of G modulo a locally extended theory $\mathcal{T} \cup \mathcal{K}$ by interleaving satisfiability checks (modulo $\mathcal{T}$) with instantiation of axioms in $\mathcal{K}$. 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, the solver returns a model M such that $M \models_{\mathcal{T}} G$.

(2) Instance generation. Consider a selection function sel which is defined on G such that $M \models_{\mathcal{T}} \mathsf{sel}(G)$, and on $\mathcal{K}$ as described above. Then, instances are generated according to the following inference rule, where premises and conclusions are separated into axioms (left-hand side) and ground clauses (right-hand side). Generated axioms are added to $\mathcal{K}$, ground clauses to G.

$$LIG \qquad \frac{F \mid G_0}{F_1\sigma \mid F_0\sigma}$$

where:

(i) F is a set of (variants of) clauses from $\mathcal{K}$ and $G_0 \subseteq G$,
(ii) σ is a substitution such that $\mathsf{sel}(F)\sigma$ is a set of ground literals,
(iii) all clauses in $F\sigma$ are generalizations of clauses in $\mathcal{K}[\mathsf{sel}(G)]$,
(iv) $F_1 \subseteq F$ and $F_0 \subseteq F$ such that $F_1\sigma$ consists of axioms, $F_0\sigma$ of ground clauses, with $F_1 \cap F_0 = \emptyset$ and $F_1 \cup F_0 = F$, and
(v) $\mathsf{sel}(F)\sigma \wedge \mathsf{sel}(G_0) \models_{\mathcal{T}} \square$ (where literals in $\mathsf{sel}(F)\sigma$ and $\mathsf{sel}(G_0)$ are conjunctively connected)

As in IG, several variants of a clause $C \in \mathcal{K}$ may be instantiated in one inference. The notions of redundancy and saturation apply as before. 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. Opposed to the general case, we can restrict the search for substitutions to a finite set: the following lemma shows how side conditions (ii) and (iii) can be ensured.

Lemma 1 *Let $F = \{(L_1 \vee D_1), \ldots, (L_n \vee D_n)\}$ and G_0 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, the following two are equivalent:*

(1) σ *satisfies conditions* (ii) *and* (iii) *of the LIG inference rule.*
(2) σ *instantiates variables in the* L_i *to ground terms such that all ground extension terms in* $\mathcal{L}_i\sigma$ *are in* $\mathsf{st}(\mathsf{sel}(G))$.

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

Now suppose (2) holds. Then (ii) holds because variables in all L_i are instantiated to ground terms. $\mathcal{K}$ is Σ_1-linear, so 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 $\mathsf{st}(\mathsf{sel}(G))$, so must be all ground extension terms in $(L_i \vee D_i)\sigma$. By definition of $\mathcal{K}[\mathsf{sel}(G)]$, every $(L_i \vee D_i)\sigma$ is a generalization of a clause in $\mathcal{K}(\mathsf{sel}(G))$, satisfying (iii). □

The lemma suggests that in order to ensure conditions (ii) and (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. Then we only need to consider substitutions σ mapping variables of L to ground terms such that all ground extension terms in $\mathcal{L}\sigma$ are in $\mathsf{st}(\mathsf{sel}(G))$.

Special cases. The following special cases make search for a substitution easier, and are the reason why sel is predefined on $\mathcal{K}$ (and does not depend on a model of $\mathcal{K}\bot$, as it would in IG). 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 $\mathcal{L} = \{\mathsf{sel}(C)\}$.

Sort information. Note that we do not have to add explicit sort information to the LIG inference rule, as the restriction to clauses in $\mathcal{K}[G]$ ensures that variables will only be instantiated with terms of the right sort.

Unsatisfiable cores. In order to minimize the number of generated instances, one should eliminate premises of an inference which are not needed to satisfy condition (v) of the LIG rule. If $\mathsf{sel}(F)\sigma \wedge \mathsf{sel}(G_0) \models_{\mathcal{T}} \square$, then a decision procedure for $\mathcal{T}$ can be used to compute an *unsatisfiable core* of $\mathsf{sel}(F) \wedge \mathsf{sel}(G_0)$, i.e. smallest subsets $F' \subseteq \mathsf{sel}(F), G' \subseteq \mathsf{sel}(G_0)$ such that $F' \wedge G' \models_{\mathcal{T}} \square$. Only inferences for which the set $\mathsf{sel}(F)$ is itself an unsatisfiable core should be used, additional premises (and conclusions) can be dropped.

3.2 Examples: Behaviour of *LIG*

In the following examples, we compare the behaviour of LIG to the standard local reasoning approach from Section 2.3. We do not include a comparison to IG

since we in general do not have an answer-complete procedure for the theories we consider.

For all examples, consider the same background theory and extension as in Example 2, i.e. $\mathcal{T}$ has a partial ordering and $\mathcal{K} = \{\ x \leq y \rightarrow f(x) \leq f(y)\ \}$.

Example 3. Let $G = \{\ a \leq b,\ \neg(f(a) \leq f(b))\ \}$. Then, as seen in Example 2, standard local reasoning 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 $\mathsf{sel}(G) = G$ (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 in the model given by $\mathsf{sel}(G)$, and the resulting terms (starting with f) already appear in $\mathsf{sel}(G)$.

In the worst case this means that we produce the whole set $\mathcal{L} = \mathsf{sel}(\mathcal{K})[G]$, containing 4 instances of the literal $f(x) \leq f(y)$. The only unsatisfiable core of $\mathcal{L} \wedge \mathsf{sel}(G)$ is $\{\ f(a) \leq f(b),\ \neg(f(a) \leq f(b))\ \}$. Thus, LIG will produce the instance $a \leq b \rightarrow f(a) \leq f(b)$ with its first inference and add it to G, which makes G unsatisfiable.

We can see that the search for a substitution is reduced to the level of literals, and the resulting clause instance contributes to the proof of unsatisfiability of $\mathcal{K}[G] \cup G$. As a result, the SMT procedure does not have to search through a number of candidate models which is in the worst case exponential in the size of $\mathcal{K}[G]$. The following examples show that this effect is even bigger if G contains information that does not contribute to the proof of unsatisfiability.

Example 4. Let $G = \{\ a \leq b,\ \neg(f(a) \leq f(b)),\ c \leq d,\ f(c) \leq d,\ \neg(d \leq f(d))\ \}$.

Standard local reasoning generates the set $\mathcal{K}[G]$, consisting of all instances of the monotonicity axiom with all combinations of substituting x and y with a, b, c, d. I.e., $|\mathcal{K}[G]| = 4^2 = 16$.

As above, in LIG we have $\mathsf{sel}(\mathcal{K}) = \{f(x) \leq f(y)\}$ and $\mathsf{sel}(G) = G$. With the larger G, $\mathcal{L} = \mathsf{sel}(\mathcal{K})[G]$ contains 16 instances of $f(x) \leq f(y)$. The only unsatisfiable core of $\mathcal{L} \cup G$ is again $\{f(a) \leq f(b), \neg(f(a) \leq f(b))\}$, and we produce the same instance as above, making G unsatisfiable.

Example 5. Let G consist of the ground clauses

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

Then $\mathcal{K}[G]$ consists of all instances of the monotonicity axiom with all combinations of substituting x and y with the a_i, b_i, c_i, d_i, i.e. $|\mathcal{K}[G]| = (4n)^2$.

In LIG, $\mathsf{sel}(\mathcal{K})$ is as above, and $\mathsf{sel}(G)$ may be arbitrary, but the second literal has to be selected either in E or in F (because the selection on C_1 contradicts the first literal in either E or F). Thus, $\mathsf{sel}(G)$ contains n literals from the D_i with occurrences of either $f(a_i)$ and $f(b_i)$, or $f(c_i)$ and $f(d_i)$, for $1 \leq i \leq n$.

Furthermore, we have either $\mathsf{sel}(E) = \neg(f(a_1) \leq f(b_1))$ or $\mathsf{sel}(F) = \neg(f(c_1) \leq f(d_1))$ (but not both, because of $\mathsf{sel}(D_1)$), which means that either $f(a_1)$ and $f(b_1)$ or $f(c_1)$ and $f(d_1)$ are in $\mathsf{st}(\mathsf{sel}(G))$. Overall, $\mathsf{st}(\mathsf{sel}(G))$ contains $2n+1$ extension terms and in search for a substitution we have to generate $(2n+1)^2$ instances of the literal $f(x) \leq f(y)$, in the worst case.

Depending on sel, we will either find that $\{f(a_1) \leq f(b_1), \neg(f(a_1) \leq f(b_1))\}$ or $\{f(c_1) \leq f(d_1), \neg(f(c_1) \leq f(d_1))\}$ is the only unsatisfiable core to be found. After adding the corresponding instance of the axiom, sel will not change on most of G (if the SMT procedure is smart): because of the added instance, sel has to change on C_1, D_1, E and F, the rest of G is not affected. If the SMT procedure uses the same valuation as before on the rest of the clauses, no additional literals have to be generated when searching for a substitution in the next step. In this search, the remaining one of the two unsatisfiable cores from above will be found, and the corresponding instance added to G, making it $\mathcal{T}$-unsatisfiable.

Thus, we have proved G unsatisfiable in $\mathcal{T} \cup \mathcal{K}$ by generating only 2 instances and searching through a maximum of $(2n+1)^2$ literals, instead of generating $(4n)^2$ instances.

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 2 (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 $G^n \models_{\mathcal{T}} \square$, then $\mathcal{K}^0 \cup G^0 \models_{\mathcal{T}} \square$.*

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

The following lemma is needed for our proof of completeness:

Lemma 3 (Selection) *Let $\mathcal{T}$ be an arbitrary theory, $\mathcal{K}$ a local extension of $\mathcal{T}$ and G an arbitrary set of ground clauses. Then, if $\mathcal{S}$ is the set of all functions which select from every clause C a literal $L \in C$, we have*

$$\mathcal{K} \cup G \models_{\mathcal{T}} \square \Leftrightarrow (\mathsf{sel}(\mathcal{K}[\mathsf{sel}(G)]) \cup \mathsf{sel}(G) \models_{\mathcal{T}} \square \quad \forall\ \mathsf{sel} \in \mathcal{S}).$$

Proof: In the first step of the proof we use the fact that $G \models_{\mathcal{T}} \square$ iff $\mathsf{sel}(G) \models_{\mathcal{T}} \square$ for every $\mathsf{sel} \in \mathcal{S}$. This equivalence relies on the fact that for ground clauses existence of a model is equivalent to making true at least one literal per clause.

$$\begin{array}{rll} & G & \models_{\mathcal{T} \cup \mathcal{K}} \square \\ \Leftrightarrow & (\ \mathsf{sel}(G) & \models_{\mathcal{T} \cup \mathcal{K}} \square \quad \forall\ \mathsf{sel} \in \mathcal{S}\ ) \\ \text{(by locality)} \Leftrightarrow & (\ \mathcal{K}[\mathsf{sel}(G)] \cup \mathsf{sel}(G) & \models_{\mathcal{T}} \quad \square \quad \forall\ \mathsf{sel} \in \mathcal{S}\ ) \\ \Leftrightarrow & (\ \mathsf{sel}(\mathcal{K}[\mathsf{sel}(G)]) \cup \mathsf{sel}(G) & \models_{\mathcal{T}} \quad \square \quad \forall\ \mathsf{sel} \in \mathcal{S}\ ) \end{array}$$

In the last step we can use the same argument as in the first, since $\mathcal{K}[\mathsf{sel}(G)]$ is a set of ground clauses. □

Theorem 4 (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 with respect to LIG and a given selection function* sel*, then* G^0 *is* $(\mathcal{T} \cup \mathcal{K}^0)$*-satisfiable.*

Proof: Let $\mathcal{K}^0, G^0, \mathcal{K}^n, G^n, \mathsf{sel}$ as defined above. 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. We define a selection function sel' with $\mathsf{sel}'(\mathcal{K}^n \cup G^n) = \mathsf{sel}(\mathcal{K}^n \cup G^n)$, and which is defined on other clauses in $\mathcal{K}^0[G^0]$ in the following way: for every $C \in \mathcal{K}^0[G^0] \setminus G^n$, there is at least one $D \in \mathcal{K}^n$ such that D is a most specific generalization of C with respect to $\mathcal{K}^n \cup G^n$. If $D\sigma = C$, let $\mathsf{sel}'(C) = \mathsf{sel}(D)\sigma$ (i.e., select the instance of the literal which is selected by sel in D). If the most specific generalization is not unique, choose one.

Now, assume that $G^0 \models^{\mathcal{T} \cup \mathcal{K}^0} \square$, and define (corresponding to Lemma 3) a set of literals $\mathcal{L} = \mathsf{sel}'(\mathcal{K}^0[\mathsf{sel}'(G^0)]) \cup \mathsf{sel}'(G^0)$. Then, by Lemma 3, we have $\mathcal{L} \models_{\mathcal{T}} \square$. For every $L \in \mathcal{L}$, we have either $L \in \mathsf{sel}'(G^n)$ or there is a (non-ground) $K \in \mathsf{sel}'(\mathcal{K}^n)$ such that $K\sigma = L$, where K is selected in a clause D, L is selected in a clause C, and D is a most specific generalization of C wrt. $\mathcal{K}^n \cup G^n$. As $\mathsf{sel}'(G^n)$ is based on a $\mathcal{T}$-model of G^n and we assumed $\mathcal{L} \models_{\mathcal{T}} \square$, there must be at least one such $K \in \mathcal{L} \setminus \mathsf{sel}'(G^n)$.

Thus, there is a substitution σ instantiating literals from $\mathsf{sel}'(\mathcal{K}^n)$ to the set is $\mathcal{L}$, which means that there is a *LIG*-inference on $\mathcal{K}^n \cup G^n$ and sel. As K is selected in D, which is a most specific generalization of $C \in \mathcal{K}^0[G^0]$ wrt. $\mathcal{K}^n \cup G^n$, we must have $D\sigma \notin \mathcal{K}^n \cup G^n$, i.e. the inference is not redundant in $(\mathcal{K}^n, G^n)$ wrt. sel. This contradicts our assumption that $(\mathcal{K}^n, G^n)$ is saturated wrt. sel. $\square$

Theorem 5 (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 $\mathcal{K}[\mathsf{sel}(G)]$, where for every sel, $\mathcal{K}[\mathsf{sel}(G)] \subseteq \mathcal{K}[G]$. 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. $\square$

4 Chains of Local Theory Extensions

In Section 2.3, 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]. We will show that chains of extensions require a more sophisticated approach to incremental generation than one-time extensions, but also the possible benefit is much greater. 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 $\mathcal{K}_1, \ldots, \mathcal{K}_m$ such that $\mathcal{T}_j \subseteq \mathcal{T}_{j+1} = \mathcal{T}_j \cup \mathcal{K}_{j+1}$ is a local extension for $j = 0, \ldots, m-1$. Then, for every extension, we can use local reasoning to reduce a given ground goal G from theory $\mathcal{T}_j$ to theory $\mathcal{T}_{j-1}$:

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

If we define

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

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

We can see that 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 (starting with the reduction from $\mathcal{T}_m$ to $\mathcal{T}_{m-1}$). 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 treats all extensions at once, but still tries to keep generated instances on all levels to a minimum.

4.2 Chain-Local Instance Generation (LIG^*)

LIG^* 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.

Separation of axioms and ground goal. 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 $\mathcal{K}_j$.

Proof Procedure. Given sets of axioms $\mathcal{K}_1, \ldots, \mathcal{K}_m$ and a set of ground clauses G, LIG^* checks satisfiability of G modulo a repeatedly locally extended theory $\mathcal{T}_m = \mathcal{T}_0 \cup \mathcal{K}_1 \cup \ldots \cup \mathcal{K}_m$ by interleaving satisfiability checks (modulo $\mathcal{T}_0$) with instantiation of axioms from $\mathcal{K}_1 \cup \ldots \cup \mathcal{K}_m$. The following two steps are repeated until termination:

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

(2) Instance generation. Consider a selection function sel which is defined on G such that $M \models_{\mathcal{T}} \mathsf{sel}(G)$, and on the $\mathcal{K}_j$ as defined above. For the inference rule, define the following sets of literals:

$\mathcal{L}_{m+1} = \mathsf{sel}(G)$
$\mathcal{L}_j \quad = \mathsf{sel}(\mathcal{K}_j)[\bigcup_{j<k\leq m+1} \mathcal{L}_k]$ (for $1 \leq j \leq n$)

Then, instances are generated according to the following rule, where premises are divided (from left to right) into sets of clauses containing variants of clauses from $\mathcal{K}_1$ to $\mathcal{K}_m$, and finally clauses from G, and conclusions are divided into sets of clauses which are added to the respective sets (every $C\sigma$, where $C \in \mathcal{K}_j$, is added to $\mathcal{K}_j$ if $C\sigma$ is non-ground, and to G otherwise):

$$LIG^* \qquad \frac{F_1 \mid \ldots \mid F_m \mid G_0}{F'_1\sigma \mid \ldots \mid F'_m\sigma \mid F_{rest}\sigma}$$

where (for every $1 \leq j \leq m$):
(i) F_j consists of variants of clauses from $\mathcal{K}_j$, and $G_0 \subseteq G$
(ii) σ is a substitution such that $\mathsf{sel}(F_j)\sigma$ consists of ground literals,
(iii) $F'_j\sigma$ consists of generalizations of clauses in $\mathcal{K}_j[\bigcup_{j<k\leq m+1} \mathcal{L}_k]$,
(iv) $F'_j \subseteq F_j$ (such that $F'_j\sigma$ consists of axioms), $F_{rest} = \bigcup_{j=1}^{m}(F_j \setminus F'_j)$ (such that $F_{rest}\sigma$ consists of ground clauses), and
(v) $\mathsf{sel}(F_1)\sigma \wedge \ldots \wedge \mathsf{sel}(F_m)\sigma \wedge \mathsf{sel}(G_0) \models_{\mathcal{T}_0} \square$

Again, several variants of an axiom may be instantiated in one inference. The notions of redundancy and saturation apply 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. Unsatisfiable cores have to be computed over $\mathsf{sel}(F_1)\sigma \wedge \ldots \wedge \mathsf{sel}(F_m)\sigma \wedge \mathsf{sel}(G_0)$.

4.3 Examples: Applications of LIG^*

We show some applications of LIG^*, taken from previous work on verification of systems with a parametric number of components [9]. Due to space restrictions, we cannot go into the details of its behaviour, but it should be clear that LIG^* will in general generate fewer instances than standard local reasoning. How well LIG^* really behaves will have to be shown by an implementation.

Example 6. Consider as base theory the combination of linear arithmetic over reals and integers, $\mathbb{R} \cup \mathbb{Z}$. The system behaviour (for one step of the system) is specified by a set of update rules $\mathcal{K}_2$, and safety is proved by asserting a monotonicity condition $\mathcal{K}_1$ on function pos, and proving that the negation of that condition for function pos', together with some constraints on the constants $\mathsf{min}, \mathsf{max}, \mathsf{alarm}$ and n, is unsatisfiable in the combined theory $\mathbb{R} \cup \mathbb{Z} \cup \mathcal{K}_1 \cup \mathcal{K}_2$:

$$
\begin{aligned}
\mathcal{K}_1 = \{&x < y \;\rightarrow\; \mathsf{pos}(x) > \mathsf{pos}(y)\}\\
\mathcal{K}_2 = \{&(x = 0 \;\rightarrow\; \mathsf{pos}(x) + \mathsf{min} \leq_{\mathbb{R}} \mathsf{pos}'(x) \leq_{\mathbb{R}} \mathsf{pos}(x) + \mathsf{max}),\\
&(0 < x < \mathsf{n} \;\wedge\; \mathsf{pos}(x-1) >_{\mathbb{R}} 0 \;\wedge\; \mathsf{pos}(x-1) - \mathsf{pos}(x) \geq_{\mathbb{R}} \mathsf{l}_{\mathsf{alarm}}\\
&\;\rightarrow\; \mathsf{pos}(x) + \mathsf{min} \leq_{\mathbb{R}} \mathsf{pos}'(x) \leq_{\mathbb{R}} \mathsf{pos}(x) + \mathsf{max}),\\
&(0 < x < \mathsf{n} \;\wedge\; \mathsf{pos}(x-1) >_{\mathbb{R}} 0 \;\wedge\; \mathsf{pos}(x-1) - \mathsf{pos}(x) <_{\mathbb{R}} \mathsf{l}_{\mathsf{alarm}}\\
&\;\rightarrow\; \mathsf{pos}'(x) = \mathsf{pos}(x) + \mathsf{min}),\\
&(0 < x < \mathsf{n} \;\wedge\; \mathsf{pos}(x-1) \leq_{\mathbb{R}} 0 \;\rightarrow\; \mathsf{pos}'(x) = \mathsf{pos}(x))\}\\
G = \{&a < b,\;\; pos'(a) \leq pos'(b),\;\; n > 1,\;\; 0 \leq \mathsf{min}\\
&\mathsf{alarm} > 0,\;\; \mathsf{max} - \mathsf{min} < \mathsf{alarm}\}
\end{aligned}
$$

Standard local reasoning first computes the whole set $\mathcal{K}_2[G]$, consisting of 8 clauses (x instantiated by a or b in 4 axioms), then the set $\mathcal{K}_1[\mathcal{K}_2[G] \cup G]$, consisting of 16 clauses (all combinations of instantiating x, y by $a, a-1, b, b-1$). The SMT prover Yices reports that out of those 24 clauses, only 9 contribute to the proof of unsatisfiability.

In the optimal case, LIG^* would only produce exactly those instances that are needed to prove unsatisfiability: six instances from $\mathcal{K}_2$, and three instances from $\mathcal{K}_1$, but probably some more. The exact number of generated ground instances would depend on the implementation, and also on the SMT prover used.

Note also that G contains only constraints that are needed for proving unsatisfiability. If G contains irrelevant constraints, the gap between standard local reasoning and LIG^* becomes even bigger (as seen for LIG in Section 3.2).

Example 7. If the safety property of a system is not an inductive invariant, we may resort to bounded model checking (BMC). Consider a system where an update can be expressed as a local theory extension, and this theory 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

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

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

Since with standard local reasoning, the ground goal ($\neg\mathsf{safe}$ in this case) will blow up polynomially in each step, 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 $\mathcal{K}[G]$, with $\mathcal{L} = \bigcup_{1 \leq j \leq m+1} \mathcal{L}_j$, and with repeated application of Lemma 3.

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 $(\mathcal{K}_1^{i+1}, \ldots, \mathcal{K}_m^{i+1}, G^{i+1})$ is the result of an LIG^*-inference on $(\mathcal{K}_1^i, \ldots, \mathcal{K}_m^i, G^i)$.

Theorem 6 (Soundness) *Let* $(\mathcal{K}_1^0, \ldots, \mathcal{K}_m^0, G^0) \vdash \ldots \vdash (\mathcal{K}_1^n, \ldots, \mathcal{K}_m^n, G^n)$ *be an* LIG^**-derivation. If* $G^n \models_{\mathcal{T}_0} \square$*, then* $G^0 \models_{\mathcal{T}_m} \square$*.*

Theorem 7 (Completeness) *Let* $(\mathcal{K}_1^0, \ldots, \mathcal{K}_m^0, G^0) \vdash \ldots \vdash (\mathcal{K}_1^n, \ldots, \mathcal{K}_m^n, G^n)$ *be an* LIG^**-derivation. If* G^n *is* $\mathcal{T}_0$*-satisfiable and* $(\mathcal{K}_1^n, \ldots, \mathcal{K}_m^n, G^n)$ *is saturated with respect to* LIG^* *and a given selection function* sel*, then* G^0 *is* $\mathcal{T}_m$*-satisfiable.*

Theorem 8 (Termination) *For any input* $(\mathcal{K}_1^0, \ldots, \mathcal{K}_m^0, G^0)$ *(with the* $\mathcal{K}_i^0$ *and* G^0 *as defined above),* LIG^* *terminates after a finite number of inferences.*

5 Conclusions and Future Work

We have introduced two procedures which allow incremental generation of the instances needed for reasoning in local theory extensions and chains of local extensions, respectively. We have shown that, for the fragments we consider, both methods give decision procedures and can terminate without generating the full set of instances needed for standard local reasoning. We conjecture that our procedures are (in average) more space-efficient than the standard local reasoning approach. Time-efficiency will have to be evaluated using an implementation, which is under way.

The approach we have presented resembles in some parts quantifier handling by E-matching in SMT procedures. This holds specifically for the matching of extension terms containing variables in watched literals to ground terms appearing in the model. However, in contrast to SMT we have an additional semantic condition for instantiation. Furthermore, matching in SMT solvers is usually modulo equality, while pure syntactical matching is enough for our completeness result. It would be interesting to look at a variant of our approach which drops the semantic condition (and maybe generates all possible instances at once), and to check whether we could benefit somehow from using E-matching instead of syntactic matching.

Another approach of generating instances of axioms 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 ground terms which are in the candidate model. A variant between the two approaches would thus be to generate only instances that refute the current model, but restrict the search for substitutions to terms appearing in the model.

An obvious extension of our work would be to consider other notions of locality, like stable locality [11] or Ψ-locality [8]. This would enlarge the fragment of local theory extensions we can handle.

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). I also thank Viorica Sofronie-Stokkermans for our fruitful discussions, and the anonymous reviewers for helpful comments and suggestions.

References

1. R. Bruttomesso, A. Cimatti, A. Franzen, A. Griggio, A. Santuari and R. Sebastiani. To Ackermann-ize or Not to Ackermann-ize? On Efficiently Handling Uninterpreted Function Symbols in SMT(EUF $\cup \mathcal{T}$). In *Logic for Programming, Artificial Intelligence, and Reasoning, 13th Int. Conf. (LPAR 2006)*, LNCS 4246, pp. 557–571. 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)*, to appear.
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 and K. Korovin. New Directions in Instantiation-Based Theorem Proving. In *Proc. 18th IEEE Symposium on Logic in Computer Science,(LICS'03)*, pp. 55–64. Springer, 2003.
5. H. Ganzinger and K. Korovin. Integrating equational reasoning into instantiation-based theorem proving. In *Computer Science Logic (CSL'04)*, LNCS 3210, pp. 71–84. Springer, 2004.
6. 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.
7. 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.
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. 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.
11. 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.
12. V. Sofronie-Stokkermans. Hierarchical and Modular Reasoning in Complex Theories: The Case of Local Theory Extensions. In *Frontiers of Combining Systems, 6th International Symposium (FroCoS 2007)*, LNCS 4720, pp. 47–71. Springer, 2007.