Hierarchic Decision Procedures for Verification

Swen Jacobs

Dissertation

zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Saarbrücken
2009

Tag des Kolloquiums:	29. Januar 2010
Dekan:	Prof. Dr. Joachim Weickert
Prüfungsausschuss	
Vorsitzender:	Prof. Dr. Reinhard Wilhelm
Berichterstattende:	PD Dr. Viorica Sofronie-Stokkermans
	Prof. Dr. Bernd Finkbeiner
	Prof. Dr. Viktor Kuncak
Akademischer Mitarbeiter:	Dr. Thomas Hillenbrand

Abstract

Information-handling systems are becoming ever more complex. They may be pure hardware or software systems, or complex systems of hardware and software that act in a real-world environment.

Verification is a method to ensure that systems behave in the expected way, which is a necessity for safety-critical applications like automatic railway control. The size of such systems makes manual verification impossible. Therefore, we need automatic or computer-aided verification procedures.

Automated reasoning is already widely used in the analysis and verification of systems. For a restricted class of systems, the resulting verification problems are inherently finite and can be solved efficiently. For complex systems, such finiteness cannot be expected. To express and prove properties of these systems, we need a formal language and reasoners that can deal with universal quantification, arithmetic expressions and unbounded data structures at the same time.

Thus, in recent years there has been new interest in the handling of first-order formulas modulo a given background theory. The problem is known to be undecidable in general, and research focuses mostly on methods that solve many problem instances quickly, but sacrifice completeness. We take a different approach and focus on instances of this problem that we can show to be decidable. In this way we can solve the resulting problems efficiently and guarantee termination.

This work is based on research by Sofronie-Stokkermans on local theory extensions and on work by Ganzinger and Korovin on instantiation-based first-order theorem proving. We extend the existing work on local theory extensions, giving new examples of axioms which satisfy a locality condition and using ideas from instantiation-based first-order theorem proving to make local reasoning more efficient. Furthermore, we show that local theory extensions allow us to decide certain verification problems for parameterized systems and develop increasingly complex system models of an automatic train controller on which we demonstrate how to use local reasoning to verify safety properties of such systems.

Kurzzusammenfassung

Informationsverarbeitende Systeme werden ständig komplexer. Dies können reine Hardware- oder Softwaresysteme sein, oder komplexe Systeme von Hardware und Software, die mit ihrer physikalischen Umgebung interagieren.

Mittels Verifikation kann sichergestellt werden, dass ein System sich in der erwarteten Weise verhält. Bei sicherheitskritischen Systemen, z.B. automatischen Zugsteuerungssystemen, ist dies unumgänglich. Die Größe solcher Systeme macht es unmöglich, ihr Verhalten von Hand zu verifizieren. Deshalb benötigen wir automatische oder computergestützte Verifikationsmethoden.

Bei der Analyse und Verifikation von Systemen ist automatisches Beweisen bereits weit verbreitet. Für eine eingeschränkte Klasse von Systemen sind die auftretenden Verifikationsprobleme von Natur aus endlich and können effizient gelöst werden. Für komplexe Systeme kann eine solche Endlichkeit nicht angenommen werden. Um Eigenschaften solcher Systeme ausdrücken und beweisen zu können, brauchen wir eine formale Sprache und Beweismethoden, die mit universeller Quantifizierung, arithmetischen Ausdrücken und unbeschränkten Datentypen gleichzeitig umgehen können.

Deshalb gab es in den letzten Jahren ein neues Interesse an Methoden, die universell quantifizierte Probleme in solchen Hintergrundtheorien lösen können. Es ist bekannt, dass solche Probleme im Allgemeinen unentscheidbar sind, und die Forschung konzentriert sich auf Methoden, die unter Verzicht auf Vollständigkeit möglichst viele Probleme schnell lösen können. Wir verfolgen einen anderen Ansatz und konzentrieren uns auf Problemklassen, deren Entscheidbarkeit wir zeigen können. Dadurch können wir diese Probleme effizient lösen und gleichzeitig das Terminieren der Prozedur garantieren.

Diese Arbeit basiert auf der Forschungsarbeit von Sofronie-Stokkermans an lokalen Theorieerweiterungen, sowie der Arbeit von Ganzinger und Korovin an instanziierungs-basierten Methoden zum Theorembeweisen in Prädikatenlogik erster Ordnung. Wir führen die Arbeit an lokalen Theorieerweiterungen fort, indem wir neue Beispiele von Axiomen geben, die eine Lokalitätseigenschaft erfüllen, und benutzen Ideen aus instanziierungs-basierten Methoden zum Theorembeweisen in Prädikatenlogik, um lokales Beweisen effizienter zu machen. Weiterhin zeigen wir, dass lokale Theorieerweiterungen es uns ermöglichen, bestimmte Verifikationsprobleme für parametrisierte Systeme zu entscheiden und entwickeln eine Reihe komplexer werdender Modelle eines automatischen Zugsteuerungssystems an denen wir demonstrieren, wie man mittels lokalen Beweisens Sicherheitseigenschaften solcher Systeme verifizieren kann.

Acknowledgements

First and foremost, I thank my advisor Viorica Sofronie-Stokkermans for the good cooperation and the time and work she invested in teaching and guiding me, in collaborating with me and in making this thesis what it is today. Most of the results presented here are joint work with her, and none of them would have been possible without her guidance and support.

I thank Uwe Waldmann, who has guided me in the first years of my research and has always been a great help.

I thank Leonardo de Moura for help on integrating the Z3 SMT solver into my implementation, and for interesting discussions.

I am grateful to Reinhard Wilhelm, Bernd Finkbeiner, Viktor Kuncak and Thomas Hillenbrand for joining my thesis committee, as well as for interesting discussions and helpful comments.

Many thanks also to my fellow students Willem Hagemann, Matthias Horbach and Carsten Ihlemann for fruitful discussions and especially to Matthias for careful proofreading of drafts of this thesis.

During my graduate studies, I have been supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS). I am grateful to the DFG for the financial support, and to my colleagues in AVACS for making it an interesting and inspiring research project.

I have been very lucky to work at Max-Planck-Institut für Informatik (MPII). Thanks to all my colleagues at MPII who have made my stay there worthwhile. I fear that I will sorely miss the relaxed and at the same time inspiring atmosphere of "our" institute.

Finally, I thank my family and friends, who gave me support and have been there for me through the last five years. I am very happy to have all of you in my life.

Contents

List of Figures

Chapter 1

Introduction

Information-handling systems have become ubiquitous in our society and are becoming more complex day by day. They may be pure hardware or software systems, or embedded systems of hardware and software that act in a real-world environment, resulting in complex systems of several components. Verification is a method to ensure that systems behave in the expected way, which is a necessity for safety-critical applications such as computer-controlled medical equipment, automatic transportation control for cars, trains and aircraft, as well as computer-controlled processes in power plants, like emergency shutdown measures in nuclear reactors. For most of these systems, their size makes it impossible to verify their correctness by hand. Therefore, we need automatic or computer-aided verification procedures.

A back-end of many automatic and computer-aided verification methods is automated reasoning, which comes in different levels of complexity:

For pure hardware systems, propositional satisfiability (SAT) solving is sufficient because of their finite structure. There are dedicated, technically mature SAT solvers like Chaff [47] or MiniSAT [15], which can decide such problems very efficiently.

Going beyond the expressiveness of SAT, we have on the one hand software that uses unbounded data types, but is otherwise finite, e.g. in the number of program steps. Given procedures that can decide ground satisfiability problems in the theory of such data types, verification problems for these systems can often be encoded as ground satisfiability modulo theories (SMT) problems. In this case, ground SMT solvers like CVC [5], Yices [14] or Z3 [11] decide the given problems.

On the other hand, there is software that can be abstracted from concrete data types, but has an inherent infiniteness, e.g. because a loop may be repeated for an unbounded number of times. Such problems can be encoded in general first-order logic (FOL), which allows to make statements over infinitely many elements in the same formula by universal quantification. In general, FOL with universal quantification is only semi-decidable, i.e. termination of any automatic procedure that tries to prove a conjecture can only be guaranteed if the conjecture can be proved — otherwise, the procedure may run forever. Formulas in FOL can efficiently be handled by first-order theorem provers like E [55], SPASS [67] and Vampire [54].

Finally, some software systems and most complex systems require to com-

bine reasoning modulo theories and quantification over infinitely many elements. None of the methods above is sufficient for this kind of problems. Handling universal quantifiers when reasoning modulo theories is in general even more difficult than for pure FOL and not even semi-decidable. Consequently, most of the research in this direction focuses on methods that either consider rather uniform background theories (as opposed to complex combinations of several background theories) that allow quantifier elimination, like linear arithmetic [68], fragments of the theory of arrays [28, 9] or sets with cardinality constraints [43], or on heuristics for quantifier instantiation, which work in complex theories, but sacrifice completeness [12, 26]. There has also been some work on integrating theory reasoning with first-order theorem proving methods [62, 23, 53], but in general these also sacrifice completeness.

We take a different approach and focus on instances of the problem that we can show to be decidable, even in the presence of complex background theories. We use the notion of *local theory extensions*, introduced by Sofronie-Stokkermans [56], which are extensions of a background theory with additional axioms that allow for quantifier elimination and reduction of reasoning to the background theory. In this way, we can guarantee termination of our automated reasoning procedures even for complex background theories, resulting in either a proof or a counterexample for any possible conjecture.

We will show that such local theory extensions allow us to reason about many useful properties of mathematical functions, data structures and parameterized systems. Moreover, local reasoning gives us decision procedures for these problems, where other state-of-the-art approaches resort to heuristics or cannot be used at all.

Besides the research by Sofronie-Stokkermans on local theory extensions [56, 59], our work is also based on work by Ganzinger and Korovin on instantiation-based first-order theorem proving [21, 22, 23]. In this dissertation, we give new results in the framework of local theory extensions, bring these two strings of research together and show applications of the resulting methods in the verification of parameterized systems.

1.1 Contributions

This dissertation advances the state of the art by improving reasoning in local theory extensions in several ways:

- We extend the applicability of local reasoning by identifying new local theory extensions.
- We combine the existing local reasoning approach with ideas from first-order instance generation, providing a more efficient method to solve the resulting satisfiability problems incrementally.
- We provide prototype implementations of both the standard approach to local reasoning and the new, incremental approach and give a comparison of their behavior.
- We show applications of our results in the verification of complex systems with a parametric number of components, as an example for verification tasks that are beyond the state-of-the-art provers mentioned above.

We describe the contributions in more detail:

1.1.1 Identifying New Local Theory Extensions

We identify several theory extensions, represented as sets of axioms, that are local with respect to a given base theory. The locality property allows us to treat problems in the extended theory efficiently by finite instantiation of the axioms and reduction to the base theory.

The new local theory extensions include strictly monotone functions, functions that are strictly bounded by other function terms, sets of guarded boundedness properties with mutually exclusive guards and several combinations of (strict) monotonicity and (strict) boundedness conditions. Furthermore, we will show locality properties for a notion of quasi-monotone functions, recursive data structures and functions modeling cardinality of sets.

1.1.2 An Incremental Approach to Local Reasoning

The standard approach to local reasoning instantiates the given axioms eagerly to a set of instances that are known to be equisatisfiable to the original first-order problem. We apply ideas from resolution-based instance generation (developed by Ganzinger and Korovin [21, 22, 23]) to obtain an incremental approach that generates only a few ground instances at a time and interleaves instantiation with satisfiability checks on the ground level. These checks can be done by a black-box SMT solver. If the ground problem is satisfiable, we obtain a candidate model that can be used to guide further instantiation steps.

The benefit of this approach is that, both in the satisfiable and unsatisfiable case, it can terminate without generating all instances that would be needed for the reduction to the base theory with the standard approach to local reasoning. We prove that the incremental approach remains sound and complete.

1.1.3 Implementation of Both Approaches

As a proof of concept, we have implemented the standard approach to local reasoning as well as the new, incremental approach. We compare the standard approach to two different strategies for incremental instance generation with respect to their behavior, focusing on space and time efficiency. We show that the incremental approach is more efficient for a set of crafted examples as well as for a set of verification benchmarks taken from the SMT benchmark library SMT-LIB.

1.1.4 Applications in Verification

We show that locality of theory extensions allows us to decide invariant checking and bounded model checking problems for a certain class of parameterized systems. We apply our results to the verification of a case study taken from the European Train Control System (ETCS) standard. We introduce successively more complex models that can all be verified by using reasoning in local theory extensions. The models we consider feature a parametric number of components, which means they cannot be treated by the standard approaches.

1.2 Outline

In Chapter 2, we introduce the notion of local theory extensions and local reasoning, and give some examples of local theory extensions that have been identified before. We introduce a number of new theory extensions and prove their locality in Chapter 3. Chapter 4 introduces our incremental approach to local reasoning. We evaluate efficiency of an implementation of the new approach, compared to the standard approach to local reasoning. In Chapter 5, we show that locality results allow us to decide verification problems for certain parameterized systems, and present in detail an application of our theoretical results to the ETCS case study. Finally, we draw conclusions and present possible directions for future work in Chapter 6.

Chapter 2

Hierarchic Reasoning in Local Theory Extensions

Research on *local theory extensions* is based on the notion of *local theories* introduced by Givan and McAllester [30, 31], and results of Basin and Ganzinger [7, 20] studying these local theories. *Hierarchic reasoning in local theory extensions*, or in short: *local reasoning*, has been introduced by Sofronie-Stokkermans [56], based on work of Ganzinger, Sofronie-Stokkermans and Waldmann [25].

This chapter gives an overview of the previous results on local reasoning. After defining the necessary concepts of sorted first-order logic (with equality), we will first introduce the notion of *local theory extensions*, which are a generalization of local theories. Then we present the method of *local reasoning*, which allows us to decide satisfiability problems in local theory extensions. After showing how extensions with a locality property can be identified, we give some examples of local theory extensions. Finally, we introduce a refinement of the local reasoning approach which not only considers a single local extension, but a *chain of extensions*.

2.1 Preliminaries

Signatures and Formulas

Definition 2.1 (Countable)**.** We say that a set is *countable* if it is either finite or countably infinite.

Definition 2.2 (Signature)**.** A *signature* Π is a tuple $(S, \Sigma, \mathsf{Pred})$, where:

- S is a countable set of sorts.
- Σ is a countable set of function symbols, each with an *arity* $n \geq 0$ and a *type* $s_1 \times \ldots \times s_n \to s_{n+1}$, with $s_i \in S$ for all i.[1] Function symbols with arity 0 are called *constant symbols*.
- Pred is a countable set of predicate symbols, each with a type $s_1 \times \ldots \times s_n$, $0 \leq n$, $s_i \in S$ for all i.

[1] These notions of arity and type for many-sorted logic are according to Zhang et al. [69].

Definition 2.3 (Signature extension)**.** Given two signatures $\Pi = (S, \Sigma, \mathsf{Pred})$ and $\Pi' = (S', \Sigma', \mathsf{Pred}')$, we say that Π is a *signature extension* of Π' if $S \subseteq S'$, $\Sigma \subseteq \Sigma'$ and $\mathsf{Pred} \subseteq \mathsf{Pred}'$. In this case we write $\Pi \subseteq \Pi'$.

Definition 2.4 (Variables)**.** A *variable* is a symbol with an associated sort. We assume that for a given signature $\Pi = (S, \Sigma, \mathsf{Pred})$ we have a set $X = \bigcup_{s \in S} X_s$ of Π-*variables*, where each X_s consists of a countably infinite number of variables of sort s.

Definition 2.5 (Term, ground term)**.** For a given signature $\Pi = (S, \Sigma, \mathsf{Pred})$ and a set of Π-variables $X = \bigcup_{s \in S} X_s$, a Π-*term* (over X) is defined recursively:

- every $x \in X_s$ is a Π-term of sort s,
- every constant symbol $c \in \Sigma$ of type $\to s_1$ is a term of sort s_1, and
- if $f \in \Sigma$ is a function symbol of type $s_1 \times \ldots \times s_n \to s_{n+1}$ and $t_1, \ldots, t_n$ are Π-terms of sort $s_1, \ldots, s_n$, respectively, then $f(t_1, \ldots, t_n)$ is a Π-term of sort s_{n+1}.

A term is *ground* if it does not contain variables. The set of all Π-terms over X is denoted by $T_\Sigma(X)$, and the set of ground Π-terms by T_Σ.

Definition 2.6 (Subterm)**.** The *subterms* of a term are defined recursively:

- every term is a subterm of itself, and
- the subterms of $f(t_1, \ldots, t_n)$ include also the subterms of the t_i.

A subterm is *ground* if it does not contain variables.

Definition 2.7 (Atom)**.** For a given signature $\Pi = (S, \Sigma, \mathsf{Pred})$, a Π-*atom* is either

- $t_1 = t_2$, where t_1 and t_2 are Π-terms of the same sort, or
- $P(t_1, \ldots, t_n)$, where $P \in \mathsf{Pred}$ with type $s_1 \times \ldots \times s_n$, and each t_i is a Π-term of sort s_i.

Definition 2.8 (Literal)**.** For a given signature $\Pi = (S, \Sigma, \mathsf{Pred})$, a Π-*literal* is either a Π-atom A or its negation $\neg A$. If A is $t_1 = t_2$ for Π-terms t_1 and t_2, we write $\neg A$ also as $t_1 \neq t_2$.

Definition 2.9 (Formula, sentence)**.** For a given signature $\Pi = (S, \Sigma, \mathsf{Pred})$, a Π-*formula* is defined recursively as

- a Π-literal,
- $\neg F$, for a Π-formula F,
- $F_1 \wedge F_2$, for Π-formulas F_1 and F_2,
- $F_1 \vee F_2$, for Π-formulas F_1 and F_2,
- $F_1 \to F_2$, for Π-formulas F_1 and F_2,
- $\forall\, x\!:\!s.\ F$, for a Π-formula F and a variable x of sort $s \in S$, or

- $\exists\, x\!:\!s.\ F$, for a Π-formula F and a variable x of sort $s \in S$.

A Π-formula is called a Π-*sentence* if every variable is bound by a quantifier. It is *prenex* if it consists of a (possibly empty) sequence of quantifiers, followed by a quantifier-free formula. It is *universal* if it is prenex and does not contain existential quantifiers. It is a $\forall\exists$ *formula* if it is prenex and no universal quantifier appears after an existential quantifier.

Note that whenever Π and Π' are signatures with $\Pi \subseteq \Pi'$, any Π-formula is also a Π'-formula.

Definition 2.10 (Clause, ground clause, Horn clause)**.** For a given signature Π, a Π-*clause* C is a disjunction of Π-literals, where all variables in C are implicitly universally quantified. We call C a *ground clause* if it does not contain variables, a *non-ground clause* otherwise. A clause is *Horn* if it contains at most one positive literal.

Definition 2.11 (Substitution)**.** For a given signature $\Pi = (S, \Sigma, \mathsf{Pred})$, a Π-*substitution* is a function mapping Π-variables of sort $s \in S$ to Π-terms of sort s. For a Π-variable x, we write $x\sigma$ to denote the result of applying σ to x. For terms t and formulas F, we write $t\sigma$ and $F\sigma$, respectively, to denote the result of applying σ simultaneously to all variables in t or F.

Definition 2.12 (Instance)**.** If C is a Π-clause and σ a Π-substitution, then $C\sigma$ is a Π-*instance* of C.

Definition 2.13 ($\mathsf{st}(F)$)**.** For a formula F, let $\mathsf{st}(F)$ be the set of ground subterms appearing in F.

In the following, whenever Π is clear from the context or we need not specifically refer to it, we omit the prefix Π from all notions defined above. We use x, y, i, j to denote variables, a, b, c, d for constants, f as a non-constant function symbol and t for terms. We write $t[x_1, \ldots, x_n]$ to denote that no other variables than $x_1, \ldots, x_n$ are subterms of t. A sequence of variables $x_1, \ldots, x_n$ will sometimes be abbreviated as $\overline{x}$, and a sequence of terms $t_1, \ldots, t_n$ as $\overline{t}$.

L denotes literals, C and D clauses, and F formulas. The empty clause is denoted by $\Box$. Sets of formulas represent the conjunction of their elements. We denote sets of clauses by $\mathcal{K}$, sets of ground clauses by G.

Structures and Models

Definition 2.14 (Structure)**.** For a signature $\Pi = (S, \Sigma, \mathsf{Pred})$, a Π-*structure* is a map M that assigns to every sort $s \in S$ a non-empty set of elements s_M, to every $f \in \Sigma$ with type $s_1 \times \ldots \times s_n \to s_{n+1}$ a total function $f_M : s_{1_M} \times \ldots \times s_{n_M} \to s_{n+1_M}$, and to every $P \in \mathsf{Pred}$ with type $s_1 \times \ldots \times s_n$ a set $P_M \subseteq s_{1_M} \times \ldots \times s_{n_M}$.

Definition 2.15 (Variable assignment)**.** For a given Π-structure M and a set of Π-variables $X = \bigcup_{s \in S} X_s$, a Π-*variable assignment* is a map β that assigns to every $x \in X_s$ an element of s_M. For any Π-term t, we denote by $\beta(t)$ the result of applying β recursively to its subterms.

For a given Π-variable assignment β, a Π-variable x of sort s and $e \in s_M$, let $\beta[x \mapsto e]$ be the Π-variable assignment that maps x to e, and otherwise behaves like β.

Definition 2.16 (Satisfaction, model)**.** For a signature $\Pi = (S, \Sigma, \mathsf{Pred})$ and set of Π-variables X, we define when a pair (M, β) of a Π-structure M and a Π-variable assignment β *satisfies* a Π-literal L, written $(M, \beta) \models L$:

- $(M, \beta) \models P(t_1, \ldots, t_m)$ iff $(\beta(t_1), \ldots, \beta(t_m)) \in P_M$.
- $(M, \beta) \models \neg P(t_1, \ldots, t_m)$ iff $(\beta(t_1), \ldots, \beta(t_m)) \notin P_M$.
- $(M, \beta) \models t_1 = t_2$ iff $\beta(t_1) = \beta(t_2)$.
- $(M, \beta) \models t_1 \neq t_2$ iff $\beta(t_1) \neq \beta(t_2)$.

Then, satisfaction of formulas is defined recursively:

- $(M, \beta) \models \neg F$ iff not $(M, \beta) \models F$
- $(M, \beta) \models F_1 \wedge F_2$ iff $(M, \beta) \models F_1$ and $(M, \beta) \models F_2$,
- $(M, \beta) \models F_1 \vee F_2$ iff $(M, \beta) \models F_1$ or $(M, \beta) \models F_2$,
- $(M, \beta) \models F_1 \rightarrow F_2$ iff $(M, \beta) \models \neg F_1$ or $(M, \beta) \models F_2$
- $(M, \beta) \models \forall\, x.\ F$ iff $(M, \beta[x \mapsto e]) \models F$ for every $e \in s_M$ (where $x \in X_s$),
- $(M, \beta) \models \exists\, x.\ F$ iff $(M, \beta[x \mapsto e]) \models F$ for some $e \in s_M$ (where $x \in X_s$).

We say that a Π-structure M *satisfies* a Π-formula F if $(M, \beta) \models F$ for every Π-variable assignment β. In this case, we write $M \models F$ and call M a Π-*model* of F. For a set of Π-formulas $\mathcal{F}$, we write $M \models \mathcal{F}$ and call M a Π-*model* of $\mathcal{F}$ if $M \models F$ for every $F \in \mathcal{F}$.

Definition 2.17 (Consequence)**.** A Π-formula F is a Π-*consequence* of a set of Π-formulas $\mathcal{T}$, written $\mathcal{T} \models F$, if F is satisfied by every Π-model of $\mathcal{T}$.

Definition 2.18 (Unsatisfiable)**.** A set of Π-formulas $\mathcal{T}$ is *unsatisfiable* if there is no Π-structure[2] that satisfies $\mathcal{T}$. In this case we write $\mathcal{T} \models \square$.

Theories and Reasoning Modulo Theories

Unlike some of the previous work on local theory extensions which allows theories to be either sets of formulas or sets of structures, we restrict ourselves to theories which are defined as certain sets of formulas. The following definition is according to van Dalen [66].

Definition 2.19 (Theory)**.** A Π-*theory* is a set $\mathcal{T}$ of Π-sentences that is closed under consequences, i.e. $\mathcal{T} \models F$ only if $F \in \mathcal{T}$, for every Π-formula F.

Definition 2.20 (Axiomatization, axiom)**.** A set of Π-sentences $\mathcal{T}'$ is an *axiomatization* of a Π-theory $\mathcal{T}$ if, for all Π-formulas F, $\mathcal{T}' \models F$ iff $F \in \mathcal{T}$. The elements of $\mathcal{T}'$ are called *axioms* of $\mathcal{T}$.

Convention. For an axiomatization $\mathcal{T}'$ of a theory $\mathcal{T}$ and a formula F we have $\mathcal{T} \models F$ if and only if $\mathcal{T}' \models F$. Because of this equivalence wrt. consequences, we will use axiomatizations to represent theories. Whenever a set of axioms is used in place of a theory, the theory itself can be obtained from its axiomatization by considering its closure under consequences.

[2] Note that if there is no Π-structure that satisfies $\mathcal{T}$, then there can also be no Π'-structures, $\Pi \subseteq \Pi'$, that satisfy $\mathcal{T}$.

Definition 2.21 (Universal theory)**.** A Π-theory is *universal* if it can be axiomatized by a set of universal Π-sentences.

Definition 2.22 (Empty theory)**.** The *empty theory* is the theory that is axiomatized by the empty set.

Definition 2.23 (Decidable theories, decision procedures)**.** A Π-theory $\mathcal{T}$ is *decidable* if there is a procedure that decides for every Π-formula F whether $\mathcal{T} \models F$. We say that the *universal fragment of* $\mathcal{T}$ *is decidable* if this holds for all universal Π-formulas F. Similarly, the $\forall\exists$ *fragment of* $\mathcal{T}$ *is decidable* if this holds for all $\forall\exists$ Π-formulas F.

A procedure that decides (a fragment of) a theory $\mathcal{T}$ is called a *decision procedure* for (this fragment of) $\mathcal{T}$.

Definition 2.24 (Arithmetic theories)**.** In our examples and applications, we will often refer to the theories of *Presburger arithmetic* [52] $\mathcal{T}_{\mathbb{N}}$, *linear integer arithmetic* $\mathcal{T}_{\mathbb{Z}}$, *linear rational arithmetic* $\mathcal{T}_{\mathbb{Q}}$ and *non-linear real arithmetic* $\mathcal{T}_{\mathbb{R}}$ (also known as real closed fields or elementary algebra [65]). Axiomatizations of these theories are well-known, see e.g. Bradley and Manna [8]. All of these theories are decidable.

Remark. As we can consider Π-formulas also as Π′-formulas, for any $\Pi \subseteq \Pi'$, we can also consider Π-theories as Π′-theories. Then, decidability of a Π-theory $\mathcal{T}$ depends not only on $\mathcal{T}$ (as a set of sentences) itself, but also on Π. E.g., $\mathcal{T}_{\mathbb{Z}}$ is decidable when we consider it as a theory over its standard signature $\Pi_{\mathbb{Z}}$ (with a function symbol for addition, constant symbols for every integer and function symbols for multiplication with integer constants), but not when we consider it as a theory over the extended signature $\Pi' \supseteq \Pi_{\mathbb{Z}}$ that additionally contains a unary function symbol f. For $\mathcal{T}_{\mathbb{Z}}$ as a Π′-theory, only the universal fragment is decidable.[3]

Definition 2.25 (Theory extension, extension symbols, extension terms)**.** Consider signatures $\Pi_0 = (S, \Sigma_0, \mathsf{Pred})$ and $\Pi_1 = (S, \Sigma_0 \cup \Sigma_1, \mathsf{Pred})$, with $\Sigma_0 \cap \Sigma_1 = \varnothing$. A Π_1-theory $\mathcal{T}_1$ is a *theory extension* of a Π_0-theory $\mathcal{T}_0$ if $\mathcal{T}_0 \subseteq \mathcal{T}_1$. As an abbreviation, we say that $\mathcal{T}_0 \subseteq \mathcal{T}_1$ is a theory extension.

Function symbols in Σ_1 are called *extension symbols*, terms $f(t_1, \ldots, t_n)$ with $f \in \Sigma_1$ are called *extension terms*.

Conventions. According to our convention that we represent theories by their axiomatizations, we will also consider extensions of axiomatizations with additional axioms, and call them theory extensions.

In the following, we will only consider theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ such that $\mathcal{T}_0$ is (an axiomatization of) a Π_0-theory, $\mathcal{K}$ is a set of Π_1-clauses and $\Sigma_1 \neq \varnothing$.

Furthermore, whenever we write $\mathcal{T} \models F$, we assume that F is in the signature of $\mathcal{T}$, except for additional constant symbols. Similarly, if $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is a theory extension and we write $\mathcal{T} \cup \mathcal{K} \models F$ (or $\mathcal{T} \cup \mathcal{K} \cup F \models \square$), we assume that F is in the signature of $\mathcal{T} \cup \mathcal{K}$, except for additional constant symbols.

[3] We will see in Section 2.5 that decidability of the universal fragment in fact follows from previous results on local theory extensions.

2.2 Locality of Theories and Theory Extensions

2.2.1 Local Theories

Work on local theories goes back to Givan and McAllester [31, 30], who first looked into locality of inference relations defined as sets of Horn clauses. This work was continued by Ganzinger [20], who defined the notion of local theories (which are the same as local inference relations) and generalized it to stably local theories.[4]

Definition 2.26 ($\mathcal{K}_\Theta$)**.** For a set of clauses $\mathcal{K}$ and a set of ground terms Θ, define

$$\mathcal{K}_\Theta = \{\ C\sigma\ \mid C \in \mathcal{K}, C\sigma \text{ ground and } \mathsf{st}(C\sigma) \subseteq \Theta\ \}.$$

Definition 2.27 (Local theory [20])**.** A *local theory* is a set of Horn clauses $\mathcal{K}$ such that for any ground Horn clause C we have $\mathcal{K} \models C$ only if $\mathcal{K}_{\mathsf{st}(\mathcal{K}\cup C)} \models C$.

That is, to decide whether C is a consequence of $\mathcal{K}$, it is enough to generate the finite set of ground instances $\mathcal{K}_{\mathsf{st}(\mathcal{K}\cup C)}$ and to check whether C is a consequence of these instances.

Definition 2.28 ($\mathcal{K}^\Theta$)**.** For a set of clauses $\mathcal{K}$ and a set of ground terms Θ, define

$$\mathcal{K}^\Theta = \{\ C\sigma\ \mid C \in \mathcal{K} \text{ and } x\sigma \in \Theta \text{ for every variable } x\ \}.$$

Definition 2.29 (Stably local theory [20])**.** A *stably local theory* is a set of Horn clauses $\mathcal{K}$ such that for any ground Horn clause C we have $\mathcal{K} \models C$ only if $\mathcal{K}^{\mathsf{st}(\mathcal{K}\cup C)} \models C$.

2.2.2 Local Theory Extensions

The notion of local theories has been generalized to local theory extensions by Sofronie-Stokkermans [56]. With the following definitions, a local theory can be defined as a local theory extension of the empty theory.

Definition 2.30 ($\mathcal{K}[G]$)**.** Let $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ be a theory extension. For a set G of ground clauses, let

$$\begin{aligned}\mathcal{K}[G] = \{\ C\sigma\ \mid\ & C \in \mathcal{K} \text{ and } \sigma \text{ is such that}\\ & (1.)\ \text{for each extension term } f(t) \text{ in } C,\ f(t)\sigma \in \mathsf{st}(\mathcal{K} \cup G)\\ & (2.)\ x\sigma = x, \text{ if } x \text{ does not occur in an extension term }\}.\end{aligned}$$

That is, $\mathcal{K}[G]$ is the set of instances that results from matching extension terms in $\mathcal{K}$ to ground extension terms in $\mathcal{K} \cup G$, and not instantiating other variables. Note that if all variables in $\mathcal{K}$ appear in extension terms, then $\mathcal{K}[G] = \mathcal{K}_{\mathsf{st}(K\cup G)}$.

Definition 2.31 (Local theory extension [56])**.** A theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is *local* if it satisfies condition (Loc):[5]

[4] There is also work on locality by Basin and Ganzinger [7], who looked into sets of (not necessarily Horn) clauses and defined the notion of order locality wrt. a given term ordering. We will not consider order locality in this work.

[5] In the literature, this notion of locality is sometimes called (Loc^f). There, the superscript f refers to the restriction to finite sets of ground clauses G, while the general notion also allows for infinite sets. For the purposes of this work, finite sets will always be sufficient.

(Loc) For every finite set G of ground clauses,
$\mathcal{T}_0 \cup \mathcal{K} \cup G \models \Box \Leftrightarrow \mathcal{T}_0 \cup \mathcal{K}[G] \cup G \models \Box$.

Note that $\mathcal{K}[G] \cup G$ is a set of clauses in the signature of the theory extension, so a decision procedure for $\mathcal{T}_0$ (in its standard signature) is not sufficient for deciding satisfiability of $\mathcal{T}_0 \cup \mathcal{K}[G] \cup G$. The same holds for the satisfiability problems for other notions of locality, which will be defined in the following.

We will see in Section 2.3 how we can decide such satisfiability problems, given a decision procedure for a fragment of $\mathcal{T}_0$.

Other Notions of Locality

In previous work [56, 36, 35], some other notions of locality have been introduced, including stable locality, extended locality, Ψ-locality and combinations of these different notions. All of them allow for a larger fragment of first-order logic to be treated with local reasoning, and therefore require a larger set of instances to be generated (which can even be infinite in case of stable locality). In the following, we want to introduce formally the notions of stable locality, Ψ-locality and stable Ψ-locality.

Definition 2.32 ($T_\Sigma(\Theta)$)**.** For a set of ground terms Θ and a set of function symbols Σ, let $T_\Sigma(\Theta)$ be the set of all terms generated from Θ by applying Σ-function symbols (repeatedly) to terms in Θ.

Definition 2.33 ($\mathcal{K}^{[G]}$)**.** Let $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ be a theory extension . For a set G of ground clauses, let

$$\begin{aligned}\mathcal{K}^{[G]} = \{\ C\sigma \ \mid\ & C \in \mathcal{K} \text{ and } \sigma \text{ is such that} \\ & (1.)\ \sigma(x) = t \text{ for some } t \in T_{\Sigma_0}(\mathsf{st}(\mathcal{K} \cup G)), \\ & \qquad \text{if } x \text{ appears in an extension term in } C \\ & (2.)\ \sigma(x) = x, \text{ if } x \text{ does not occur in an extension term } \}.\end{aligned}$$

That is, $\mathcal{K}^{[G]}$ is the set of instances that results from instantiating variables below extension symbols to terms (of the same sort) in $T_{\Sigma_0}(\mathsf{st}(\mathcal{K} \cup G))$, and not instantiating other variables. Note that if all variables in $\mathcal{K}$ appear in extension terms, then $\mathcal{K}^{[G]} = \mathcal{K}^{T_{\Sigma_0}(\mathsf{st}(\mathcal{K} \cup G))}$.

Definition 2.34 (Stably local theory extension [56])**.** A theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is *stably local* if it satisfies condition (SLoc):

(SLoc) For every finite set G of ground clauses,
$\mathcal{T}_0 \cup \mathcal{K} \cup G \models \Box \Leftrightarrow \mathcal{T}_0 \cup \mathcal{K}^{[G]} \cup G \models \Box$.

Note that as soon as Σ_0 contains non-constant function symbols, $\mathcal{K}^{[G]}$ will in general not be finite. Therefore, we will usually restrict the use of stable locality to cases where the base theory has no function symbols that generate terms of an input sort for our extension functions. Even though $T_{\Sigma_0}(\mathsf{st}(\mathcal{K} \cup G))$ can still be infinite in this case, we will only have finitely many terms of an input sort for our extension functions, and therefore $\mathcal{K}^{[G]}$ will be finite.

We can generalize the notions of locality and stable locality by allowing to instantiate the axioms wrt. a set of ground terms that is computed from $\mathsf{st}(\mathcal{K} \cup G)$, instead of instantiating wrt. $\mathsf{st}(\mathcal{K} \cup G)$ itself.

Definition 2.35 (Closure operator)**.** A function Ψ on sets of ground terms is a *closure operator* if (for all sets of ground terms Θ, Θ'):

1. $\Theta \subseteq \Psi(\Theta)$
2. if $\Theta \subseteq \Theta'$, then $\Psi(\Theta) \subseteq \Psi(\Theta')$
3. $\Psi(\Psi(\Theta)) \subseteq \Psi(\Theta)$

Definition 2.36 ($\mathcal{K}^{\Psi}[G]$)**.** Let $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ be a theory extension. For a set G of ground clauses and a closure operator Ψ on sets of $(\Sigma_0 \cup \Sigma_1)$-terms, let

$$\begin{array}{rll} \mathcal{K}^{\Psi}[G] = & \{\ C\sigma & \mid \ C \in \mathcal{K} \text{ and } \sigma \text{ is such that} \\ & & \quad (1.)\ \text{for each extension term } f(t) \text{ in } C, \\ & & \qquad f(t)\sigma \in \Psi(\mathsf{st}(\mathcal{K} \cup G)) \\ & & \quad (2.)\ \sigma(x) = x, \text{ if } x \text{ does not occur in an extension term } \}. \end{array}$$

Definition 2.37 (Ψ-local theory extension [36, 35])**.** A theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is *Ψ-local* (for a given closure operator Ψ) if it satisfies condition (Loc_{Ψ}):

(Loc_{Ψ}) For every finite set G of ground clauses,
$\mathcal{T}_0 \cup \mathcal{K} \cup G \models \Box \Leftrightarrow \mathcal{T}_0 \cup \mathcal{K}^{\Psi}[G] \cup G \models \Box$.

Finally, we can combine stable and Ψ-locality.

Definition 2.38 ($\mathcal{K}^{\Psi[G]}$)**.** Let $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ be a theory extension . For a set G of ground clauses and a closure operator Ψ on sets of $(\Sigma_0 \cup \Sigma_1)$-terms, let

$$\begin{array}{rll} \mathcal{K}^{\Psi[G]} = & \{\ C\sigma & \mid \ C \in \mathcal{K} \text{ and } \sigma \text{ is such that} \\ & & \quad (1.)\ \sigma(x) = t \text{ for some } t \in \Sigma_0(\Psi(\mathsf{st}(\mathcal{K} \cup G))), \\ & & \qquad \text{if } x \text{ appears in an extension term in } C \\ & & \quad (2.)\ \sigma(x) = x, \text{ if } x \text{ does not occur in an extension term } \}. \end{array}$$

Definition 2.39 (Stably Ψ-local theory extension [36, 35])**.** A theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is *stably Ψ-local* (for a given closure operator Ψ) if it satisfies condition (SLoc_{Ψ}):

(SLoc_{Ψ}) For every finite set G of ground clauses,
$\mathcal{T}_0 \cup \mathcal{K} \cup G \models \Box \Leftrightarrow \mathcal{T}_0 \cup \mathcal{K}^{\Psi[G]} \cup G \models \Box$.

2.3 Local Reasoning

In this section, we show how the locality conditions defined before allow for a special approach of reasoning, called *hierarchical reasoning in local theory extensions* [56, 36], or simply *local reasoning* in the following.

Let $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ be a theory extension satisfying one of the locality conditions (Loc), (SLoc), (Loc_{Ψ}) or (SLoc_{Ψ}). Let $\mathcal{K}*[G]$ stand for either $\mathcal{K}[G]$, $\mathcal{K}^{[G]}$, $\mathcal{K}^{\Psi}[G]$ or $\mathcal{K}^{\Psi[G]}$, depending on which notion of locality is satisfied. If, for a given set of ground clauses G, $\mathcal{K}*[G]$ is finite, then the locality conditions give us a possibility to check satisfiability of G modulo the extended theory $\mathcal{T}_0 \cup \mathcal{K}$:

Step 1: Use (stable)(Ψ-)locality.
By the locality condition, $\mathcal{T}_0 \cup \mathcal{K} \cup G \models \Box \ \Leftrightarrow\ \mathcal{T}_0 \cup \mathcal{K}*[G] \cup G \models \Box$.

Step 2: Test satisfiability of $\mathcal{K}*[G] \cup G$.
There are two ways to check whether $\mathcal{T}_0 \cup \mathcal{K}*[G] \cup G \models \Box$:

i) if $\mathcal{K}*[G] \cup G$ contains only ground clauses[6], we can solve the problem with a procedure that solves satisfiability problems for the universal fragment of $\mathcal{T}_0$ with additional free function symbols;

ii) since all terms in $\mathcal{K}*[G]$ that start with Σ_1-symbols are ground (by definition of $\mathcal{K}*[G]$), we can also remove the extension symbols by Ackermann's reduction:

   (a) we replace each extension term $f(t_1, \ldots, t_n)$ in $\mathcal{K}*[G]$ by a fresh constant $c_{f(t_1,\ldots,t_n)}$.

   (b) for all extension functions f, we add the according instances of the congruence axiom: for every pair of terms $f(t_1, \ldots, t_n), f(s_1, \ldots, s_n)$ that have been replaced by $c_{f(t_1,\ldots,t_n)}$ and $c_{f(s_1,\ldots,s_n)}$, respectively, we add a clause $\bigwedge_{i=1}^{n} t_i = s_i \rightarrow c_{f(t_1,\ldots,t_n)} = c_{f(s_1,\ldots,s_n)}$.

   The resulting set of clauses only contains function symbols from Σ_0, its satisfiability can be checked by any satisfiability procedure for $\mathcal{T}_0$.

Regarding automated reasoning, the first approach is somewhat simpler: existing SMT solvers decide the universal fragment of many interesting theories and can usually also handle additional function symbols. However, with the second approach we can also obtain a decision procedure for the universal fragment of $\mathcal{T} \cup \mathcal{K}$ if $\mathcal{K}*[G] \cup G$ does not necessarily consist only of ground clauses, if the $\forall\exists$ fragment of the background theory is decidable (like e.g., $\mathcal{T}_{\mathbb{N}}, \mathcal{T}_{\mathbb{Z}}$ or $\mathcal{T}_{\mathbb{Q}}$).

Definition 2.40 ($\forall\exists$-reducing, universally reducing)**.** A theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is $\forall\exists$-*reducing* if it satisfies (Loc), (SLoc), (Loc_Ψ) or (SLoc_Ψ), and $\mathcal{K}*[G] \cup G$ is finite for every finite set of ground clauses G.

A $\forall\exists$-reducing theory extension is *universally reducing* if every variable in $\mathcal{K}$ has at least one appearance in an extension term.

Theorem 2.41 ([56, 36])**.** *Consider a theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$.*

i) If $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is $\forall\exists$-reducing and the $\forall\exists$ fragment of $\mathcal{T}_0$ is decidable, then the universal fragment of $\mathcal{T}_0 \cup \mathcal{K}$ is decidable.

ii) If $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is universally reducing and the universal fragment of $\mathcal{T}_0$ is decidable, then the universal fragment of $\mathcal{T}_0 \cup \mathcal{K}$ is decidable.

Example 2.42. Suppose $\mathcal{T}_0$ is any theory with a partial ordering (like $\mathcal{T}_{\mathbb{Z}}, \mathcal{T}_{\mathbb{Q}}$ or $\mathcal{T}_{\mathbb{R}}$) and consider its extension with a monotone function f, i.e.

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

We want to check whether the following set of ground clauses G is satisfiable with respect to the extended theory $\mathcal{T}_0 \cup \mathcal{K}$:

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

By locality of the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$, we know that

$$\mathcal{T}_0 \cup \mathcal{K} \cup G \models \square \ \Leftrightarrow\ \mathcal{T}_0 \cup \mathcal{K}[G] \cup G \models \square,$$

[6] A sufficient condition for this is that every variable in $\mathcal{K}$ has at least one appearance below an extension symbol.

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

We have two possibilities to check whether $\mathcal{T}_0 \cup \mathcal{K}[G] \cup G \models \Box$:

i) We give $\mathcal{K}[G] \cup G$ to an SMT solver that decides satisfiability problems in $\mathcal{T}_0$ with uninterpreted functions.

ii) We eliminate function symbol f in $\mathcal{K}[G]$ and G, resulting in

$$G' = \{\ a \leq b,\ \neg(c_1 \leq c_2)\ \},$$

$$\mathcal{K}[G]' = \quad \{\ a \leq a \rightarrow c_1 \leq c_1, \quad a \leq b \rightarrow c_1 \leq c_2 \\ b \leq a \rightarrow c_2 \leq c_1, \quad b \leq b \rightarrow c_2 \leq c_2\ \},$$

and a singleton set of instances of the congruence axiom

$$D = \{\ a = b \rightarrow c_1 = c_2\ \}.$$

Now, $\mathcal{T}_0 \cup \mathcal{K}[G] \cup G \models \Box$ is equivalent to $\mathcal{T}_0 \cup G' \cup \mathcal{K}[G]' \cup D \models \Box$, and the latter can be checked by any decision procedure for $\mathcal{T}_0$ (without additional function symbols).

2.4 Identifying Local Theory Extensions

In order to apply local reasoning, we need to be sure that $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is a (stably, Ψ-) local theory extension. To this end, we can either prove the corresponding locality condition directly, or use a method developed by Sofronie-Stokkermans [56], based on proving embeddability of partial into total models. In this section we will introduce this method, as well as results that allow us to combine disjoint local theory extensions such that the combined extension is again local.

2.4.1 Additional Definitions

We will need the following additional notation:

Definition 2.43 (Partial Π-structure)**.** For a signature $\Pi = (S, \Sigma, \mathsf{Pred})$, a *partial* Π*-structure* is a map M that assigns to every sort $s \in S$ a non-empty set of elements s_M, to every $f \in \Sigma$ with arity $s_1 \times \ldots \times s_n \rightarrow s_{n+1}$ a partial function $f_M : s_{1_M} \times \ldots \times s_{n_M} \rightarrow s_{n+1_M}$, and to every $P \in \mathsf{Pred}$ with arity $s_1 \times \ldots \times s_n$ a set $P_M \subseteq s_{1_M} \times \ldots \times s_{n_M}$.

That is, a partial Π-structure is defined like a Π-structure, except that functions can be partial. The evaluation of a term t with variables from a set V wrt. a variable assignment β in a partial structure M is the same as for total Π-structures, except that the evaluation is undefined for $t = f(t_1, \ldots, t_n)$ if at least one of $\beta(t_i)$ is undefined, or if $(\beta(t_1), \ldots, \beta(t_n))$ is not in the domain of f_M.

Definition 2.44 (Completion)**.** Let $f : O_1 \rightarrow O_2$ be a partial function. A total function $\overline{f} : O_1 \rightarrow O_2$ is called a *completion* of f if $\overline{f}(x) = f(x)$ whenever $f(x)$ is defined.

Definition 2.45 (Weak Σ-homomorphism)**.** Let $\Pi = (S, \Sigma, \mathsf{Pred})$ be a signature, M and N partial Π-structures. For every $s \in S$, let $h_s : s_M \to s_N$ be a total function. Let $U_M = \bigcup_{s \in S} s_M$, $U_N = \bigcup_{s \in S} s_N$ and $h : U_M \to U_N$ the total function with $h(x) = h_s(x)$ whenever $x \in s_M$.

We call $h : U_M \to U_N$ a *weak Σ-homomorphism* if whenever $f_M(a_1, \ldots, a_n)$ is defined, then $f_N(h(a_1), \ldots, h(a_n))$ is also defined, and $h(f_M(a_1, \ldots, a_n)) = f_N(h(a_1), \ldots, h(a_n))$, for every $f \in \Sigma$.

Definition 2.46 (Weak embedding)**.** Let $\Pi = (S, \Sigma, \mathsf{Pred})$ be a signature, and M and N partial Π-structures. Let $U_M = \bigcup_{s \in S} s_M$, $U_N = \bigcup_{s \in S} s_N$.

A weak Σ-homomorphism $i : U_M \to U_N$ is a *weak embedding* of M into N if i is injective and an embedding with respect to Pred, i.e. $(i(a_1), \ldots, i(a_n)) \in P_N$ if and only if $(a_1, \ldots, a_n) \in P_M$, for every $P \in \mathsf{Pred}$. We say that a partial Π-structure M *weakly embeds* into a (total) Π-structure N if there exists a weak embedding of M into N.

Definition 2.47 (Weak satisfaction, weak partial model)**.** We define when a pair (M, β) of a partial Π-structure M and a Π-variable assignment β *weakly satisfies* a Π-literal L, written $(M, \beta) \models_w L$:

- $(M, \beta) \models_w P(t_1, \ldots, t_n)$ iff either $(\beta(t_1), \ldots, \beta(t_n)) \in P_M$ or $\beta(t_i)$ is undefined for some i.
- $(M, \beta) \models_w \neg P(t_1, \ldots, t_n)$ iff either $(\beta(t_1), \ldots, \beta(t_n)) \notin P_M$ or $\beta(t_i)$ is undefined for some i.
- $(M, \beta) \models_w t_1 = t_2$ iff either $\beta(t_1) = \beta(t_2)$ or $\beta(t_i)$ is undefined for some i.
- $(M, \beta) \models_w t_1 \neq t_2$ iff either $\beta(t_1) \neq \beta(t_2)$ or $\beta(t_i)$ is undefined for some i.

That is, for a given pair (M, β), we can have both $(M, \beta) \models_w L$ and $(M, \beta) \models_w \neg L$. Based on weak satisfaction of Π-literals, weak satisfaction of Π-formulas is defined recursively in the same way as standard satisfaction. We write $(M, \beta) \models_w F$ if (M, β) *weakly satisfies* F. If $(M, \beta) \models_w F$ for all Π-variable assignments β, we write $M \models_w F$ and call M a *weak partial Π-model* of F.

Similarly, we can define *Evans satisfaction Evans partial models*, with the difference that we have a special treatment of the equality symbol (which is treated like any other predicate symbol in weak partial models):

Definition 2.48 (Evans Satisfaction, Evans partial model)**.** We define when a pair (M, β) of a partial Π-structure M and a Π-variable assignment β *(Evans) satisfies* a Π-literal L, written $(M, \beta) \models_e L$:

- $(M, \beta) \models_e P(t_1, \ldots, t_n)$ iff either $(\beta(t_1), \ldots, \beta(t_n)) \in P_M$ or $\beta(t_i)$ is undefined for some i.
- $(M, \beta) \models_e \neg P(t_1, \ldots, t_n)$ iff either $(\beta(t_1), \ldots, \beta(t_n)) \notin P_M$ or $\beta(t_i)$ is undefined for some i.
- $(M, \beta) \models_e t_1 = t_2$ iff either $\beta(t_1) = \beta(t_2)$ or both $\beta(t_1)$ and $\beta(t_2)$ are undefined or $\beta(t_1)$ is defined, $t_2 = f(t'_1, \ldots, t'_n)$ and $\beta(t'_i)$ is undefined for some i.
- $(M, \beta) \models_e t_1 \neq t_2$ iff either $\beta(t_1) \neq \beta(t_2)$ or $\beta(t_i)$ is undefined for some i.

Again, Evans satisfaction of Π-formulas is defined recursively in the same way as standard satisfaction. We write $(M, \beta) \models_e F$ if (M, β) *(Evans) satisfies* F. If $(M, \beta) \models_e F$ for all Π-variable assignments β, we write $M \models_e F$ and call M an *Evans partial* Π*-model* of F.

2.4.2 Embeddability Implies Locality

Let $\mathcal{T}_0$ be a theory with signature $\Pi_0 = (S, \Sigma_0, \mathsf{Pred})$ and $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}$ an extended theory with signature $\Pi = (S, \Sigma_0 \cup \Sigma_1, \mathsf{Pred})$. Theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1$ satisfying locality conditions (Loc) or (SLoc) can be recognized by showing that certain partial models of $\mathcal{T}_1$ can be embedded into total models. We consider the following embeddability (and completability) conditions:[7]

($\mathsf{Emb_w}$) Every weak partial Π_1-model M of $\mathcal{T}_1$ where Σ_0-functions are total and Σ_1-functions have a finite domain weakly embeds into a (total) Π_1-model M' of $\mathcal{T}_1$.

($\mathsf{Comp_w}$) Every weak partial Π_1-model M of $\mathcal{T}_1$ where Σ_0-functions are total and Σ_1-functions have a finite domain weakly embeds into a (total) Π_1-model M' of $\mathcal{T}_1$ s.t. $M_{|\Pi_0}$ and $M'_{|\Pi_0}$ are isomorphic.

(Emb) Every Evans partial Π_1-model M of $\mathcal{T}_1$ where Σ_0-functions are total and Σ_1-functions have a finite domain weakly embeds into a (total) Π_1-model M' of $\mathcal{T}_1$.

Definition 2.49 (Σ_1-flat, Σ_1-linear)**.** We say that a formula is Σ_1*-flat* if it does not contain occurrences of constant or function symbols below an extension symbol. A Σ_1-flat formula is Σ_1-linear if all extension terms that contain the same variable are syntactically equal, and no extension term contains two occurrences of the same variable.

The following theorem states that for Σ_1-linear sets of clauses $\mathcal{K}$, locality of a theory extension follows from condition ($\mathsf{Emb_w}$):

Theorem 2.50 ([56, 61])**.** *Let $\mathcal{T}_0$ be a theory with signature $\Pi_0 = (S, \Sigma_0, \mathsf{Pred})$ and $\mathcal{K}$ a set of Π-clauses, with $\Pi = (S, \Sigma_0 \cup \Sigma_1, \mathsf{Pred})$. If all clauses in $\mathcal{K}$ are Σ_1-linear and the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ satisfies ($\mathsf{Emb_w}$) then it satisfies (Loc).*

Similarly, for universal base theories, stable locality of a theory extension follows from condition (Emb):

Theorem 2.51 ([56])**.** *Let $\mathcal{T}_0$ be a universal theory with signature $\Pi_0 = (S, \Sigma_0, \mathsf{Pred})$ and $\mathcal{K}$ a finite set of Π-clauses, with $\Pi = (S, \Sigma_0 \cup \Sigma_1, \mathsf{Pred})$. If the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ satisfies (Emb) then it satisfies (SLoc).*

Note that, as the notion of local theory extensions is a generalization of local theories, these results are a generalization of the embeddability results for local theories by Ganzinger [20].

Theorems 2.50 and 2.51 allow us to identify many examples of local theory extensions and local theories (see Sections 2.5 for existing results and Chapter 3 for new results we establish in this thesis).

[7] Sofronie and Ihlemann [61] call these conditions ($\mathsf{Emb_w^{fd}}$),($\mathsf{Comp_w^{fd}}$) and ($\mathsf{Emb^{fd}}$), respectively. There, the additional superscript fd refers to the finite domain of Σ_1-functions, while the names used here are used for more general notions that allow partial functions with an infinite definition domain. Since in our locality conditions we only consider finite sets of ground clauses, we do not need embeddability conditions in their full generality.

Restriction to Countable Models

In the literature, theory extensions have also been allowed to be collections of structures instead of sets of (first-order) sentences. As we only consider theories as sets of sentences, we can use slightly weaker conditions to prove locality. Define $(\mathsf{Emb}^{\mathsf{c}}_{\mathsf{w}}), (\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$ and $(\mathsf{Emb}^{\mathsf{c}})$ as variants of the notions above that only require countable models to be embeddable into total models.

Then the following is a consequence of Theorems 2.50 and 2.51:

Corollary 2.52. *For theory extensions defined by sets of first-order sentences, locality already follows from* $(\mathsf{Emb}^{\mathsf{c}}_{\mathsf{w}})$ *(together with the other assumptions of Theorem 2.50), and stable locality from* $(\mathsf{Emb}^{\mathsf{c}})$ *(together with the assumptions from Theorem 2.51).*

Proof: To show that embeddability of countable partial models implies locality, we need to show that $\mathcal{T}_0 \cup \mathcal{K} \cup G \models \square \Leftrightarrow \mathcal{T}_0 \cup \mathcal{K} * [G] \cup G \models \square$ holds whenever every (weak or Evans) partial model of $\mathcal{T}_0 \cup \mathcal{K}$ can be embedded into a total model of $\mathcal{T}_0 \cup \mathcal{K}$, according to $(\mathsf{Emb}^{\mathsf{c}}_{\mathsf{w}})$ or $(\mathsf{Emb}^{\mathsf{c}})$, respectively.

If $\mathcal{T}_0 \cup \mathcal{K} * [G] \cup G$ is a set of first-order sentences, by the Downward Löwenheim-Skolem Theorem (see e.g. Barwise [6]), it has a countable model whenever it has any model. Let M be such a countable model. Like in the proof by Ihlemann and Sofronie [61], we can obtain a partial model P from M by restricting the domains of the extension functions. In our case, the resulting partial model will be countable, and therefore embeddability of countable partial models is enough to prove (stable) locality. The rest of the proof works just like the original ones. □

2.4.3 Combination of Local Theory Extensions

Given two local theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1$ and $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_2$ such that the extension symbols of the extensions are disjoint, we can find cases where the combined extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ is also local.

First consider the case where both extensions satisfy condition $(\mathsf{Comp}_{\mathsf{w}})$. It has been shown that then their combination also satisfies condition $(\mathsf{Comp}_{\mathsf{w}})$, and hence also condition (Loc):

Theorem 2.53 ([59])**.** *Let* $\mathcal{T}_0$ *be a theory with signature* $\Pi_0 = (S, \Sigma_0, \mathsf{Pred})$ *and for* $i \in \{1, 2\}$*, let* $\mathcal{K}_i$ *be sets of* Π_i*-clauses, with* $\Pi_i = (S, \Sigma_0 \cup \Sigma_i, \mathsf{Pred})$*, with* $\Sigma_1 \cap \Sigma_2 = \varnothing$ *and .*

If both $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1$ *and* $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_2$ *are theory extensions and satisfy condition* $(\mathsf{Comp}_{\mathsf{w}})$*, then* $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ *is a theory extension and satisfies condition* $(\mathsf{Comp}_{\mathsf{w}})$*. If, additionally, all clauses in both* $\mathcal{K}_i$ *are* Σ_i*-linear, then the extension* $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ *satisfies* (Loc)*.*

Combination results for stably local and Ψ-local extensions are currently under investigation [35].

We will give some examples of combined local theory extensions at the end of Section 2.5.

2.5 Examples of Local Theory Extensions

In this section, we give an (incomplete) overview of theory extensions that have been identified to satisfy locality conditions. In previous papers [56, 57, 58, 59, 61, 36], the following theory extensions have been proved to satisfy the following conditions. Note that all of them except for shallow extensions are universally reducing.

Free functions [56]. The extension of any theory with a set of free function symbols Σ_1 and

$$\mathcal{K} = \varnothing$$

satisfies condition ($\mathsf{Comp_w}$), and by Theorem 2.50 condition (Loc).

Monotonicity [56]. Consider a theory $\mathcal{T}_0$ with sorts A, B, binary predicates $\leq_A$ and $\leq_B$, an extension symbol f of type $A \to B$ and an axiom

$$\mathsf{Mon}(f) = \{\ x \leq_A y \ \to f(x) \leq_B f(y)\ \}.$$

If $(A_M, \leq_{A_M})$ and $(B_M, \leq_{B_M})$ are partially ordered sets for every model M of $\mathcal{T}_0$, then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{Mon}(f)$ satisfies ($\mathsf{Emb_w}$). If in all models $(A_M, \leq_{A_M})$ and $(B_M, \leq_{B_M})$ are (dense) totally-ordered sets, semilattices, distributive lattices, boolean algebras, then the extension satisfies ($\mathsf{Comp_w}$). In both cases, by Theorem 2.50 the extension satisfies condition (Loc).

Generalized monotonicity [61]. Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n$, B and binary predicates $\leq_s$ for every sort s, such that in all models M of $\mathcal{T}_0$, the $(A_{i_M}, \leq_{A_{i_M}})$ are partially ordered sets. Consider an extension symbol f of type $A_1 \times \ldots \times A_n \to B$ and an axiom

$$\mathsf{GMon}(f) = \{\ \textstyle\bigwedge_{i=1}^{n} x_i \sim_i y_i \ \to f(x_1, \ldots, x_n) \leq_B f(y_1, \ldots, y_n)\ \},$$

where $\sim_i \in \{=, \leq_{A_{i_M}}\}$ for all i. The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{GMon}(f)$ satisfies ($\mathsf{Emb_w}$) if in all models M of $\mathcal{T}_0$, $(B_M, \leq_{B_M})$ is a partially ordered set, and it satisfies ($\mathsf{Comp_w}$) if in all models $(B_M, \leq_{B_M})$ is a (dense) totally-ordered set, semilattice, distributive lattice, boolean algebra, $\mathbb{Z}$ or $\mathbb{R}$. In both cases, by Theorem 2.50 the extension satisfies condition (Loc).

Piecewise and blockwise monotonicity [58, 36]. Let $\mathcal{T}_0$ be a theory with sort A, constants $l_1, \ldots, l_m, u_1, \ldots, u_m$ of type A and a binary predicate $\leq$ such that in all models of $\mathcal{T}_0$, $(A_M, \leq)$ is a totally ordered set and $l_{1_M} \leq u_{1_M} < l_{2_M} \leq u_{2_M} < \ldots < l_{m_M} \leq u_{m_M}$. Consider an extension symbol f of type $A \to A$ and a set of axioms

$$\mathsf{PMon}(f) = \left\{ \begin{array}{l} l_1 \leq x \leq y \leq u_1 \ \to f(x) \leq f(y) \\ \cdots \\ l_m \leq x \leq y \leq u_m \ \to f(x) \leq f(y) \end{array} \right\}.$$

The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{PMon}(f)$ satisfies ($\mathsf{Comp_w}$).

For the same base theory, the extension with a symbol f of type $A \to A$ and axioms

$$\mathsf{BMon}(f) = \left\{ \begin{array}{l} l_1 \leq x \leq u_1 < l_2 \leq y \leq u_2 \ \to f(x) \leq f(y) \\ \cdots \\ l_{m-1} \leq x \leq u_{m-1} < l_m \leq y \leq u_m \ \to f(x) \leq f(y) \end{array} \right\}$$

also satisfies ($\mathsf{Comp_w}$).

In both cases, by Theorem 2.50 the extension satisfies condition (Loc). Similar conditions can be defined for n-ary functions over possibly many-sorted theories.

Boundedness [57, 61]. Let $\mathcal{T}_0$ be a base theory with sorts $A_1, \ldots, A_n, B$ and a binary predicate $\leq$ on B such that $\mathcal{T}_0 \models x \leq x$. Consider an extension symbol f of type $A_1 \times \ldots \times A_n \to B$ and an axiom

$$\mathsf{Bound}^t(f) = \{\ f(\overline{x}) \leq t[\overline{x}]\ \},$$

where $t[\overline{x}]$ is a term in the base theory. The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{Bound}^t(f)$ satisfies ($\mathsf{Comp_w}$), and by Theorem 2.50 condition (Loc).

We can also consider an extension with

$$\mathsf{Bound}^{s,t}(f) = \{\ s[\overline{x}] \leq f(\overline{x}) \leq t[\overline{x}]\ \},$$

where $s[\overline{x}]$ is another term in the base theory. If in all models M of $\mathcal{T}_0$, $\leq_M$ is both reflexive and transitive and $M \models s[\overline{x}] \leq t[\overline{x}]$, then this extension also satisfies ($\mathsf{Comp_w}$).

Guarded boundedness [61]. The boundedness condition above can also be guarded by additional constraints. Consider

$$\mathsf{GBound}^{s,t}_{\phi}(f) = \{\ \phi(\overline{x}) \ \to\ s[\overline{x}] \leq f(\overline{x}) \leq t[\overline{x}]\ \},$$

where $\phi[\overline{x}]$ is a conjunction of literals in the base theory with variables among $\overline{x}$, and everything else is as above. The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{GBound}^{s,t}_{\phi}(f)$ satisfies ($\mathsf{Comp_w}$), and by Theorem 2.50 condition (Loc).

Monotonicity and boundedness [57]. Like in the case of generalized monotonicity, consider a base theory $\mathcal{T}_0$ with sorts $A_1, \ldots, A_n$, B and binary predicates $\leq_s$ for every sort s, such that in all models M of $\mathcal{T}_0$, the $(A_{i_M}, \leq_{A_{i_M}})$ are partially ordered sets. Consider an extension symbol f of type $A_1 \times \ldots \times A_n \to B$ that satisfies not only $\mathsf{GMon}(f)$, but is in addition bounded by a term $t[\overline{x}]$ in the base theory with the same monotonicity as f. That is, $\mathcal{T}_0$ is extended with

$$\mathcal{K} = \{\ \mathsf{GMon}(f),\ \mathsf{Bound}^t(f)\ \}$$

as defined above, and we additionally require that $\mathcal{T}_0 \models \mathsf{GMon}(t)$ with the same $\sim_i$ as $\mathsf{GMon}(f)$ for all i.

The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{GMon}(f)$ satisfies ($\mathsf{Emb_w}$) if in all models M of $\mathcal{T}_0$, $(B_M, \leq_{B_M})$ is a partially ordered set, and it satisfies ($\mathsf{Comp_w}$) if in all models $(B_M, \leq_{B_M})$ is a (dense) totally-ordered set, semilattice, distributive lattice, boolean algebra, $\mathbb{Z}$ or $\mathbb{R}$. In both cases, by Theorem 2.50 the extension satisfies condition (Loc).

Lipschitz functions [56]. For the base theory $\mathcal{T}_{\mathbb{R}}$, consider an extension symbol f of type $\mathbb{R} \to \mathbb{R}$ satisfying the Lipschitz condition for some λ at a given point x_0, i.e.

$$\mathsf{Lip}^{\lambda}_{x_0}(f) = \{\ |f(x) - f(x_0)| \leq \lambda \cdot |x - x_0|\ \}.$$

The extension $\mathcal{T}_{\mathbb{R}} \subseteq \mathcal{T}_{\mathbb{R}} \cup \mathsf{Lip}^{\lambda}_{x_0}(f)$ satisfies ($\mathsf{Comp_w}$), and by Theorem 2.50 condition (Loc).

Injectivity [36]. Assume our base theory $\mathcal{T}_0$ has sorts A and B and is such that in all models M of $\mathcal{T}_0$, $|A_M| \leq |B_M|$. Consider an extension symbol f of type $A \to B$ and a clause

$$\mathsf{Inj}(f) = \{\ x \neq y \ \to f(x) \neq f(y)\ \}.$$

The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{Inj}(f)$ satisfies ($\mathsf{Comp_w}$), and by Theorem 2.50 condition (Loc).

Selector functions [56]. Let $\mathcal{T}_0$ be a base theory with a constructor c of type $A_1 \times \ldots \times A_n \to B$. Consider extension symbols s_i of types $B \to A_i$, for $1 \leq i \leq n$, and a set of clauses

$$\mathsf{Sel}(c) = \left\{ \begin{array}{l} x = c(x_1, \ldots, x_n) \ \to \ c(s_1(x), \ldots, s_n(x)) = x, \\ s_1(c(x_1, \ldots, x_n)) = x_1, \\ \ldots \\ s_n(c(x_1, \ldots, x_n)) = x_n \end{array} \right\}.$$

If in every model M of $\mathcal{T}_0$, c_M is injective, then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{Sel}(c)$ satisfies ($\mathsf{Comp_w}$), and by Theorem 2.50 condition (Loc) . Otherwise, the extension satisfies (Emb), and by Theorem 2.51 condition (SLoc).

Shallow extensions [56, 24, 25]. We call a clause *shallow* if extension symbols only occur in positive literals and only at the root of terms. If $\mathcal{K}$ is a set of shallow clauses, then for any base theory $\mathcal{T}_0$, the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ satisfies condition (Emb), and by Theorem 2.51 condition (SLoc).

Array properties [36, 35]. The *array property fragment* [9] is a fragment of the theory of arrays with $\mathcal{T}_\mathbb{Z}$ as index theory and a parametric element theory $\mathcal{T}_E$. Consider the disjoint combination $\mathcal{T}_0 = \mathcal{T}_\mathbb{Z} \cup \mathcal{T}_E$, and its extension with functions $\mathsf{read}, \mathsf{write}$ and clauses[8]:

$$\mathsf{read}(\mathsf{write}(a, i, e), i) = e,$$
$$j \neq i \ \to \mathsf{read}(\mathsf{write}(a, i, e), j) = \mathsf{read}(a, j).$$

The array property fragment is defined as follows:

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

The decision procedure proposed originally decides satisfiability of formulas in negation normal form in the array property fragment in the following steps [9].

i) Replace all existentially quantified array variables with Skolem constants; replace all terms of the form $\mathsf{read}(a, i)$ with $a(i)$; eliminate all terms of the form $\mathsf{write}(a, i, e)$ by replacing the formula $\phi(\mathsf{write}(a, i, e))$ with the conjunction of the formula $\phi(b)$ (obtained by introducing a fresh array

[8]The considerations below are for arrays of dimension 1, the general case is similar.

name b for $\mathsf{write}(a, i, e)$) with $(b(i) = e) \wedge \forall j(j \leq i - 1 \vee i + 1 \leq j \rightarrow b(j) = a(j))$.[9]

ii) Existentially quantified index variables are replaced with Skolem constants.

iii) Universal quantification over index variables is replaced by conjunction of suitably chosen instances of the variables.

Let F be a formula in the array property fragment. Assume that we apply transformation steps i) and ii). In the resulting formula, let Σ_1 be the set of array function symbols, let R be the set of ground index terms that appear below Σ_1-symbols, and let B be the set of ground index terms that appear in index guards. Furthermore, separate the resulting formula into a set of non-ground clauses $\mathcal{K}$ and a set of ground clauses G, and let $I = R \cup B$.

For $\Psi(T) = T \cup \{ f(i_1, \ldots, i_n) \mid f \in \Sigma_1, i_1, \ldots, i_n \in I \}$, we have

$$\mathcal{T}_0 \cup \mathcal{K} \cup G \models \Box \Leftrightarrow \mathcal{T}_0 \cup \mathcal{K}^{\Psi}[G] \cup G \models \Box.$$

That is, any $\mathcal{K}$ resulting from this transformation satisfies condition (Loc_Ψ) for Ψ chosen as above.

Local pointer data structures [36, 35]. McPeak and Necula investigated reasoning in pointer data structures [46]. The given signature has sorts p (pointer) and s (scalar). Sets of pointer and scalar fields are modeled by sets Σ_p and Σ_s of function symbols of type $\mathsf{p} \rightarrow \mathsf{p}$ and $\mathsf{p} \rightarrow \mathsf{s}$, respectively. The sort p contains a constant null. There are no predicates of sort p (besides equality); predicates of sort s can have any arity. Then, *local equality axioms* are of the form

$$\mathcal{E} \vee \mathcal{C},$$

where $\mathcal{E}$ is a disjunction of positive pointer equalities, $\mathcal{C}$ contains arbitrary scalar constraints, and it is assumed that for all terms $f_1(f_2(\ldots f_n(p)))$ occurring in an axiom, $\mathcal{E}$ also contains the disjunction $p = \mathsf{null} \vee f_n(p) = \mathsf{null} \vee \cdots \vee f_2(\ldots f_n(p)) = \mathsf{null}$, in order to exclude null pointer errors. Quantification is only allowed over variables of sort p.

Let $\Psi(T) = T \cup \{ f(t) \mid t \in \mathsf{st}(\mathcal{K}) \cup T, f \in \Sigma_s \}$. Consider a base theory $\mathcal{T}_0 = \mathcal{T}_\mathsf{p} \cup \mathcal{T}_\mathsf{s}$, where $\mathcal{T}_\mathsf{p}$ is the empty theory with single sort p and $\Sigma_\mathsf{p} = \{\mathsf{null}\}$, and $\mathcal{T}_\mathsf{s}$ is an arbitrary scalar theory.

Then, any extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$, where $\mathcal{K}$ is a set of local equality axioms, satisfies condition (SLoc_Ψ).

Note that all quantified variables in $\mathcal{K}$ are of sort p, and in our base signature Σ_0 we do not have function symbols that generate new terms of sort p from $\Psi(\mathsf{st}(\mathcal{K} \cup G))$. Thus, $\mathcal{K}^{\Psi[G]}$ is finite, and even polynomial in the size of $\mathsf{st}(\mathcal{K} \cup G)$, for every $\mathcal{K}$ and G.

Combinations of extensions. By Theorem 2.53, we can combine any number of signature-disjoint extensions that satisfy (Comp_w). E.g., the base theory $\mathcal{T}_\mathbb{R}$ can be extended simultaneously by two monotone functions f and g, and an injective function h. The extension $\mathcal{T}_\mathbb{R} \subseteq \mathcal{T}_\mathbb{R} \cup \mathsf{Mon}(f) \cup \mathsf{Mon}(g) \cup \mathsf{Inj}(h)$ is local.

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

2.6 Chains of extensions

Locality of a theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ allows us to solve the satisfiability problem for arbitrary set of ground clauses G modulo the extended theory $\mathcal{T}_0 \cup \mathcal{K}$. In Section 2.5, we have seen several examples of local theory extensions. But even if we have a set of clauses $\mathcal{K}$ and a theory $\mathcal{T}_0$ such that $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is not (or not known to be) a local theory extension, we may still be able to treat it within the local reasoning framework:

Assume that we can split $\mathcal{K}$ into two disjoint sets $\mathcal{K}_1$ and $\mathcal{K}_2$ such that both $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1$ and $\mathcal{T}_0 \cup \mathcal{K}_1 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ are local extensions. This means we can extend the base theory $\mathcal{T}_0$ to $\mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ in two steps, and use the local reasoning procedure repeatedly to reduce the problem of checking satisfiability of a set of ground clauses in $\mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$ to a satisfiability problem in the base theory $\mathcal{T}_0$. This approach can be generalized to an arbitrary number of steps, and also works for Ψ-local extensions. We call such a repeated extension a *chain of extensions*.

2.6.1 Local Reasoning in Chains of Extensions

Consider a base theory $\mathcal{T}_0$ and clause sets $\mathcal{K}_1, \ldots, \mathcal{K}_m$. For $0 \leq j \leq m-1$, 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 universally reducing theory extension for $1 \leq j \leq m-1$, and $\forall\exists$-reducing for $j = 0$. Then, for every extension, we can use local reasoning to reduce the problem of checking satisfiability of a given set of ground clauses G from theory $\mathcal{T}_j$ to theory $\mathcal{T}_{j-1}$:

$$\mathcal{T}_j \cup G \models \Box \Leftrightarrow \mathcal{T}_{j-1} \cup \mathcal{K}_j^{(\Psi)}[G] \cup G \models \Box.$$

If we define the sets G_j recursively by

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

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

Definition 2.54 ($\forall\exists$-reducing chain, universally reducing chain)**.** A chain of theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \ldots \subseteq \mathcal{T}_m$ is $\forall\exists$-*reducing* if the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ is $\forall\exists$-reducing and all extensions $\mathcal{T}_j \subseteq \mathcal{T}_{j+1}$, $1 \leq j \leq m-1$, are universally reducing.

A $\forall\exists$-*reducing* chain of extensions is *universally reducing* if $\mathcal{T}_0 \subseteq \mathcal{T}_1$ is also universally reducing.

The following is a consequence of the considerations above and Theorem 2.41:

Theorem 2.55. *Let $\mathcal{T}_0$ be a theory, $\mathcal{K}_1, \ldots, \mathcal{K}_m$ sets of clauses and let $\mathcal{T}_{j+1} = \mathcal{T}_j \cup \mathcal{K}_{j+1}$ for $0 \leq j \leq m-1$. Consider the chain of theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \ldots \subseteq \mathcal{T}_m$.*

i) If $\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \ldots \subseteq \mathcal{T}_m$ is $\forall\exists$-reducing and the $\forall\exists$-fragment of $\mathcal{T}_0$ is decidable, then the universal fragment of $\mathcal{T}_m = \mathcal{T}_0 \cup \mathcal{K}_1 \cup \ldots \cup \mathcal{K}_m$ is decidable.

ii) If $\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \ldots \subseteq \mathcal{T}_m$ is universally reducing and the universal fragment of $\mathcal{T}_0$ is decidable, then the universal fragment of $\mathcal{T}_m = \mathcal{T}_0 \cup \mathcal{K}_1 \cup \ldots \cup \mathcal{K}_m$ is decidable.

Example 2.56. Consider a base theory $\mathcal{T}_0$ with a partial ordering $\leq$. Suppose we want to define a monotone function f with

$$\mathcal{K}_1 = \{\ x \leq y \ \rightarrow\ f(x) \leq f(y)\ \},$$

and a function g that is bounded by f:

$$\mathcal{K}_2 = \{\ g(x) \leq f(x)\ \}.$$

We want to decide the satisfiability of sets of ground clauses G in the extended theory $\mathcal{T}_2 = \mathcal{T}_0 \cup \mathcal{K}_1 \cup \mathcal{K}_2$. We cannot use the approach from Section 2.3 directly, since we do not know whether the extension $\mathcal{T}_0 \subseteq \mathcal{T}_2$ is local.

However, if we let $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}_1$, then we do know that both $\mathcal{T}_0 \subseteq \mathcal{T}_1$ and $\mathcal{T}_1 \subseteq \mathcal{T}_2$ are local extensions (see the examples in Section 2.5). Therefore, we can use local reasoning in chains of extensions to reduce a satisfiability problem in the theory $\mathcal{T}_2$ in two steps to a satisfiability problem in $\mathcal{T}_0$.

Suppose we want to find out whether

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

is satisfiable in $\mathcal{T}_2$. By the approach from Section 2.6.1, we have

$$\begin{array}{ll} & \mathcal{T}_2 \cup G \models \square \\ \Leftrightarrow & \mathcal{T}_1 \cup \underbrace{\mathcal{K}_2[G] \cup G}_{G_1} \models \square \\ \Leftrightarrow & \mathcal{T}_0 \cup \mathcal{K}_1[G_1] \cup G_1 \models \square, \end{array}$$

where

$$\mathcal{K}_2[G] = \{\ g(a) \leq f(a)\ \}$$

and

$$\mathcal{K}_1[G_1] = \left\{ \begin{array}{ll} a \leq a \ \rightarrow\ f(a) \leq f(a), & a \leq b \ \rightarrow\ f(a) \leq f(b), \\ b \leq a \ \rightarrow\ f(b) \leq f(a), & b \leq b \ \rightarrow\ f(b) \leq f(b) \end{array} \right\}.$$

In $\mathcal{T}_0$, G is unsatisfiable together with $g(a) \leq f(a)$ and $a \leq b \ \rightarrow\ f(a) \leq f(b)$, where f and g are considered as free function symbols.

Note that this also proves that for $\mathcal{K} = \mathcal{K}_1 \cup \mathcal{K}_2$, the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is not local: for the given G we have

$$\mathcal{K}[G] = \{\ g(a) \leq f(a),\ b \leq b \ \rightarrow\ f(b) \leq f(b)\ \},$$

and $\mathcal{K}[G] \cup G$ is satisfiable in $\mathcal{T}_0$, even though we have just shown that $\mathcal{K} \cup G$ is not.

Chains of Extensions in Verification

Chains of extensions are especially useful in the context of verification. We give a brief overview of the method here and refer to Chapter 5 for details.

The approach described above can be used for invariant checking and bounded model checking of parameterized systems. To this end, the initial condition and the invariant of a system are expressed as sets of clauses (Init) and (Inv), such that $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Init})$ and $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Inv})$ are local extensions for a suitable base theory $\mathcal{T}_0$. Furthermore, the transition relation of the system is modeled by a set of clauses (Update), where $\mathcal{T}_0 \cup (\mathsf{Inv}) \subseteq \mathcal{T}_0 \cup (\mathsf{Inv}) \cup (\mathsf{Update})$ is another local extension. We have seen several examples of extensions that can be used as initial conditions and invariants in Section 2.5. We will present new results and a class of extensions specifically suited to model transition relations in Chapter 3.

To check whether (Inv) is a safety invariant for a given safety property (Safe), we need to prove

i) $\mathcal{T}_0 \cup (\mathsf{Inv}) \models (\mathsf{Safe})$,

ii) $\mathcal{T}_0 \cup (\mathsf{Init}) \models (\mathsf{Inv})$, and

iii) $\mathcal{T}_0 \cup (\mathsf{Inv}) \cup (\mathsf{Update}) \cup \neg(\mathsf{Inv}') \models \square$,

where (Inv') is a variant of (Inv) with renamed function and constant symbols (such that they refer to values of system variables after the update). We will show in Chapter 5 that under suitable assumptions this can be done with the method from above.

If furthermore suitably renamed variants of (Update) can be repeatedly added, preserving a locality condition in each step, we can check whether unsafe states can be reached in n update steps of the system by checking

$$\mathcal{T}_0 \cup (\mathsf{Init}_0) \cup (\mathsf{Update}_i)_{1 \leq i \leq n} \cup \neg(\mathsf{Safe}_n) \models \square,$$

where formulas with subscript contain renamed function and constant symbols which refer to their value after n transition steps of the system.

This can be done even if we cannot find an invariant that implies the safety condition, as long as the safety condition is universal, i.e. its negation is a set of ground clauses.

Chapter 3

New Locality Results

In this section, we introduce some theory extensions that have before not been known to be local, and prove their locality. Like most of the examples in Section 2.5, all of the theory extensions we introduce are universally reducing. We will give example applications of some of these new extensions in Chapter 5.

3.1 Enrichment with Ground Clauses

First of all, we want to show that any local extension can be enriched with an arbitrary set of ground clauses in the extended signature without loss of locality.

Theorem 3.1. *Let $\mathcal{T}_0$ be a base theory, $\mathcal{K}$ a set of clauses such that $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is a local theory extension. Then, for any set of ground clauses H in the signature of $\mathcal{T}_0 \cup \mathcal{K}$, the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K} \cup H$ is local.*

Proof: Assume that $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is a local extension, and both H and G are arbitrary sets of ground clauses in the signature of $\mathcal{T}_0 \cup \mathcal{K}$. We want to show

$$\mathcal{T}_0 \cup \mathcal{K} \cup H \cup G \models \square \;\Leftrightarrow\; \mathcal{T}_0 \cup (\mathcal{K} \cup H)[G] \cup G \models \square.$$

By locality of $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$, we have

$$\mathcal{T}_0 \cup \mathcal{K} \cup H \cup G \models \square \;\Leftrightarrow\; \mathcal{T}_0 \cup \mathcal{K}[H \cup G] \cup H \cup G \models \square.$$

By definition of $\mathcal{K}[G']$ for sets of ground clauses G', we know that $\mathcal{K}[H \cup G] \cup H = (\mathcal{K} \cup H)[G]$ (since in the latter, clauses in H are not instantiated and in both cases all ground terms from $\mathcal{K}, H$ and G are considered). Therefore,

$$\mathcal{T}_0 \cup \mathcal{K}[H \cup G] \cup H \cup G \models \square \;\Leftrightarrow\; (\mathcal{T}_0 \cup \mathcal{K} \cup H)[G] \cup G \models \square.$$

□

3.2 Strictly Monotone Functions

Locality of extensions with unary, non-strictly monotone functions has already been shown by Sofronie-Stokkermans [56], and this result was later extended to strictly monotone functions from $\mathbb{Z}$ to $\mathbb{R}$ [40] and to monotone functions with

arbitrary arity (and different monotonicity in different arguments) [61]. The strictly monotone version has been used to model safety properties of parameterized systems (see Sections 5.4 and 5.5 for example applications). Here, we present a generalization of the locality result for strictly monotone functions. We first consider the case of domains with a total ordering, then domains with a partial ordering.

3.2.1 Preliminaries

Definition 3.2 (Strict order)**.** For any (partial) order $\leq$, there is an associated *strict (partial) order* $<$, defined by

$$x < y \Leftrightarrow x \leq y \ \wedge \ x \neq y.$$

Whenever we use two such orders $\leq$ and $<$ (possibly with subscript), we assume that $\leq$ is a (partial) order and $<$ is the associated strict (partial) order.

In the following, we will consider extension functions f of type $A \to B$, where $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$ is a product of sorts A_i, and B is a sort of the base theory $\mathcal{T}_0$. For simplicity of presentation, we represent f as a unary function, although its type can also be written as $A_1^{k_1} \times \ldots \times A_n^{k_n} \to B$. To describe monotonicity of such n-ary function symbols, we will also need the following definition.

Definition 3.3 (Lexicographic order)**.** Let $O_1, \ldots, O_n$ be sets with orders $<_i$. A *lexicographic order* $<$ on $O_1 \times \ldots \times O_n$ is defined by

$$\begin{array}{lll} (a_1, \ldots, a_n) < (b_1, \ldots, b_n) \Leftrightarrow & & a_1 <_1 b_1 \\ & \vee & (a_1 = b_1 \ \wedge \ a_2 <_2 b_2) \\ & \vee & \ldots \\ & \vee & (\bigwedge_{i=1}^{n-1} a_i = b_i \ \wedge \ a_n <_n b_n), \end{array}$$

We say that $<$ is based on the $<_i$, $1 \leq i \leq n$.

It is easy to see that a lexicographic order is total if it is based on total orders $<_i$, and it is strict (i.e., irreflexive) if all $<_i$ are strict.

Definition 3.4 (Monotone functions)**.** For sets O_1 and O_2 with given (partial) orders $\leq_1$ and $\leq_2$, a *monotone (partial) function* is a function $f : O_1 \to O_2$ that satisfies

$$x \leq_1 y \ \to \ f(x) \leq_2 f(y),$$

whenever $f(x)$ and $f(y)$ are defined. It is a *strictly monotone (partial) function* if it also satisfies

$$x <_1 y \ \to \ f(x) <_2 f(y)$$

whenever $f(x)$ and $f(y)$ are defined.

Definition 3.5 (Open Set)**.** A set O is *open* wrt. an irreflexive relation $<$ if for every $a \in O$ there are $a_1, a_2 \in O$ with $a_1 < a$ and $a < a_2$.

Definition 3.6 (Dense Set)**.** A set O is *dense* wrt. an irreflexive relation $<$ if for all $a_1, a_2 \in O$ with $a_1 < a_2$ there is an $a \in O$ with $a_1 < a < a_2$.

Definition 3.7 ((Open) Interval)**.** Let O be a set and $<$ a binary relation on O. An *interval* of O (wrt. $<$) is a subset of O defined by either $\{\ c \in O \mid c < a\ \}$, $\{\ c \in O \mid a < c\ \}$, or $\{\ c \in O \mid a < c < b\ \}$, for some $a, b \in O$. It is an *open* interval if $<$ is irreflexive. In this case, intervals of the last kind are denoted by (a, b).

Note that if O is dense wrt. $<$, then (a, b) is an open set for any $a, b \in O$. Similarly, $\{\ c \in O \mid c < a\ \}$ and $\{\ c \in O \mid a < c\ \}$ are open sets for any $a \in O$ if O is dense and open wrt. $<$.

3.2.2 Totally Ordered Domains

For the proof of locality, we will need the following lemma:

Lemma 3.8. *Let O_1 be a countable set with strict total order $<_1$ and O_2 a set with strict total order $<_2$ such that O_2 is dense and open wrt. $<_2$. Then there is a strictly monotone function $f : O_1 \to O_2$.*

Proof: Since O_1 is countable, there is a bijective function $i : \mathbb{N} \to O_1$. We show how to inductively define $f(i(n))$ for all $n \in \mathbb{N}$:

Case $n = 1$: Assume f is nowhere defined. Then choose an arbitrary $b \in O_2$ and let $f(i(1)) := b$. Clearly, the function defined thus far is strictly monotone (on its domain).

Case $n \to n+1$: Assume $f : \{\ i(1), \ldots, i(n)\ \} \to O_2$ is strictly monotone. Let $a = i(n+1)$, $D^- = \{\ i(j) \mid 1 \leq j \leq n, i(j) <_1 a\ \}$ and $D^+ = \{\ i(j) \mid 1 \leq j \leq n, a <_1 i(j)\ \}$. Let $l = max(D^-)$ and $r = min(D^+)$, or let them be undefined if the respective sets are empty. Note that only one of them can be undefined. Then, define

$$f(i(n+1)) := \begin{cases} \text{arbitrary } b \in O_2 \text{ with } b <_2 f(r), & \text{if } l \text{ is undefined} \\ \text{arbitrary } b \in O_2 \text{ with } b >_2 f(l), & \text{if } r \text{ is undefined} \\ \text{arbitrary } b \in (f(l), f(r)), & \text{otherwise} \end{cases}$$

Such a b can always be found, as we assumed that O_2 is dense and open wrt. $<_2$. Based on the assumption that $f : \{\ i(1), \ldots, i(n)\ \} \to O_2$ is strictly monotone, it is easy to see that $f : \{\ i(1), \ldots, i(n), i(n+1)\ \} \to O_2$ is also strictly monotone. □

Theorem 3.9. *Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n, B$ and a binary predicate $<_B$ of type $B \times B$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$, and $<_A$ a binary predicate*[1] *of type $A \times A$, such that for every model M of $\mathcal{T}_0$, $<_{A_M}$ and $<_{B_M}$ are strict total orders, and B_M is dense and open wrt. $<_{B_M}$. Consider an extension symbol f of type $A \to B$ satisfying*

$$\mathsf{SMon}(f) = \{\ x <_A y \to f(x) <_B f(y)\ \}.$$

The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{SMon}(f)$ satisfies ($\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}}$), and thus it is local .

Proof: Let M be a countable weak partial model of $\mathcal{T}_0 \cup \mathsf{SMon}(f)$, where $f_M(x)$ is defined for each $x \in D = \{\ a_1, \ldots, a_n\ \} \subseteq A_M$, and undefined otherwise, and

[1] In fact, $<_A$ need not be a predicate in the signature, but can also be defined as a formula over this signature, like the lexicographic order.

all other functions are total. Assume wlog. that $a_i <_{A_M} a_j$ if $i < j$. Then we can partition A_M into D and sets $A_0, \ldots, A_n$ such that

$$\begin{array}{lll} A_0 & = \{\ a \in A_M \mid a <_{A_M} a_1\ \}, & \\ A_i & = (a_i, a_{i+1}), & \text{for } 1 \leq i \leq n-1, \text{ and} \\ A_n & = \{\ a \in A_M \mid a >_{A_M} a_n\ \}. & \end{array}$$

Since M is a partial model, we have $f_M(a_i) <_{B_M} f_M(a_j)$ if $a_i <_{A_M} a_j$ and $a_i, a_j \in D$. Thus, we can partition B_M into $F = \{\ f_M(a_1), \ldots, f_M(a_n)\ \}$ and intervals $B_0, \ldots, B_n$ such that

$$\begin{array}{lll} B_0 & = \{\ b \in B_M \mid b <_{B_M} f(a_1)\ \}, & \\ B_i & = (f(a_i), f(a_{i+1})), & \text{for } 1 \leq i \leq n-1, \text{ and} \\ B_n & = \{\ b \in B_M \mid b >_{B_M} f(a_n)\ \}. & \end{array}$$

Since B_M is dense and open wrt. $<_{B_M}$, all B_i are non-empty open sets. By Lemma 3.8, there is a strictly monotone function $f_i : A_i \to B_i$ for every pair of sets (A_i, B_i). Now, define

$$\overline{f}(x) = \begin{cases} f_M(x) & \text{if } x \in D \\ f_i(x) & \text{if } x \in A_i \end{cases}$$

$\overline{f}$ is a completion of f_M. It satisfies $\mathsf{SMon}(\overline{f})$ because every f_i is strictly monotone, and for every a_i we have $f_{i-1}(x) <_{B_M} f_M(a_i)$ for all $x \in A_{i-1}$, as well as $f_M(a_i) < f_i(x)$ for all $x \in A_i$.

Thus, define the structure N to be the same as M, except that $f_N(x) = \overline{f}(x)$ for all x. N is a total structure and a model of $\mathcal{T}_0 \cup \mathsf{SMon}(f)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$. □

Example 3.10. By Theorem 3.9, extensions of the following base theories (with the usual orderings[2]) with $\mathsf{SMon}(f)$ are local:

- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Z} \to \mathbb{Q}$,
- $\mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Q} \to \mathbb{Q}$,
- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Z} \to \mathbb{R}$,
- $\mathcal{T}_{\mathbb{Q}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Q} \to \mathbb{R}$,
- $\mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{R} \to \mathbb{R}$.

Note that because first-order logic cannot distinguish between countable and uncountable models, we can even extend the combination $\mathcal{T}_{\mathbb{Q}} \cup \mathcal{T}_{\mathbb{R}}$ of linear rational arithmetic and real closed fields with $\mathsf{SMon}(f)$ for an extension symbol f of type $\mathbb{R} \to \mathbb{Q}$, and the resulting theory extension is not inconsistent.

Example 3.11. We can also consider extension symbols f of type $A_1 \times \ldots \times A_n \to B$, with relations $<_i$ of types $A_i \times A_i$ and $<_B$ of type $B \times B$ that are strict total orders in any model of the base theory. If we use the lexicographic order based on the $<_i$ on the domain of f (and the usual order on the codomain), extensions of the following base theories with $\mathsf{SMon}(f)$ are local (again, by Theorem 3.9):

[2]Actually, we can choose one of the usual orderings for every theory, i.e. either $<$ or $>$.

- $\mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Q}^n \to \mathbb{Q}$,
- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Z}^m \times \mathbb{Q}^n \to \mathbb{Q}$,
- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{Q}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Z}^l \times \mathbb{Q}^m \times \mathbb{R}^n \to \mathbb{R}$.

However, especially when considering n-ary functions, we would like to consider also partial orderings, e.g. the ordering $<$ defined by

$$(x_1, y_1) < (x_2, y_2) \Leftrightarrow x_1 <_1 x_2 \ \wedge \ y_1 >_2 y_2,$$

where the $<_i$ are strict total orders. Locality of extensions with $\mathsf{SMon}(f)$ for domains with partial orderings will be considered in the next section.

3.2.3 Partially Ordered Domains

Theorem 3.12. *Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n, B$ and a binary predicate $<_B$ of type $B \times B$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$, and $<_A$ a binary predicate[3] of type $A \times A$, such that for every model M of $\mathcal{T}_0$, $<_{A_M}$ is a strict partial order on A_M and $<_{B_M}$ is a strict total order on B_M, such that B_M is dense and open wrt. $<_{B_M}$. Consider an extension symbol f of type $A \to B$ satisfying*

$$\mathsf{SMon}(f) = \{ \ x <_A y \ \to \ f(x) <_B f(y) \ \}.$$

The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{SMon}(f)$ satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$, and thus it is local.

Proof: By Szpilrajn's Theorem [64], we can find a strict total order $<$ on A_M such that $a <_A b \ \to \ a < b$ for all $a, b \in A_M$. Thus, every function f that is monotone wrt. $<$ is also monotone wrt. $<_{A_M}$.

That is, we can use the same completion as in the proof of Theorem 3.9, based on the total ordering $<$, and the resulting function $\overline{f}$ will satisfy $\mathsf{SMon}(\overline{f})$ wrt. $<_A$. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$ and is therefore local. □

Example 3.13. Consider a theory with sorts $A_1, \ldots, A_n$ and binary predicates $<_i$ of types $A_i \times A_i$ such that every $<_i$ is a strict partial order on A_i in any model of the theory.

Define the partial order $<$ by

$$(a_1, \ldots, a_n) < (b_1, \ldots, b_n) \Leftrightarrow \textstyle\bigwedge_{i=1}^{n} a_i <_i b_i.$$

By Theorem 3.12, extensions of the following base theories with $\mathsf{SMon}(f)$ based on such a partial ordering $<$ on the domain of f are local:

- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Z}^m \times \mathbb{Q}^n \to \mathbb{Q}$,
- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{Q}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Z}^m \times \mathbb{Q}^n \times \mathbb{R}^k \to \mathbb{R}$.

[3] Again, $<_A$ can also be defined as a formula over the given signature.

3.2.4 Bounded Strict Monotonicity

In our applications, we noticed that it is sometimes useful to have locality only on a bounded interval of the given domain. The following is an easy consequence of Theorems 3.9 and 3.12:

Corollary 3.14. *Consider a theory $\mathcal{T}_0$ that satisfies the restrictions of Theorem 3.9 or Theorem 3.12, constants a_1, a_2 and an extension symbol f of type $A \to B$ satisfying*

$$\mathsf{BSMon}(f) = \{\ a_1 <_A x <_A y <_A a_2 \ \to f(x) <_B f(y)\ \}.$$

Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{BSMon}(f)$ satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$, and thus it is local.

Proof: We can use the same completion $\overline{f}$ as in the proof of Theorem 3.9, except that $\overline{f}(x)$ can be arbitrary whenever $a_{1_M} <_A x <_A a_{2_M}$ does not hold.

If $<_{A_M}$ is a strict partial order, we first complete it like in the proof of Theorem 3.12 and then construct the completion $\overline{f}$ like in Theorem 3.9. □

3.3 Strict Boundedness

Like we did for monotonicity, we can also define a strict version of the boundedness axiom $\mathsf{Bound}^{\mathsf{t}}(f)$ (defined in Section 2.5). In the following, we will show that this also results in a local extension of suitable base theories.

Theorem 3.15. *Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n, B$, and a binary predicate $<_B$ of type $B \times B$ such that for every model M of $\mathcal{T}_0$, $<_{B_M}$ is an irreflexive relation and B_M is open wrt. $<_{B_M}$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$, and $t[x]$ be a term of sort B in the base signature, with x of sort A. Consider an extension symbol f of type $A \to B$ and an axiom*

$$\mathsf{SBound}^t(f) = \{\ f(x) <_B t[x]\ \}.$$

Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{SBound}^t(f)$ satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$, and thus it is local.

Proof: Let M be a countable weak partial model of $\mathcal{T}_0 \cup \mathsf{SBound}^t(f)$, where $f_M(x)$ is defined for $x \in D = \{\ a_1, \ldots, a_n\ \} \subseteq A_M$, and undefined otherwise, and all other functions are total. Since M is a partial model, we have $f_M(a_i) <_B t_M[a_i]$ for $a_i \in D$. We define

$$\overline{f}(a) = \begin{cases} f_M(a), & \text{if } a \in D \\ b \text{ with } b <_B t_M[a] \text{ arbitrary}, & \text{otherwise.} \end{cases}$$

Since B_M is open wrt. $<_{B_M}$, there is such a value b for every $a \in A_M \setminus D$. Clearly, the completion $\overline{f}$ of f satisfies $\mathsf{SBound}^t(\overline{f})$.

Thus, define the structure N to be the same as M, except that $f_N(x) = \overline{f}(x)$ for all x. N is a total structure and a model of $\mathcal{T}_0 \cup \mathsf{SBound}^t(f)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$. □

Example 3.16. Consider a term $t[x] = 5 \cdot x + 7$ in some arithmetical base theory $\mathcal{T} \in \{\ \mathcal{T}_{\mathbb{Z}}, \mathcal{T}_{\mathbb{Q}}, \mathcal{T}_{\mathbb{R}}\ \}$, and f of type $A \to A$, $A \in \{\ \mathbb{Z}, \mathbb{Q}, \mathbb{R}\ \}$. By Theorem 3.15, the extension of $\mathcal{T}$ with $\mathsf{SBound}^t(f)$ is local.

With different assumptions on the base theory, we can also prove locality for a strict version of $\mathsf{Bound}^{s,t}(f)$.

Theorem 3.17. *Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n, B$, and a binary predicate $<_B$ of type $B \times B$ such that for every model M of $\mathcal{T}_0$, $<_{B_M}$ is an irreflexive and transitive relation and B_M is dense wrt. $<_{B_M}$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$, and $s[x], t[x]$ be terms of sort B in the base signature with x of sort A and such that $\mathcal{T}_0 \models s[x] <_B t[x]$. Consider an extension symbol f of type $A \to B$ and*

$$\mathsf{SBound}^{s,t}(f) = \{\ s[x] <_B f(x) <_B t[x]\ \}.$$

Then the extension $\mathcal{T}_A \cup \mathcal{T}_B \subseteq \mathcal{T}_A \cup \mathcal{T}_B \cup \mathsf{SBound}^{s,t}(f)$ satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$, and thus it is local.

Proof: Let M be a weak partial model of $\mathcal{T}_0 \cup \mathsf{SBound}^{s,t}(f)$, where $f_M(x)$ is defined for $x \in D = \{\ a_1, \ldots, a_n\ \} \subseteq A_M$, and undefined otherwise, and all other functions are total. Since M is a partial model, we have $s_M[a_i] <_{B_M} f_M(a_i) <_{B_M} t_M[a_i]$ for $a_i \in D$. We define

$$\overline{f}(a) = \begin{cases} f_M(a), & \text{if } a \in D \\ b \text{ with } s_M[a] <_{B_M} b <_B t_M[a] \text{ arbitrary}, & \text{otherwise.} \end{cases}$$

Since $s_M[a] <_{B_M} t_M[a]$ for every $a \in A_M$ and B_M is dense wrt. $<_{B_M}$, we can find such a value b for every $a \in A_M \setminus D$. Clearly, the completion $\overline{f}$ of f satisfies $\mathsf{SBound}^t(\overline{f})$.

Thus, define the structure N to be the same as M, except that $f_N(x) = \overline{f}(x)$ for all x. N is a total structure and a model of $\mathcal{T}_0 \cup \mathsf{SBound}^t(f)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$. □

Example 3.18. Consider terms $s[x] = 5 \cdot x + 6$ and $t[x] = 5 \cdot x + 7$ in an arithmetical base theory $\mathcal{T} \in \{\ \mathcal{T}_{\mathbb{Q}}, \mathcal{T}_{\mathbb{R}}\ \}$, and f of type $A \to A$, $A \in \{\ \mathbb{Q}, \mathbb{R}\ \}$. By Theorem 3.17, the extension of $\mathcal{T}$ with $\mathsf{SBound}^{s,t}(f)$ is local.

3.4 Piecewise (Strict) Boundedness

The following is a generalization of $\mathsf{GBound}^{s,t}_{\phi}$ (see Section 2.5), which allows several axioms of this kind with mutually exclusive guards, and with possibly strict boundedness as defined in Section 3.3.

A combination of such axioms has been used to model update rules of parameterized systems [40] (see Sections 5.4 and 5.5 for example applications), and has later been extended to this general result.

Theorem 3.19. *[39, 58, 61, 36] Consider a theory $\mathcal{T}_0$ with sorts $A_1, \ldots, A_n, B$ and a binary predicate $\leq_B$ of type $B \times B$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$ and consider conjunctions $\phi_i[x]$, $1 \leq i \leq m$, of Π_0-literals, with x of sort A as their only variable, and Π_0-terms $s_i[x]$ and $t_i[x]$ of sort B with x of sort A.*

Let $\mathcal{T}_0$ be such that $\mathcal{T}_0 \cup \phi_i[x] \cup \phi_j[x] \models \Box$ for $i \neq j$, $\mathcal{T}_0 \models \phi_i[x] \to s_i[x] \leq_B t_i[x]$ for $1 \leq i \leq m$, and for every model M of $\mathcal{T}_0$, $\leq_M$ is either a partial order or a strict and dense partial order on B_M.

Consider an extension symbol f of type $A \to B$ and

$$\mathsf{PBound}(f) = \{\ \phi_i[x] \ \to\ s_i[x] \leq f(x) \leq t_i[x]\ \}_{1 \leq i \leq m}.$$

Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{PBound}(f)$ satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$, and thus it is local.

Proof: Let $\mathsf{PBound}(f)$ be a set of clauses satisfying the properties from above, and let M be a countable partial model of $\mathcal{T}_0 \cup \mathsf{PBound}(f)$, where everything except f_M is totally defined.

We extend M to a total structure by defining a completion $\overline{f}$ of f_M: Assume that $f_M(a)$ is undefined. As the $\phi_i[x]$ are mutually exclusive, at most one of the $\phi_i[a]$ can be satisfied by M. If this is the case for some i, define $\overline{f}(a)$ so that $s_{i_M}[a] \leq \overline{f}(a) \leq t_{i_M}[a]$. This is possible since $\mathcal{T}_0 \models \phi_i[x] \ \to\ s_i[x] \leq t_i[x]$ for all i, and B_M is dense wrt. $\leq_M$ if $\leq_M$ is a strict order. If none of the $\phi_i(a)$ are satisfied by M, we can freely define $\overline{f}(a)$ to an arbitrary value. Since the ϕ_i, s_i and t_i are in the base signature Π_0, definitions of $\overline{f}$ on different elements are independent and we can define $\overline{f}$ for all undefined points simultaneously.

The resulting function $\overline{f}$ satisfies $\mathsf{PBound}(f)$ for every possible value a of x: either $f_M(a)$ was already defined in M (and thus $\mathsf{PBound}(f)$ is also satisfied wrt. $\overline{f}$), or $\overline{f}(a)$ has been defined such that in every clause, either the antecedent is false or the succedent is true.

Thus, define the structure N to be the same as M, except that $f_N(x) = \overline{f}(x)$ for all x. N is a total structure and a model of $\mathcal{T}_0 \cup \mathsf{PBound}(f)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$. □

Example 3.20. Piecewise boundedness axioms can be used to define step-functions: consider an arithmetical base theory $\mathcal{T}_0$, an extension symbol f and the set of clauses

$$\mathcal{K} = \left\{ \begin{array}{ll} x < 0 & \to f(x) = 0 \\ x \geq 0 \ \wedge\ x < 2 & \to f(x) = 2 \\ x \geq 2 & \to f(x) = 4 \end{array} \right.$$

By Theorem 3.19 (with $s_1[x] = t_1[x] = 0$, $s_2[x] = t_2[x] = 2$ and $s_3[x] = t_3[x] = 4$), the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is local.

We present more applications of piecewise boundedness axioms in Section 3.8.2 and Chapter 5.

3.5 New Combinations of Monotonicity and Boundedness

In Section 2.5, we have seen that for suitable base theories the extension with a combination of $\mathsf{GMon}(\mathsf{f})$ and $\mathsf{Bound}^{\mathsf{t}}(\mathsf{f})$ is local if $\mathsf{GMon}(\mathsf{t})$ holds in the base theory. With the strict versions of monotonicity and boundedness, we can define several new combinations of this kind and prove their locality.

In Section 3.5.1, we consider two terms $s[x]$, $t[x]$ in the base signature that are (not necessarily strictly) monotone and satisfy $s[x] < t[x]$ in the base theory. We show that under suitable assumptions on the base theory, extensions with

a strictly monotone function f that is strictly bounded by $s[x]$ from below and by $t[x]$ from above are local.

In Section 3.5.2 we prove a similar locality result: there, we consider terms $s[x]$, $t[x]$ which are strictly monotone and satisfy $s[x] \leq t[x]$, and extend the base theory with a function that is strictly monotone and (not necessarily strictly) bounded by $s[x]$ and $t[x]$. That is, we show that locality of the extension is preserved when considering non-strict (instead of strict) boundedness between the terms and f, if we require the terms to be strictly monotone in the base theory. After that, we also show that locality is lost if we consider similar extensions with a strictly monotone f, where $s[x], t[x]$ are not strictly monotone and we consider non-strict boundedness.

Finally, in Section 3.5.3 we will show that extensions with (not necessarily strictly) monotone functions f with the same boundedness conditions as in Section 3.5.1 also satisfy a locality property.

3.5.1 Strict Monotonicity and Strict Boundedness

For the proof of locality, we will need the following lemma:

Lemma 3.21. *Let O_1 be a countable set with strict total order $<_1$ and O_2 a set with strict total order $<_2$ such that O_2 is open and dense wrt. $<_2$. If $f_s : O_1 \to O_2$ and $f_t : O_1 \to O_2$ are (not necessarily strictly) monotone partial functions with $f_s(x) <_2 f_t(x)$ for all $x \in O_1$ with $f_s(x)$ and $f_t(x)$ defined, then there is a (total) strictly monotone function $f : O_1 \to O_2$ such that $f_s(x) <_2 f(x)$ for all $x \in O_1$ with $f_s(x)$ defined, and $f(x) <_2 f_t(x)$ for all $x \in O_1$ with $f_t(x)$ defined.*

Proof: Since O_1 is countable, there exists a bijective map $i : \mathbb{N} \to O_1$. We show how to inductively define $f(i(n))$ for all $n \in \mathbb{N}$:

Case $n = 1$: Suppose f is nowhere defined on O_1, and let $i(1) = a$. Then choose $b \in O_2$ such that $f_s(a) <_2 b$ if $f_s(a)$ is defined and $b <_2 f_t(a)$ if $f_t(a)$ is defined. If both are undefined, choose $b \in O_2$ arbitrary. Such a b exists since by assumption $f_s(a) <_2 f_t(a)$ and O_2 is dense and open wrt. $<_2$. Let $f(a) := b$. Clearly, the mapping defined thus far is strictly monotone.

Case $n \to n+1$: Suppose $f : \{\ i(1), \ldots, i(n)\ \} \to O_2$ is strictly monotone, and we have $f_s(x) <_2 f(x)$ whenever $f_s(x)$ is defined, and $f(x) <_2 f_t(x)$ whenever $f_t(x)$ is defined (for $x \in \{\ i(1), \ldots, i(n)\ \}$). Let $a = i(n+1)$, and let

$$\begin{aligned} D^- = &\ \{\ f(i(j)) \mid 1 \leq j \leq n,\ i(j) <_1 a\ \}, \\ D^+ = &\ \{\ f(i(j)) \mid 1 \leq j \leq n,\ a <_1 i(j)\ \}. \end{aligned}$$

Then, let $l = max(D^- \cup \{f_s(a)\})$ if $f_s(a)$ is defined, or $l = max(D^-)$ otherwise. If $D^- = \varnothing$ and $f_s(a)$ is undefined, l is also undefined. Similarly, let $r = min(D^+ \cup \{f_t(a)\}$ if $f_t(a)$ is defined, or $r = min(D^+)$ otherwise. If $D^+ = \varnothing$ and $f_t(a)$ is undefined, r is also undefined. Note that only one of l and r can be undefined, since by assumption not both D^- and D^+ can be empty. Then, define

$$f(a) := \begin{cases} \text{arbitrary } b \in O_2 \text{ with } b <_2 r, & \text{if } l \text{ is undefined} \\ \text{arbitrary } b \in O_2 \text{ with } l <_2 b, & \text{if } r \text{ is undefined} \\ \text{arbitrary } b \in (l, r), & \text{otherwise} \end{cases}$$

By definition of l and r we have $l <_2 r$ if both are defined (f is strictly monotone on $\{\ i(1),\ldots,i(n)\ \}$, and we have $f_s(a) <_2 f_t(a)$ if both are defined). Since O_2 is dense and open wrt. $<_2$, we can always find such a b.

By assumption, $f : \{\ i(1),\ldots,i(n)\ \} \to O_2$ is strictly monotone and satisfies $f_s(x) <_2 f(x)$ whenever $f_s(x)$ is defined, and $f(x) <_2 f_t(x)$ whenever $f_t(x)$ is defined. It is easy to see that $f : \{\ i(1),\ldots,i(n),i(n+1)\ \} \to O_2$ is also strictly monotone and satisfies $f_s(x) <_2 f(x)$ whenever $f_s(x)$ is defined, and $f(x) <_2 f_t(x)$ whenever $f_t(x)$ is defined. □

Theorem 3.22. *Let $\mathcal{T}_0$ be a theory with sorts $A_1,\ldots,A_n,B$, and a binary predicate $<_B$ of type $B\times B$. Let $A = A_1^{k_1}\times\ldots\times A_n^{k_n}$, $<_A$ a binary predicate of type $A\times A$ and $s[x],t[x]$ terms of sort B in the base signature, with x of sort A as their only variable. Furthermore, let $\mathcal{T}_0$ be such that for every countable model M of $\mathcal{T}_0$,*

i) $<_{B_M}$ is a strict total order on B_M,

ii) B_M is dense and open wrt. $<_{B_M}$,

iii) $<_{A_M}$ is a strict total order on A_M,

iv) $\mathsf{Mon}(s_M)$ and $\mathsf{Mon}(t_M)$ hold, and

v) $s_M[a] <_{B_M} t_M[a]$ for all $a\in A_M$.

Consider an extension symbol f of type $A\to B$ satisfying

$$\mathcal{K} = \{\ \mathsf{SMon}(f),\ \mathsf{SBound}^{s,t}(f)\ \}.$$

Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0\cup\mathcal{K}$ satisfies ($\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}}$), and thus it is local.

Proof: Let M be a countable weak partial model of $\mathcal{T}_0\cup\mathcal{K}$, where $f_M(x)$ is defined for $x\in D = \{\ a_1,\ldots,a_n\ \}\subseteq A_M$, and undefined otherwise, and all other functions are total. Since M is a partial model, we have $f_M(a_i) <_{B_M} f_M(a_j)$ if $a_i <_A a_j$ and $a_i,a_j\in D$, and $s_M[a_i] <_{B_M} f_M(a_i) <_B t_M[a_i]$ for all $a_i\in D$.

Assume wlog. that $a_i <_{A_M} a_j$ if $i<j$. Then, we can partition A_M into D and open intervals $A_0,\ldots,A_n$ such that

$$\begin{array}{lll} A_0 & = \{\ a\in A_M \mid a <_{A_M} a_1\ \}, & \\ A_n & = \{\ a\in A_M \mid a >_{A_M} a_n\ \}, & \text{and} \\ A_i & = (a_i,a_{i+1}) & \text{for } 1\le i\le n-1. \end{array}$$

Similarly, we can partition B_M into open intervals $B_0,\ldots,B_n$ such that

$$\begin{array}{lll} B_0 & = \{\ b\in B_M \mid b <_{B_M} f_M(a_1)\ \}, & \\ B_n & = \{\ b\in B_M \mid b >_{B_M} f_M(a_n)\ \}, & \text{and} \\ B_i & = (f_M(a_i),f_M(a_{i+1})) & \text{for } 1\le i\le n-1. \end{array}$$

Since B_M is dense and open wrt. $<_{B_M}$, all B_i are non-empty open sets. For $0\le i\le n$, let $s_i : A_i\to B_i$ and $t_i : A_i\to B_i$ be the restrictions of (the functions defined by) s_M and t_M to A_i and B_i. That is, $s_i(x)$ is undefined if $s_M[x]\notin B_i$, and similarly for $t_i(x)$. Note that (by definition of the A_i,B_i) whenever $s_i(x)$ is undefined we must have $s_M[x]\le_{B_M} f_M(a_i)$, and whenever $t_i(x)$ is undefined we must have $f_M(a_{i+1})\le_{B_M} t_M[x]$.

By Lemma 3.21, there is a strictly monotone function $f_i : A_i \to B_i$ for every pair of intervals (A_i, B_i) and monotone functions $s_i : A_i \to B_i$ and $t_i : A_i \to B_i$, $0 \leq i \leq n$, such that $s_i(x) <_{B_M} f_i(x)$ for all $x \in A_i$ with $s_i(x)$ defined, and $f_i(x) <_{B_M} t_i(x)$ for all $x \in A_i$ with $t_i(x)$ defined. Now, define

$$\overline{f}(x) = \begin{cases} f_M(x) & \text{if } x \in D \\ f_i(x) & \text{if } x \in A_i \end{cases}$$

$\overline{f}$ is a completion of f_M. It satisfies $\mathsf{SMon}(\overline{f})$ because every $f_i : A_i \to B_i$ is a strictly monotone function, and $f_i(x) <_{B_M} f_M(a_{i+1})$ for every $x \in A_i$, $0 \leq i \leq n-1$.

It remains to be shown that $\overline{f}$ satisfies $\mathsf{SBound}^{s,t}(\overline{f})$: All f_i are such that $s_i(x) <_{B_M} f_i(x) <_{B_M} t_i(x)$ if both $s_i(x)$ and $t_i(x)$ are defined. For $x \in A_i$ with $s_i(x)$ undefined, we have $f_M(a_i) <_{B_M} f_i(x)$ and $s_M[x] \leq_{B_M} f_M(a_i)$, which implies $s_M[x] <_{B_M} f_i(x)$. Similarly, for $x \in A_i$ with $t_i(x)$ undefined, we have $f_i(x) <_{B_M} f_M(a_{i+1})$ and $f_M(a_{i+1}) \leq_{B_M} t_M[x]$, which implies $f_i(x) <_{B_M} t_M[x]$. Finally, for $x \in D$ we have $s_M[x] <_{B_M} f_M(x) <_{B_M} t_M[x]$ by assumption, so $\mathsf{SBound}^{s,t}(\overline{f})$ is satisfied for all $x \in A_M$.

Thus, define the structure N to be the same as M, except that $f_N(x) = \overline{f}(x)$ for all x. N is a total structure and a model of $\mathcal{T}_0 \cup \mathsf{SBound}^{s,t}(f)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$. □

The following corollary is a consequence of Theorem 3.22 and Lemma 3.21:

Corollary 3.23. *Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n, B$, and a binary predicate $<_B$ of type $B \times B$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$, $<_A$ a binary predicate of type $A \times A$, and $s[x], t[x]$ terms of sort B in the base signature, with x of sort A. Furthermore, let $\mathcal{T}_0$ be such that for every countable model M of $\mathcal{T}_0$,*

i) $<_{B_M}$ is a strict total order on B_M,

ii) B_M is dense and open wrt. $<_{B_M}$,

iii) $<_{A_M}$ is a strict total order on A_M,

iv) $\mathsf{Mon}(s_M)$ and $\mathsf{Mon}(t_M)$ hold, and

v) $s_M[a] <_{B_M} t_M[a]$ for all $a \in A_M$.

Consider an extension symbol f of type $A \to B$ satisfying either of

$$\begin{aligned} \mathcal{K}_1 &= \{\ \mathsf{SMon}(f),\ \mathsf{SBound}^t(f)\ \}, \\ \mathcal{K}_2 &= \{\ \mathsf{SMon}(f),\ t[x] <_B f(x)\ \}, \\ \mathcal{K}_3 &= \{\ \mathsf{SMon}(f),\ \mathsf{SBound}^{s,t}(f)\ \}. \end{aligned}$$

Then each of the extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_i$ (for $i \in \{\ 1, 2, 3\ \}$) satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$, and thus is local.

Proof: For $\mathcal{K}_3$, this follows directly Theorem 3.22 and Lemma 3.21. For $\mathcal{K}_1$ and $\mathcal{K}_2$, we can modify the proof of Theorem 3.22, with either only an upper or only a lower bound. By Lemma 3.21, the needed functions between intervals A_i and B_i exist for lower and upper bounds, so in particular they also exist for only one bound. □

Example 3.24. Consider the terms $s[x] = 5 \cdot x + 6$ and $t[x] = 5 \cdot x + 7$ in some arithmetical base theory, and an extension symbol f satisfying either of

i) $\{\ \mathsf{SMon}(f),\ \mathsf{SBound}^{s,t}(f)\ \}$

ii) $\{\ \mathsf{SMon}(f),\ \mathsf{SBound}^{t}(f)\ \}$

iii) $\{\ \mathsf{SMon}(f),\ s[x] <_B f(x)\ \}$.

By Theorem 3.22 and Corollary 3.23, the extensions of the following base theories with one of the sets above are local:

- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Z} \to \mathbb{Q}$
- $\mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Q} \to \mathbb{Q}$
- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Z} \to \mathbb{R}$
- $\mathcal{T}_{\mathbb{Q}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Q} \to \mathbb{R}$.

Extensions with n-ary extension symbols f with types like $\mathbb{Q}^n \to \mathbb{Q}$ are also local, but only for total orders such that $\mathsf{Mon}(s)$ and $\mathsf{Mon}(t)$ hold. We leave open whether the results of this Section can be generalized to partial orders on A and/or B. The simple solution from Theorem 3.12 is not sufficient, since we cannot be sure that $s[x]$ and $t[x]$ will also have the desired properties $\mathsf{Mon}(s)$ and $\mathsf{Mon}(t)$ wrt. the completion of a given partial order.

3.5.2 Strict Monotonicity and (Non-strict) Boundedness

In this section, we show a similar result as in Section 3.5.1. Instead of terms which are monotone in the base theory and are strict bounds for a strictly monotone extension function, we consider terms which are strictly monotone, but only consider non-strict boundedness.

Again, we will need a lemma to prove the locality result. It is similar to Lemma 3.21, except that here we consider strictly monotone partial functions f_s and f_t and show that there is a strictly monotone function f that is non-strictly bounded by f_s and f_t:

Lemma 3.25. *Let O_1 be a countable set with strict total order $<_1$ and O_2 a set with strict total order $<_2$ such that O_2 is open and dense wrt. $<_2$. If $f_s : O_1 \to O_2$ and $f_t : O_1 \to O_2$ are strictly monotone partial functions with $f_s(x) \leq_2 f_t(x)$ for all $x \in O_1$ with $f_s(x)$ and $f_t(x)$ defined, then there is a strictly monotone (total) function $f : O_1 \to O_2$ such that $f_s(x) \leq_2 f(x)$ for all $x \in O_1$ with $f_s(x)$ defined, and $f(x) \leq_2 f_t(x)$ for all $x \in O_1$ with $f_t(x)$ defined.*

Proof: Since O_1 is countable, there exists a bijective function $i : \mathbb{N} \to O_1$. We show how to inductively define $f(i(n))$ for all $n \in \mathbb{N}$:

Case $n = 1$: Suppose f is not defined for any $a \in O_1$, and let $i(1) = a$. Then choose $b \in O_2$ such that $f_s(a) \leq_2 b$ if $f_s(a)$ is defined and $b \leq_2 f_t(a)$ if $f_t(a)$ is defined. If both are undefined, choose $b \in O_2$ arbitrary. Such a b exists since by assumption $f_s(a) \leq_2 f_t(a)$. Let $f(a) := b$. Clearly, the function defined thus far is strictly monotone.

Case $n \to n+1$**:** Suppose $f : \{\ i(1), \ldots, i(n)\ \} \to O_2$ is strictly monotone, and we have $f_s(x) \leq_2 f(x)$ whenever $f_s(x)$ is defined, and $f(x) \leq_2 f_t(x)$ whenever $f_t(x)$ is defined (for $x \in \{\ i(1), \ldots, i(n)\ \}$). Let $a = i(n+1)$, and let

$$\begin{aligned} D^- &= \{\ f(i(j)) \mid 1 \leq j \leq n,\ i(j) <_1 a\ \}, \\ D^+ &= \{\ f(i(j)) \mid 1 \leq j \leq n,\ a <_1 i(j)\ \}. \end{aligned}$$

Then, let $l = max(D^- \cup \{f_s(a)\})$ if $f_s(a)$ is defined, or $l = max(D^-)$ otherwise. If $D^- = \varnothing$ and $f_s(a)$ is undefined, l is also undefined. Similarly, let $r = min(D^+ \cup \{f_t(a)\}$ if $f_t(a)$ is defined, or $r = min(D^+)$ otherwise. If $D^+ = \varnothing$ and $f_t(a)$ is undefined, r is also undefined. Note that only one of l and r can be undefined, since by assumption not both D^- and D^+ can be empty. Furthermore, if $l = r$ then we must have $l = f_s(a)$ and $r = f_t(a)$: since D^- only contains values $f(x)$ with $x <_1 a$ and $f(x) \leq_2 f_t(x)$ by assumption, we know that $max(D^-) <_2 f_t(a)$ (since f_t is strictly monotone), and with a similar argument we conclude that $f_s(a) <_2 min(D^+)$. Then, define

$$f(a) := \begin{cases} \text{arbitrary } b \in O_2 \text{ with } b <_2 r, & \text{if } l \text{ is undefined} \\ \text{arbitrary } b \in O_2 \text{ with } l <_2 b, & \text{if } r \text{ is undefined} \\ l & \text{if } l = r \text{ (both defined)} \\ \text{arbitrary } b \in (l, r), & \text{otherwise} \end{cases}$$

Since O_2 is dense and open wrt. $<_2$, we can always find such a b (when necessary).

By assumption, $f : \{\ i(1), \ldots, i(n)\ \} \to O_2$ is strictly monotone and satisfies $f_s(x) \leq_2 f(x)$ whenever $f_s(x)$ is defined, and $f(x) \leq_2 f_t(x)$ whenever $f_t(x)$ is defined. It is easy to see that $f : \{\ i(1), \ldots, i(n), i(n+1)\ \} \to O_2$ is also strictly monotone and satisfies $f_s(x) \leq_2 f(x)$ whenever $f_s(x)$ is defined, and $f(x) \leq_2 f_t(x)$ whenever $f_t(x)$ is defined. □

Like the previous lemma, the following theorem is similar to the corresponding result in Section 3.5.1, in this case Theorem 3.22. The difference is again that we consider strictly (instead of non-strictly) monotone terms, and non-strict (instead of strict) boundedness.

Theorem 3.26. *Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n, B$, and a binary predicate $<_B$ of type $B \times B$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$, $<_A$ a binary predicate of type $A \times A$ and $s[x], t[x]$ terms of sort B in the base signature, with x of sort A as their only variable. Furthermore, let $\mathcal{T}_0$ be such that for every countable model M of $\mathcal{T}_0$,*

i) $<_{B_M}$ is a strict total order on B_M,

ii) B_M is dense and open wrt. $<_{B_M}$,

iii) $<_{A_M}$ is a strict total order on A_M,

iv) $\mathsf{SMon}(s_M)$ and $\mathsf{SMon}(t_M)$ hold, and

v) $s_M[a] \leq_{B_M} t_M[a]$ for all $a \in A_M$.

Consider an extension symbol f of type $A \to B$ satisfying

$$\mathcal{K} = \{\ \mathsf{SMon}(f),\ \mathsf{Bound}^{s,t}(f)\ \}.$$

Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$, and thus it is local.

Proof: Let M be a countable weak partial model of $\mathcal{T}_0 \cup \mathcal{K}$, where $f_M(x)$ is defined for $x \in D = \{\ a_1, \ldots, a_n\ \} \subseteq A_M$, and undefined otherwise, and all other functions are total. Since M is a partial model, we have $f_M(a_i) <_{B_M} f_M(a_j)$ if $a_i <_{A_M} a_j$ and $a_i, a_j \in D$, and $s_M[a_i] \leq_{B_M} f_M(a_i) \leq_{B_M} t_M[a_i]$ for all $a_i \in D$.

Assume wlog. that $a_i <_{A_M} a_j$ if $i < j$. Then, we can partition A_M into D and open intervals $A_0, \ldots, A_n$ such that

$$\begin{array}{lll} A_0 & = \{\ a \in A_M \mid a <_{A_M} a_1\ \}, & \\ A_i & = (a_i, a_{i+1}), & \text{for } 1 \leq i \leq n-1, \textit{and} \\ A_n & = \{\ a \in A_M \mid a >_{A_M} a_n\ \}. & \end{array}$$

Similarly, we can partition B_M into open intervals $B_0, \ldots, B_n$ such that

$$\begin{array}{lll} B_0 & = \{\ b \in B_M \mid b <_{B_M} f_M(a_1)\ \}, & \\ B_i & = (f_M(a_i), f_M(a_{i+1})), & \text{for } 1 \leq i \leq n-1, \text{ and} \\ B_n & = \{\ b \in B_M \mid b >_{B_M} f_M(a_n)\ \}. & \end{array}$$

Since $<_{B_M}$ is dense and B_M is open, all B_i are non-empty. For $0 \leq i \leq n$, let $s_i : A_i \to B_i$ be the reduct of s_M to A_i, with $s_i(x)$ undefined if $s_M[x] \notin B_i$, and $t_i : A_i \to B_i$ the reduct of t_M to A_i, with $t_i(x)$ undefined if $t_M[x] \notin B_i$. Since s_M and t_M are monotone and by assumption $s_M[a_i] \leq_{B_M} f_M(a_i) \leq_{B_M} t_M[a_i]$, we know that for $x \in A_i$ with $s_i(x)$ undefined, we must have $s_M[x] \leq_{B_M} f_M(a_i)$, and for $x \in A_i$ with $t_i(x)$ undefined, we have $f_M(a_{i+1}) \leq_{B_M} t_M[x]$.

By Lemma 3.25, there is a strictly monotone function $f_i : A_i \to B_i$ for every pair of intervals (A_i, B_i) with strictly monotone partial functions $s_i : A_i \to B_i$ and $t_i : A_i \to B_i$, such that $s_i(x) \leq_{B_M} f_i(x)$ for all $x \in A_i$ with $s_i(x)$ defined, and $f_i(x) \leq_{B_M} t_i(x)$ for all $x \in A_i$ with $t_i(x)$ defined. Now, define

$$\overline{f}(x) = \begin{cases} f_M(x) & \text{if } x \in D \\ f_i(x) & \text{if } x \in A_i \end{cases}$$

$\overline{f}$ is a completion of f_M. It satisfies $\mathsf{SMon}(\overline{f})$ because every $f_i : A_i \to B_i$ is a strictly monotone function, and $f_i(a) < f_M(a_{i+1})$ for every $a \in A_i$, $1 \leq i \leq n$.

It remains to be shown that $\overline{f}$ satisfies $\mathsf{Bound}^{s,t}(\overline{f})$: All f_i are such that $s_i(x) \leq_{B_M} f_i(x) \leq_{B_M} t_i(x)$ if both $s_i(x)$ and $t_i(x)$ are defined. For $x \in A_i$ with $s_i(x)$ undefined, we have $f_M(a_i) <_{B_M} f_i(x)$ and $s_M[x] \leq f_M(a_i)$, which implies $s_M[x] \leq_{B_M} f_i(x)$. Similarly, for $x \in A_i$ with $t_i(x)$ undefined, we have $f_i(x) <_{B_M} f_M(a_{i+1})$ and $f_M(a_{i+1}) \leq_{B_M} t_M[x]$, which implies $f_i(x) \leq_{B_M} t_M[x]$. Finally, for $x \in D$ we have $s_M[x] \leq_{B_M} f_M(x) \leq_{B_M} t_M[x]$ by assumption, so $\mathsf{SBound}^{s,t}(\overline{f})$ is satisfied for all $x \in A$.

Thus, define the structure N to be the same as M, except that $f_N(x) = \overline{f}(x)$ for all x. N is a total structure and a model of $\mathcal{T}_0 \cup \mathsf{SBound}^{s,t}(f)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$. □

From Theorem 3.26 and Lemma 3.25, we can derive the following corollary.

Corollary 3.27. *Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n, B$, and a binary predicate $<_B$ of type $B \times B$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$, $<_A$ a binary predicate of type $A \times A$, and $s[x], t[x]$ terms of sort B in the base signature, with x of sort A as their only variable. Furthermore, let $\mathcal{T}_0$ be such that for every countable model M of $\mathcal{T}_0$,*

i) $<_{B_M}$ *is a strict total order on* B_M*,*

ii) B_M *is dense and open wrt.* $<_{B_M}$*,*

iii) $<_{A_M}$ *is a strict total order on* A_M*,*

iv) $\mathsf{SMon}(s_M)$ *and* $\mathsf{SMon}(t_M)$ *hold, and*

v) $s_M[a] \leq_{B_M} t_M[a]$ *for all* $a \in A_M$*.*

Consider an extension symbol f *of type* $A \to B$ *satisfying either of*

$$\begin{array}{ll} \mathcal{K}_1 = & \{\ \mathsf{SMon}(f),\ \mathsf{Bound}^t(f)\ \}, \\ \mathcal{K}_2 = & \{\ \mathsf{SMon}(f),\ t[x] \leq_B f(x)\ \}, \\ \mathcal{K}_3 = & \{\ \mathsf{SMon}(f),\ \mathsf{Bound}^{s,t}(f)\ \}. \end{array}$$

Then each of the extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_i$ *(for* $i \in \{\ 1,2,3\ \}$*) satisfies* $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$*, and thus is local.*

Proof: For $\mathcal{K}_3$, this follows directly Theorem 3.26. For $\mathcal{K}_1$ and $\mathcal{K}_2$, we can modify the proof of Theorem 3.26, with either only an upper or only a lower bound. By Lemma 3.25, the needed maps between intervals A_i and B_i exist for lower and upper bounds, so in particular they also exist for only one bound. □

Example 3.28. Consider the terms $s[x] = 5 \cdot x + 6$ and $t[x] = 5 \cdot x + 7$ in some arithmetical base theory, and an extension symbol f satisfying either of

i) $\{\ \mathsf{SMon}(f),\ \mathsf{Bound}^{s,t}(f)\ \}$

ii) $\{\ \mathsf{SMon}(f),\ \mathsf{Bound}^{t}(f)\ \}$

iii) $\{\ \mathsf{SMon}(f),\ s[x] \leq_B f(x)\ \}$.

By Theorem 3.26 and Corollary 3.27, the extensions of the following base theories with one of the sets above are local:

- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Z} \to \mathbb{Q}$
- $\mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Q} \to \mathbb{Q}$
- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Z} \to \mathbb{R}$
- $\mathcal{T}_{\mathbb{Q}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Q} \to \mathbb{R}$
- $\mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{R} \to \mathbb{R}$.

Again, extensions with n-ary extension symbols f with types like $\mathbb{R}^n \to \mathbb{R}$ are also local, but the restriction to total orders such that $\mathsf{SMon}(s)$ and $\mathsf{SMon}(t)$ are satisfied means that they are usable only in special cases.

As before, we leave open whether these results can be extended to partial orders on A and/or B.

Non-strict Monotonicity of Terms and Non-strict Boundedness

In Section 3.5.1 we have seen that theory extensions with a strictly monotone function f that is bounded by terms $s[x], t[x]$ in the base theory are local, if we assume non-strict monotonicity for the ground terms and strict boundedness between $s[x], f(x)$ and $t[x]$. In this section we have seen that the same holds if we relax the condition on boundedness by making it non-strict, if at the same time we require $s[x]$ and $t[x]$ to be strictly monotone.

We can easily show that we lose locality if we relax both conditions, i.e. if we have non-strictly monotone terms $s[x], t[x]$ and only require non-strict boundedness:

Example 3.29. Consider the base theory $\mathcal{T}_\mathbb{Q}$, terms $s[x] = t[x] = 0$ and $\mathcal{K} = \{ \mathsf{SMon}(f),\ \mathsf{Bound}^{s,t}(f) \}$. Then $\mathcal{T}_\mathbb{Q} \cup \mathcal{K} \models \square$, since there can be no strictly monotone function which is constantly 0. Thus, $\mathcal{T}_\mathbb{Q} \cup \mathcal{K} \cup G \models \square$ for any set of ground clauses G. However, if we consider $G = \{ 0 = 0 \}$, then $\mathcal{K}[G] = \varnothing$ and therefore $\mathcal{T}_\mathbb{Q} \cup \mathcal{K}[G] \cup G$ is satisfiable, showing that $\mathcal{T}_\mathbb{Q} \subseteq \mathcal{T}_\mathbb{Q} \cup \mathcal{K}$ is not local.

3.5.3 (Non-strict) Monotonicity and Strict Boundedness

In this section we show that we can combine non-strict monotonicity of an extension function with strict boundedness by terms in the base signature. Assumptions and results in this section are similar to Section 3.5.1, except that we allow the extension function to be non-strictly monotone.

In this case, we do not need to prove a lemma about existence of monotone functions that are strictly bounded by other monotone functions. Instead, we will use Lemma 3.21 again: it asserts existence of strictly monotone functions under the conditions we consider here, which are in particular monotone functions.

Theorem 3.30. *Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n, B$, and a binary predicate $<_B$ of type $B \times B$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$, $<_A$ a binary predicate of type $A \times A$ and $s[x], t[x]$ terms of sort B in the base signature, with x of sort A. Furthermore, let $\mathcal{T}_0$ be such that for every countable model M of $\mathcal{T}_0$,*

i) $<_{B_M}$ is a strict total order on B_M,

ii) B_M is dense and open wrt. $<_{B_M}$,

iii) $<_{A_M}$ is a strict total order on A_M,

iv) $\mathsf{Mon}(s_M)$ and $\mathsf{Mon}(t_M)$ hold, and

v) $s_M[a] <_{B_M} t_M[a]$ for all $a \in A_M$.

Consider an extension symbol f of type $A \to B$ satisfying

$$\mathcal{K} = \{ \mathsf{Mon}(f),\ \mathsf{SBound}^{s,t}(f) \}.$$

Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ satisfies ($\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}}$), and thus it is local.

Proof: Let M be a countable weak partial model of $\mathcal{T}_0 \cup \mathcal{K}$, where $f_M(x)$ is defined for $x \in D = \{ a_1, \ldots, a_n \} \subseteq A_M$, and undefined otherwise, and all other

functions are total. Since M is a partial model, we have $f_M(a_i) \leq_{B_M} f_M(a_j)$ if $a_i \leq_{A_M} a_j$ and $a_i, a_j \in D$, and $s_M[a_i] <_{B_M} f_M(a_i) <_{B_M} t_M[a_i]$ for all $a_i \in D$.

Assume wlog. that $a_i <_{A_M} a_j$ if $i < j$. Then, we can partition A_M into D and open intervals $A_0, \ldots, A_n$ such that

$$\begin{array}{lll} A_0 & = \{\ a \in A_M \mid a <_{A_M} a_1\ \}, & \\ A_i & = (a_i, a_{i+1}), & \text{for } 1 \leq i \leq n-1, \textit{and} \\ A_n & = \{\ a \in A_M \mid a >_{A_M} a_n\ \}. & \end{array}$$

Similarly, we can partition B_M into open intervals $B_0, \ldots, B_n$ such that

$$\begin{array}{lll} B_0 & = \{\ b \in B_M \mid b <_{B_M} f_M(a_1)\ \}, & \\ B_i & = (f_M(a_i), f_M(a_{i+1})), & \text{for } 1 \leq i \leq n-1, \text{ and} \\ B_n & = \{\ b \in B_M \mid b >_{B_M} f_M(a_n)\ \}. & \end{array}$$

Since $<_{B_M}$ is dense and B_M is open, a B_i is non-empty whenever $f_M(a_i) <_{B_M} f_M(a_{i+1})$. For $0 \leq i \leq n$ with $B_i \neq \varnothing$, let $s_i : A_i \to B_i$ be the reduct of (the function defined by) s_M to A_i, with $s_i(x)$ undefined if $s_M[x] \notin B_i$, and $t_i : A_i \to B_i$ the reduct of t_M to A_i, with $t_i(x)$ undefined if $t_M[x] \notin B_i$. Since s_M and t_M are monotone and by assumption $s_M[a_i] <_{B_M} f_M(a_i) <_{B_M} t_M[a_i]$, we know that for $x \in A_i$ with $s_i(x)$ undefined, we must have $s_M[x] <_{B_M} f_M(a_i)$, and for $x \in A_i$ with $t_i(x)$ undefined, we have $f_M(a_{i+1}) <_{B_M} t_M[x]$.

By Lemma 3.21, there is a strictly monotone function $f_i : A_i \to B_i$ for every pair of intervals (A_i, B_i) (if $B_i \neq \varnothing$) with strictly monotone partial functions $s_i : A_i \to B_i$ and $t_i : A_i \to B_i$, such that $s_i(x) <_{B_M} f_i(x)$ for all $x \in A_i$ with $s_i(x)$ defined, and $f_i(x) <_{B_M} t_i(x)$ for all $x \in A_i$ with $t_i(x)$ defined. Every f_i is in particular a monotone function. If $B_i = \varnothing$, define $f_i(x) = f_M(a_I)$ for all $x \in A_i$. Now, define

$$\overline{f}(x) = \begin{cases} f_M(x) & \text{if } x \in D \\ f_i(x) & \text{if } x \in A_i \end{cases}$$

$\overline{f}$ is a completion of f_M. It satisfies $\mathsf{Mon}(\overline{f})$ because every $f_i : A_i \to B_i$ is monotone, and $f_i(a) \leq f_M(a_{i+1})$ for every $a \in A_i$, $0 \leq i \leq n-1$.

It remains to show that $\overline{f}$ satisfies $\mathsf{SBound}^{s,t}(\overline{f})$. For intervals A_i with $B_i \neq \varnothing$, the f_i are such that $s_M[a] <_{B_M} f_i(a) <_{B_M} t_M[a]$ for every $a \in A_i$. For $B_i = \varnothing$, we have $A_i = (a_i, a_{i+1})$ with $f_M(a_i) = f_M(a_{i+1})$. Since by assumption $s_M[a_i] <_{B_M} f_M(a_i)$ and $f_M(a_{i+1}) <_{B_M} t_M(a_{i+1})$, by monotonicity of $s[x]$ and $t[x]$ it follows that $s_M[x] <_{B_M} f_i(x) <_{B_M} t_M[x]$ holds for every $x \in A_i$ also in this case. Finally, we have $s_M[x] <_{B_M} f_M(x) <_{B_M} t_M[x]$ for every $x \in D$ by assumption.

Thus, define the structure N to be the same as M, except that $f_N(x) = \overline{f}(x)$ for all x. N is a total structure and a model of $\mathcal{T}_0 \cup \mathsf{SBound}^{s,t}(f)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$. □

The following corollary is a consequence of Theorem 3.30 and Lemma 3.21.

Corollary 3.31. *Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n, B$, and a binary predicate $<_B$ of type $B \times B$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$, $<_A$ a binary predicate of type $A \times A$, and $s[x], t[x]$ terms of sort B in the base signature, with x of sort A as their only variable. Furthermore, let $\mathcal{T}_0$ be such that for every countable model M of $\mathcal{T}_0$,*

i) $<_{B_M}$ *is a strict total order on* B_M*,*

ii) B_M *is dense and open wrt.* $<_{B_M}$*,*

iii) $<_{A_M}$ *is a strict total order on* A_M*,*

iv) $\mathsf{Mon}(s_M)$ *and* $\mathsf{Mon}(t_M)$ *hold, and*

v) $s_M[a] <_{B_M} t_M[a]$ *for all* $a \in A_M$*.*

Consider an extension symbol f *of type* $A \to B$ *satisfying either of*

$$\begin{aligned}\mathcal{K}_1 &= \{\ \mathsf{Mon}(f),\ \mathsf{SBound}^t(f)\ \},\\ \mathcal{K}_2 &= \{\ \mathsf{Mon}(f),\ t[x] <_B f(x)\ \},\\ \mathcal{K}_3 &= \{\ \mathsf{Mon}(f),\ \mathsf{SBound}^{s,t}(f)\ \}.\end{aligned}$$

Then each of the extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_i$ *(for* $i \in \{\ 1,2,3\ \}$*) satisfies* $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$*, and thus is local.*

Proof: For $\mathcal{K}_3$, this follows directly from Theorem 3.30. For $\mathcal{K}_1$ and $\mathcal{K}_2$, we can modify the proof of Theorem 3.30, with either only an upper or only a lower bound. By Lemma 3.21, the needed maps between intervals A_i and B_i exist for lower and upper bounds, so in particular they also exist for only one bound. □

Example 3.32. Consider the terms $s[x] = 5 \cdot x + 6$ and $t[x] = 5 \cdot x + 7$ in some arithmetical base theory, and an extension symbol f satisfying either of

i) $\{\ \mathsf{Mon}(f),\ \mathsf{SBound}^{s,t}(f)\ \}$

ii) $\{\ \mathsf{Mon}(f),\ \mathsf{SBound}^{t}(f)\ \}$

iii) $\{\ \mathsf{Mon}(f),\ s[x] <_B f(x)\ \}$.

By Theorem 3.30 and Corollary 3.31, the extensions of the following base theories with one of the sets above are local:

- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Z} \to \mathbb{Q}$
- $\mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Q} \to \mathbb{Q}$
- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Z} \to \mathbb{R}$
- $\mathcal{T}_{\mathbb{Q}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Q} \to \mathbb{R}$.

Again, we can also consider extensions with n-ary extension symbols f with types like $\mathbb{Q}^n \to \mathbb{Q}$, but the restriction to total orders such that $\mathsf{Mon}(s)$ and $\mathsf{Mon}(t)$ hold means that these are useful only in special cases.

Once more, we leave open whether the results of this section can be generalized to partial orders on A and/or B.

3.6 Strict Monotonicity with Minimum Slope

For the verification of safety properties, another generalization of strict monotonicity proves to be useful [40] (see also Section 5.5.2): for a discrete domain, not only should there be some small increase between values of a given function f for subsequent elements, but the distance between these values should always be greater than a certain threshold. We provide here a generalization of the original result that also works for continuous domains. In this case, the requirement of a minimum distance d between function values $f(i)$ and $f(i+1)$ generalizes to a minimum slope of d for the function f.

We will first consider the case where this minimum slope is a constant, and then the case where the minimum slope is a function itself.

3.6.1 Constant Minimum Slope

Background theory. In the following we assume that the background theory $\mathcal{T}_0$ is a combination of arithmetic theories $\mathcal{T}_A$ and $\mathcal{T}_B$, where $\mathcal{T}_A \in \{\mathcal{T}_\mathbb{N}, \mathcal{T}_\mathbb{Z}, \mathcal{T}_\mathbb{Q}, \mathcal{T}_\mathbb{R}\}$ (with sort A), $\mathcal{T}_B \in \{\mathcal{T}_\mathbb{Q}, \mathcal{T}_\mathbb{R}\}$ (with sort B), possibly extended with a suitably axiomatized function symbol $\cdot$ of type $A \times B \to B$ for multiplication. If $\mathcal{T}_A = \mathcal{T}_B = \mathcal{T}_\mathbb{R}$ then we already have such a function symbol, otherwise we need to consider an extension of the combination $\mathcal{T}_A \cup \mathcal{T}_B$ with such a symbol.

To improve readability, in this section we omit subscripts that indicate sorts or evaluation in a specific model for function symbols from the signature of the background theory.

Theorem 3.33. *Let $\mathcal{T}_0$ be as described above. Consider a constant d of sort B with $\mathcal{T}_0 \models d \geq_B 0$, an extension symbol f of type $A \to B$ and*

$$\mathsf{SMonD}^d(f) = \{\ x <_A y \to f(y) - f(x) >_B (y - x) \cdot d\ \}.$$

Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{SMonD}^d(f)$ satisfies ($\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}}$), and thus it is local.

Proof: Let M be a countable weak partial model of $\mathcal{T}_0 \cup \mathsf{SMonD}^d(f)$, where $f_M(x)$ is defined for $x \in D = \{\ a_1, \ldots, a_n\ \} \subseteq A_M$, and undefined otherwise, and all other functions are total. Since M is a partial model, we have $f_M(a_j) - f_M(a_i) >_{B_M} (a_j - a_i) \cdot d_M$ for $a_i <_{A_M} a_j \in A$, where $(a_i - a_j) \cdot d_M \geq_{B_M} 0$.

Assume wlog. that $a_i <_{A_M} a_j$ if $i < j$. Then we can partition A_M into D and open intervals $A_0, \ldots, A_n$ such that

$$\begin{array}{lll} A_0 & = \{\ a \in A_M \mid a <_{A_M} a_1\ \}, & \\ A_i & = (a_i, a_{i+1}), & \text{for } 1 \leq i \leq n-1, \text{ and} \\ A_n & = \{\ a \in A_M \mid a >_{A_M} a_n\ \}. & \end{array}$$

Similarly, we can partition B_M into open intervals $B_0, \ldots, B_n$ such that

$$\begin{array}{lll} B_0 & = \{\ b \in B_M \mid b <_{B_M} f_M(a_1)\ \}, & \\ B_i & = (f_M(a_i), f_M(a_{i+1})), & \text{for } 1 \leq i \leq n-1, \text{ and} \\ B_n & = \{\ b \in B_M \mid b >_{B_M} f_M(a_n)\ \}. & \end{array}$$

For $1 \leq i < n$, let $d_i = \frac{f_M(a_{i+1}) - f_M(a_i)}{a_{i+1} - a_i}$, and choose $d_0 >_{B_M} d_M$, $d_n >_{B_M} d_M$ arbitrary from B_M. Note that with this definition, we have $d_i > d_M$ for $0 \leq i \leq n$. Define $f_i : A_i \to B_i$ as

$$\begin{array}{ll} f_i(x) = f_M(a_i) + (x - a_i) \cdot d_i, & \text{for } 1 \leq i \leq n, \text{ and} \\ f_0(x) = f_M(a_1) + (x - a_1) \cdot d_0. & \end{array}$$

That is, every f_i is a linear function with slope $d_i >_{B_M} d_M$. Now define

$$\overline{f}(x) = \begin{cases} f_M(x) & \text{if } x \in D \\ f_i(x) & \text{if } x \in A_i \end{cases}$$

$\overline{f}$ is a completion of f_M. We need to show that it satisfies $\mathsf{SMonD}^d(\overline{f})$.

This is clear for every interval A_i, since for every f_i, the derivative is $d_i > d_M$. Now consider an arbitrary a_i and the adjacent intervals A_{i-1}, A_i.

For $x \in A_i$ (i.e., $a_i <_{A_M} x$), we have

$$\overline{f}(x) = f_i(x) = f_M(a_i) + (x - a_i) \cdot d_i.$$

Therefore,

$$\begin{aligned} \overline{f}(x) - \overline{f}(a_i) &= f_M(a_i) + (x - a_i) \cdot d_i - f_M(a_i) \\ &= (x - a_i) \cdot d_i \\ &>_{B_M} (x - a_i) \cdot d_M. \end{aligned}$$

For $x \in A_{i-1}$ (i.e., $x <_{A_M} a_i$) and $1 < i \leq n$, we have

$$\overline{f}(x) = f_{i-1}(x) = f_M(a_{i-1}) + (x - a_{i-1}) \cdot d_{i-1}.$$

Therefore,

$$\begin{aligned} \overline{f}(a_i) - \overline{f}(x) &= f_M(a_i) - (f_M(a_{i-1}) + (x - a_{i-1}) \cdot d_{i-1}) \\ &= \underbrace{f_M(a_i) - f_M(a_{i-1})}_{=(a_i - a_{i-1}) \cdot d_{i-1}} - (x - a_{i-1}) \cdot d_{i-1} \\ &= (a_i - a_{i-1} - x + a_{i-1}) \cdot d_{i-1} \\ &= (a_i - x) \cdot d_{i-1} \\ &> (a_i - x) \cdot d_M \end{aligned}$$

Finally, for $x \in A_0$ (i.e., $i = 1$ and $x <_{A_M} a_1$), we have

$$\overline{f}(x) = f_0(x) = f_M(a_1) + (x - a_1) \cdot d_0.$$

Therefore,

$$\begin{aligned} \overline{f}(a_1) - \overline{f}(x) &= f_M(a_1) - (f_M(a_1) + (x - a_1) \cdot d_0) \\ &= (a_1 - x) \cdot d_0 \\ &>_{B_M} (a_1 - x) \cdot d_M, \end{aligned}$$

which shows that $\mathsf{SMonD}^d(\overline{f})$ is satisfied.

Thus, define the structure N to be the same as M, except that $f_N(x) = \overline{f}(x)$ for all x. N is a total structure and a model of $\mathcal{T}_0 \cup \mathsf{SMonD}^d(f)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$. □

As for strict monotonicity without distances, a bounded version of this axiom also satisfies a locality property:

Corollary 3.34. *Consider a base theory $\mathcal{T}_0$ as in Theorem 3.33, constants a_1, a_2 of sort A and an extension symbol f of type $A \to B$ satisfying*

$$\mathsf{BSMonD}^d(f) = \{\ a_1 <_A x <_A y <_A a_2 \ \to f(y) - f(x) >_B (y - x) \cdot d\ \}.$$

The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{BSMonD}^d(f)$ satisfies $(\mathsf{Comp}_{\mathsf{w}})$, and thus it is local.

Proof: We can use the same completion as in the proof of Theorem 3.33, except that $\overline{f}(x)$ is arbitrary whenever $a_{1_M} <_{A_M} x <_{A_M} a_{2_M}$ does not hold. □

Note that if $d = 0$, then SMonD^d is equivalent to SMon, and BSMonD^d is equivalent to BSMon.

Restriction to linear fragments. Note that we do not need the additional multiplication function if we restrict d to be a rational constant with fixed value in $\mathcal{T}_0$. In this case, the right-hand side of the implication SMonD^d is equivalent to an inequality which can be expressed by addition instead of multiplication. E.g., if $d = 2$ then the right-hand side of the implication is equivalent to $f(y)-f(x) > (y-x)+(y-x)$, and if $d = 1/2$ it is equivalent to $(f(y)-f(x))+(f(y)-f(x)) > y - x$.

Example 3.35. Consider some arithmetical base theory and an extension symbol f satisfying

$$x < y \rightarrow f(y) - f(x) > (y - x) \cdot 5.$$

By Theorem 3.33, the extensions of the following base theories with the axiom above are local:

- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Z} \rightarrow \mathbb{Q}$
- $\mathcal{T}_{\mathbb{Q}}$, with f of type $\mathbb{Q} \rightarrow \mathbb{Q}$
- $\mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Z} \rightarrow \mathbb{R}$
- $\mathcal{T}_{\mathbb{Q}} \cup \mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{Q} \rightarrow \mathbb{R}$
- $\mathcal{T}_{\mathbb{R}}$, with f of type $\mathbb{R} \rightarrow \mathbb{R}$.

In Section 5.5.2 we will see that even if d does not have a fixed value, a decision procedure for a linear fragment of arithmetic is sufficient for certain problems.

3.6.2 Variable Minimum Slope

For discrete domains of f, this result can be generalized by considering not a fixed minimum distance between values of f for adjacent elements, but to consider a term $d[x]$ (that is positive for all $x \in A$) in the base theory such that the distance between $f(i)$ and $f(i + 1)$ should always be greater than $d[i]$.

Background theory. In the following we assume that the background theory $\mathcal{T}_0$ is a combination of arithmetic theories $\mathcal{T}_A$ and $\mathcal{T}_B$, where $\mathcal{T}_A \in \{\mathcal{T}_{\mathbb{N}}, \mathcal{T}_{\mathbb{Z}}, \mathcal{T}_{\mathbb{Q}}, \mathcal{T}_{\mathbb{R}}\}$ (with sort A), $\mathcal{T}_B \in \{\mathcal{T}_{\mathbb{Q}}, \mathcal{T}_{\mathbb{R}}\}$ (with sort B), possibly extended with a suitably axiomatized function symbol $\sum$ of type $A \times A \times T(x) \rightarrow B$ for summation of terms, where $T(x)$ is the set of all terms $t[x]$ of sort B with variable x of sort A in the base theory.

Theorem 3.36. *Let $\mathcal{T}_0$ be as described above. Let $d[x] \in T(x)$ with $\mathcal{T}_0 \models d[x] \geq_B 0_B$, and consider an extension symbol f of type $A \rightarrow B$ satisfying*

$$\mathsf{SMonVD}^d(f) = \{\ x <_A y \rightarrow f(y) - f(x) >_B \textstyle\sum_{k=x}^{y-1} d[k]\ \}.$$

The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{SMonVD}^t(f)$ satisfies ($\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}}$), and thus it is local.

Proof: Let M be a countable weak partial model of $\mathcal{T}_0 \cup \mathsf{SMonVD}^d(f)$, where $f_M(x)$ is defined for $x \in D = \{\ a_1, \ldots, a_n\ \} \subseteq A_M$, and undefined otherwise, and all other functions are total. Assume that $a_i <_{A_M} a_j$ if $i < j$.

Since M is a partial model, we have $f_M(a_j) - f_M(a_i) >_{B_M} \sum_{k=a_i}^{a_j-1} d[k]$ for $a_i <_{A_M} a_j \in A$, where $d[k] \geq_{B_M} 0$ for all $k \in A_M$. Because of density of B_M, there exists $\varepsilon \in B_M$ such that $f_M(a_j) - f_M(a_i) >_{B_M} \sum_{k=a_i}^{a_j-1}(d[k] + \varepsilon)$ whenever $a_i <_{A_M} a_j \in A_M$.

Based on this ε, we define the following completion $\overline{f}$ of f_M:

- for $x <_{A_M} a_1$, let $\overline{f}(x) = f_M(a_1) - \sum_{i=x}^{a_1-1}(d[i] + \varepsilon)$.
- for $x \geq a_1$, define inductively:

  (i) $\overline{f}(a_1) = f_M(a_1)$, and

  (ii) $\overline{f}(x+1) = \begin{cases} f_M(x+1), & \text{if } f_M(x+1) \text{ is defined} \\ \overline{f}(x) + d[x] + \varepsilon, & \text{otherwise} \end{cases}$

We show that this completion satisfies $\mathsf{SMonVD}^d(\overline{f})$ for all $a <_{A_M} a'$ in A: if $a <_{A_M} a' <_{A_M} a_1$, then

$$\begin{aligned} \overline{f}(a') - \overline{f}(a) &= f_M(a_1) - \textstyle\sum_{k=a'}^{a_1-1}(d[k]+\varepsilon) - (f_M(a_1) - \sum_{k=a}^{a_1-1}(d[k]+\varepsilon)) \\ &= -\textstyle\sum_{k=a'}^{a_1-1}(d[k]+\varepsilon) + \sum_{k=a}^{a_1-1}(d[k]+\varepsilon) \\ &= \textstyle\sum_{k=a}^{a'-1}(d[k]+\varepsilon) \\ &>_{B_M} \textstyle\sum_{k=a}^{a'-1} d[k] \end{aligned}$$

and

$$\begin{aligned} \overline{f}(a_1) - \overline{f}(a) &= f_M(a_1) - (f_M(a_1) - \textstyle\sum_{k=a}^{a_1-1}(d[k]+\varepsilon)) \\ &= \textstyle\sum_{k=a}^{a_1-1}(d[k]+\varepsilon) \\ &>_{B_M} \textstyle\sum_{k=a}^{a_1-1} d[k]. \end{aligned}$$

For the case $a_i <_{A_M} a <_{A_M} a' <_{A_M} a_j$, $\mathsf{SMonVD}^d(\overline{f})$ follows immediately from the definition of $\overline{f}$. Furthermore, for $a \in (a_i, a_{i+1})$ (or $a > a_n$), we have

$$\begin{aligned} \overline{f}(a) - \overline{f}(a_i) &= f_M(a_i) + \textstyle\sum_{k=a_i}^{a-1}(d[k]+\varepsilon) - f_M(a_i) \\ &= \textstyle\sum_{k=a_i}^{a-1}(d[k]+\varepsilon) \\ &>_{B_M} \textstyle\sum_{k=a_i}^{a-1} d[k] \end{aligned}$$

and

$$\begin{aligned} \overline{f}(a_{i+1}) - \overline{f}(a) &= \underbrace{f_M(a_{i+1})}_{>f_M(a_i)+\sum_{k=a_i}^{a_{i+1}-1}(d[k]+\varepsilon))} - (f_M(a_i) + \textstyle\sum_{k=a_i}^{a-1}(d[k]+\varepsilon)) \\ &> f_M(a_i) + \textstyle\sum_{k=a_i}^{a_{i+1}-1}(d[k]+\varepsilon)) - (f_M(a_i) + \sum_{k=a_i}^{a-1}(d[k]+\varepsilon)) \\ &= \textstyle\sum_{k=a_i}^{a_{i+1}-1}(d[k]+\varepsilon)) - \sum_{k=a_i}^{a-1}(d[k]+\varepsilon)) \\ &= \textstyle\sum_{k=a}^{a_{i+1}-1}(d[k]+\varepsilon) \\ &>_{B_M} \textstyle\sum_{k=a}^{a_{i+1}-1} d[k] \end{aligned}$$

From these results and the assumption that $f_M(a_i) - f_M(a_j) >_{B_M} \sum_{k=a_i}^{a_j-1}(d[k]+\varepsilon)$

whenever $a_i <_{A_M} a_j$, it easily follows that $f_M(b) - f_M(a) >_{B_M} \sum_{k=a}^{b-1}(d[k] + \varepsilon)$ holds also when $a <_{A_M} a_i \leq_{A_M} a_j <_{A_M} b$.

Thus, define the structure N to be the same as M, except that $f_N(x) = \overline{f}(x)$ for all x. N is a total structure and a model of $\mathcal{T}_0 \cup \mathsf{SMonVD}^d(f)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$. □

Finally, we can again use a bounded version of this axiom:

Corollary 3.37. *Consider a base theory $\mathcal{T}_0$ and a term $d[x]$ as described above, constants $a_1, a_2 \in A$ and an extension symbol f of type $A \to \mathbb{R}$ satisfying*

$$\mathsf{BSMonVD}^d(f) = \{\ a_1 <_A x <_A y <_A a_2 \ \to\ f(y) - f(x) >_B \textstyle\sum_{k=x}^{y-1} d[k]\ \}.$$

The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{BSMonVD}^t(f)$ satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$, and thus it is local.

Proof: We can use the same completion as in the proof of Theorem 3.36, except that $\overline{f}(x)$ is arbitrary whenever $a_{1_M} <_{A_M} x <_{A_M} a_{2_M}$ does not hold. □

Example 3.38. Consider a base theory $\mathcal{T}_0$ as described above and the term $d[x] = 2 \cdot |x| + 4$. By Theorem 3.33, the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{SMonVD}^d(f)$ is local.

Similar to what we said for extensions with $\mathsf{SMonD}^d(f)$, we will see in Section 5.5.2 that for certain satisfiability problems we do not need the summation function in the base theory, as long as sums with a fixed number of arguments can be expressed.

Continuous domains

It should be possible to generalize the result of Theorem 3.36 also to functions with continuous domains by considering an integration function $\int : A \times A \times T(x) \to B$ instead of the summation function. We leave this as an open problem here.

3.7 Quasi-Monotone Functions

We introduce a generalization of the known locality results for (non-strict) monotone functions [61]: instead of defining monotonicity of a function with respect to a partial order on its domain, we can use an arbitrary transitive relation on the domain. The proof of the following theorem is mainly a demonstration that the original proof also works without assuming reflexivity or antisymmetry.

Theorem 3.39. *Let $\mathcal{T}_0$ be a theory with sorts $A_1, \ldots, A_n, B$, and a binary predicate $\leq_B$ of type $B \times B$. Let $A = A_1^{k_1} \times \ldots \times A_n^{k_n}$ and R a binary predicate of type $A \times A$.*[4] *Let $\mathcal{T}_0$ be such that in every model M of $\mathcal{T}_0$, R_M is transitive and $(B_M, \leq_M)$ is either a totally ordered set, a $\vee$-semilattice with 0 or a $\wedge$-semilattice with 1. Consider an extension symbol f of type $A \to B$, satisfying*

$$\mathsf{IMon}_t(f) = \{\ R(x, y)\ \to\ f(x) \leq f(y)\ \}.$$

The extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathsf{IMon}_t(f)$ satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$, and therefore it is local.

[4] Again, R can be a predicate symbol in the signature of $\mathcal{T}_0$, or defined by a formula in this signature.

Proof: Let M be a countable partial model of $\mathcal{T}_0 \cup \mathsf{IMon}_t(f)$, where everything except f_M is totally defined.

First, assume that $\leq_M$ is a total order on B_M. We extend M to a total structure by defining a completion $\overline{f}$ of f_M. For $a \in A_M$, let $DFR(a) = \{\ f_M(y) \mid f_M(y)$ defined and either $R_M(y,a)$ or $y = a\ \}$ and $DF = \{\ f_M(y) \mid f_M(y)$ defined $\}$. If $DF = \varnothing$, let $c \in B_M$ arbitrary. Then let

$$\overline{f}(a) = \begin{cases} max(DFR(a)) & \text{if } DFR(a) \neq \varnothing \\ min(DF) & \text{if } DFR(a) = \varnothing \ \wedge\ DF \neq \varnothing \\ c & \text{else.} \end{cases}$$

We need to show that $\mathsf{IMon}(\overline{f})$ is satisfied. Let $a, b \in A_M$ arbitrary with $R_M(a,b)$. In case $DF = \varnothing$, we immediately get $\overline{f}(a) = c = \overline{f}(b)$. Otherwise, if $DFR(a)$ is empty, $\overline{f}(a)$ is $min\{\ f_M(y) \mid f_M(y)$ defined $\}$, and $\overline{f}(b)$ is equal to some $f_M(y)$ that was defined, so we have $\overline{f}(a) \leq \overline{f}(b)$. If $DFR(a)$ is not empty, then we have $DFR(a) \subseteq DFR(b)$ (because of $R_M(a,b)$ and transitivity of R_M), and thus

$$\overline{f}(a) = max(DFR(a)) \leq max(DFR(b)) = \overline{f}(b).$$

If $(B_M, \leq_M)$ is a $\vee$-semilattice with 0, we define

$$\overline{f}(a) = sup(DFR(a)).$$

In a $\vee$-semilattice with 0 this supremum always exists — it is 0 if $DFR(a)$ is empty. For arbitrary elements $a, b \in A_M$ with $R_M(a,b)$, we again have $DFR(a) \subseteq DFR(b)$, and thus

$$\overline{f}(a) = sup(D(a)) \leq sup(D(b)) = \overline{f}(b).$$

Finally, the case with $(B_M, \leq_M)$ a $\wedge$-semilattice with 1 is dual: we define $DFR'(a) = \{\ f_M(y) \mid R_M(a,y)$ and $f_M(y)$ defined $\}$, and

$$\overline{f}(a) = inf(DFR'(a)).$$

This infimum always exists and is 0 if $DFR'(a)$ is empty. For arbitrary elements $a, b \in A_M$ with $R_M(a,b)$, we now have $DFR'(a) \supseteq DFR'(b)$, and thus

$$\overline{f}(a) = inf(DFR'(a)) \leq inf(DFR'(b)) = \overline{f}(b).$$

Thus, define the structure N to be the same as M, except that $f_N(x) = \overline{f}(x)$ for all x. N is a total structure and a model of $\mathcal{T}_0 \cup \mathsf{IMon}_t(f)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory extension satisfies $(\mathsf{Comp}^{\mathsf{c}}_{\mathsf{w}})$. □

Example 3.40. Consider a theory $\mathcal{T}_0$ with a total order $\leq$ and a monotone function f. Then the relation R defined by

$$R(x,y) \ \Leftrightarrow\ f(x) \leq f(y)$$

is transitive. Thus, by Theorem 3.39, the extension of $\mathcal{T}_0$ with an extension symbol f' satisfying

$$f(x) \leq f(y) \rightarrow f'(x) \leq f'(y)$$

is local.

3.8 Recursively Defined Data Structures

Recursive data structures can be seen as a constructive alternative to pointer data structures. Recursive definition of data structures is supported in many programming languages and is the predominant paradigm in functional programming languages.

Automated reasoning about recursive data structures has already been studied by Nelson and Oppen [48, 51]. Their decision procedures, based on (bi-directional) congruence closure, are still state of the art today. Recently, their results have been extended to allow also reasoning about numerical properties of such data structures [69], e.g. the length of lists.

In this section, we do not give new decidability results, nor a decision procedure that is generally more efficient than the existing ones. However, expressing the existing results in the framework of local theory extensions has several benefits:

- In local theory extensions, we can use the hierarchical reasoning method introduced in Section 2.3, which separates instantiation of the extension axioms from reasoning in the base theory. In this way, the overall decision procedure can be improved in a modular way: either by more efficient instantiation of axioms (as we will show in Chapter 4), or by improvements in the underlying decision procedure for the base theory (in this case the empty theory, i.e. the theory of equality).

- By expressing several known decidability results in the local reasoning framework, we can often identify further extensions or combinations of these results within this framework. E.g., we have shown [36] that the decidable pointer fragment introduced by Necula and McPeak [46] can be expressed and extended in the local reasoning framework, and can be combined with other local theory extensions to verify properties of systems based on pointer data structures. Further extensions of these results are possible and will be published in the near future.

  Similarly, the locality results for recursive data structures we present here can be extended. An extension of the results given here has recently been published [60].

First locality results for recursive data structures have been established recently [25, 56]: it has been shown that tail recursive definitions can be expressed by shallow extensions and therefore define stably local theory extensions (see Section 2.5), and that theory extensions with selector functions for an existing constructor in the base theory are local (again, see Section 2.5).

Here, we extend these locality results: we will show that there are axiomatizations for a combined theory of constructors and selectors that satisfy (stable) locality conditions.

3.8.1 Stable Locality of Recursive Data Structures

Theorem 3.41. *Consider a theory $\mathcal{T}$ with sort A, an n-ary constructor c of type $A^n \to A$ and n unary selectors s_i of type $A \to A$, satisfying*

$$
\begin{array}{ll}
(\mathsf{Sel}) & \left\{ \begin{array}{l} s_1(c(x_1,\ldots,x_n)) = x_1 \\ \ldots \\ s_n(c(x_1,\ldots,x_n)) = x_n \end{array} \right. \\
(\mathsf{Inv}) & c(s_1(x),\ldots,s_n(x)) = x.
\end{array}
$$

The theory $\mathcal{T}$ satisfies ($\mathsf{Emb}^{\mathsf{c}}$)*, and thus it is stably local.*

Proof. We prove stable locality of $\mathcal{T}$ by showing that it satisfies ($\mathsf{Emb}^{\mathsf{c}}$) (when considering it as an extension of the empty theory).

Let M be a countable Evans partial model of $(\mathsf{Sel}) \cup (\mathsf{Inv})$, where c_M is defined for finitely many elements of A_M^n and each of s_{i_M} is defined for finitely many elements of A_M. We note that in order to satisfy (Sel), c_M must be injective (where it is defined): if there are $a \neq b \in A_M$ with $c_M(a,\ldots) = c_M(b,\ldots)$ then (Sel) implies that $s_{1_M}(c_M(a,\ldots)) = a$ and $s_{1_M}(c_M(b,\ldots)) = b$, contradicting functionality of s_1. In the same way, the s_{i_M} must satisfy the condition $\bigwedge_{i=1}^{n} s_{i_M}(x) = s_{i_M}(y) \rightarrow x = y$ for all $a, b \in A_M$ where all $s_{i_M}(a), s_{i_M}(b)$ are defined. If we had $\bigwedge_{i=1}^{n} s_{i_M}(a) = s_{i_M}(b)$ for $a \neq b$, then (Inv) could not be satisfied in M for both a and b, because then (by Definition 2.48) we would have $a = c_M(s_{1_M}(a),\ldots,s_{n_M}(a))$ and $b = c_M(s_{1_M}(b),\ldots,s_{n_M}(b))$, as well as $c_M(s_{1_M}(a),\ldots,s_{n_M}(a)) = c_M(s_{1_M}(b),\ldots,s_{n_M}(b))$. The same argument holds for any total model of $(\mathsf{Sel}) \cup (\mathsf{Inv})$.

We define completions $\overline{c}$ of c_M and $\overline{s_i}$ of the s_{i_M}, considering the following cases:

i) $|A_M| = 1$,

ii) $|A_M| \geq 2$ and $n = 1$, and

iii) $|A_M| \geq 2$ and $n \geq 2$.

In case i), we can trivially complete any partial model to a total model by defining every possible function value to be equal to the single element.

In case ii), we complete M to a total model of the same size by first defining $\overline{c}(x) = y$ and $\overline{s}(y) = x$ whenever $c_M(x)$ is defined and equal to y. Note that in an Evans partial model of $(\mathsf{Sel}) \cup (\mathsf{Inv})$ $c_M(x)$ is defined and equal to y if and only if $s_{1_M}(y)$ is defined and equal to x.

If the resulting functions are not total, let $A_1 \subseteq A_M$ be the set of elements for which $\overline{c}$ is not defined, and $A_2 \subseteq A_M$ be the set of elements for which $\overline{s_1}$ is not defined. By definition of $\overline{c}$ and $\overline{s_1}$, we have $|A_1| = |A_2|$. Let $f : A_1 \rightarrow A_2$ be an arbitrary bijection, and define $\overline{c}(x) = f(x)$ for all $x \in A_1$, and $\overline{s_1}(x) = f^{-1}(x)$ for all $x \in A_2$. The resulting functions $\overline{c}$ and $\overline{s_1}$ are complete on A, and direct inverses. Thus, they satisfy (Sel) and (Inv).

In case iii), existence of more than one element and injectivity of an n-ary constructor implies that in any total model M' of $\mathcal{T}$, $A_{M'}$ must have infinitely many elements. Thus, if A_M is finite, let $\overline{A} \supseteq A_M$ be a countably infinite set of elements. In order to find completions of c_M and the s_{i_M}, we define a bijection $\overline{s} : \overline{A} \rightarrow \overline{A}^n$. Since $\overline{A}$ is countable, there exists a bijection $i : \mathbb{N} \rightarrow \overline{A}$. Let $<$ be the well-founded total order on $\overline{A}$ induced by this bijection and the usual order on $\mathbb{N}$. Let $\mathsf{Dom}(c_M) = \{\ a \in A_M^n \mid c_M(a) \text{ defined } \}$ and $\mathsf{Codom}(c_M) = \{\ a \in A_M \mid c_M(b) = a \text{ defined for some } b \in A_M^n\ \}$. We define $\overline{s}(i(m))$ inductively for all $m \in \mathbb{N}$:

Case $m = 1$: Assume $\overline{s}$ is not defined for any $a \in \overline{A}$. If $i(1) \in \mathsf{Codom}(c_M)$, then there is $a \in \mathsf{Dom}(c_M)$ such that $c_M(a) = i(1)$. Because of injectivity of c_M (on its domain of definition), this a is unique. Define $\overline{s}(i(1)) = a$. If $i(1) \notin \mathsf{Codom}(c_M)$, then s_{i_M} is undefined for $1 \leq i \leq n$, since M is an Evans partial model of $(\mathsf{Sel}) \cup (\mathsf{Inv})$ $c_M(x)$. In this case, define $\overline{s}(i(1)) = a'$, where a' is the smallest element (wrt. to the order $<$) of $\overline{A}^n$ with $a' \notin \mathsf{Dom}(c_M)$.
Case $m \to m+1$: Assume $\overline{s}$ is defined for $\{\ i(1), \ldots, i(m)\ \}$ such that $\overline{s} : \{\ i(1), \ldots, i(m)\ \} \to \overline{A}^n$ is injective. Define $\overline{s}(i(m+1))$ as above, except that in case $i(m+1) \notin \mathsf{Codom}(c_M)$, we require that $a' \notin \mathsf{Dom}(c_M) \cup \{\ i(1), \ldots, i(m)\ \}$. Because of this choice and injectivity of c_M, injectivity of $\overline{s} : \{\ i(1), \ldots, i(m+1)\ \} \to \overline{A}^n$ is immediate.

Since we always choose the smallest possible element from $\overline{A}^n$ wrt. $<$ (and $<$ is well-founded), the resulting function will also be surjective, and thus bijective. Now, define $\overline{c} = \overline{s}^{-1}$, and $\overline{s_i}(x) = x_i$ whenever $\overline{s}(x) = (x_1, \ldots, x_n)$. By definition of $\overline{s}$, these functions are completions of c_M and the s_{i_M}, respectively. Since $\overline{c}$ and $\overline{s}$ are inverse, the completions satisfy $(\mathsf{Sel}) \cup (\mathsf{Inv})$.

Thus, let the structure N be such that $A_N = \overline{A}$, $c_N(x) = \overline{c}(x)$ and $s_{i_N}(x) = \overline{s_i}(x)$ for all x, i. N is a total structure and a model of $(\mathsf{Sel}) \cup (\mathsf{Inv})$. A weak embedding from U_M into U_N is the identity function. Thus, the theory $\mathcal{T}$ satisfies $(\mathsf{Emb}^{\mathsf{c}})$. □

Example 3.42 (Binary trees as stably local theories)**.** An example of recursive data structures are binary trees, built up from a constructor *node* of type $A \times A \to A$ with selectors *left* and *right*, each of type $A \to A$:

$$\begin{array}{ll} (\mathsf{Sel}) & \left\{ \begin{array}{l} \mathit{left}(\mathit{node}(x_1, x_2) = x_1 \\ \mathit{right}(\mathit{node}(x_1, x_2) = x_2 \end{array} \right. \\ (\mathsf{Inv}) & \mathit{node}(\mathit{left}(x), \mathit{right}(x)) = x \end{array}$$

By Theorem 3.41, the theory of binary trees above is stably local.

3.8.2 Locality of Recursive Data Structures

We can prove an even stronger locality property if we flatten the axioms of $\mathcal{T}$ and add axioms for injectivity of c and the s_i.

Theorem 3.43. *Consider a theory $\mathcal{T}'$ with sort A, an n-ary constructor c of type $A^n \to A$ and n unary selectors s_i of type $A \to A$, satisfying*

$$\begin{array}{ll} (\mathsf{Sel}') & \left\{ \begin{array}{l} x = c(x_1, \ldots, x_n) \to s_1(x) = x_1 \\ \ldots \\ x = c(x_1, \ldots, x_n) \to s_n(x) = x_n \end{array} \right. \\ (\mathsf{Inv}') & \bigwedge_{i=1}^n s_i(x) = x_i \to c(x_1, \ldots, x_n) = x \\ (\mathsf{Inj}_1) & \left\{ \begin{array}{l} c(x_1, \ldots, x_n) = c(y_1, \ldots, y_n) \to x_1 = y_1 \\ \ldots \\ c(x_1, \ldots, x_n) = c(y_1, \ldots, y_n) \to x_n = y_n \end{array} \right. \\ (\mathsf{Inj}_2) & \bigwedge_{i=1}^n s_i(x) = s_i(y) \to x = y \end{array}$$

The theory $\mathcal{T}'$ satisfies $(\mathsf{Emb}^{\mathsf{c}}_{\mathsf{w}})$, and thus it is local.

Proof. We use essentially the same completion of c_M and the s_{i_M} as in the proof of Theorem 3.41, but have to consider some additional cases of undefined functions c_M and s_{i_M} (since weak partial models are less restrictive wrt. undefinedness than Evans partial models). Also, in weak partial models injectivity of c_M and the s_{i_M} does not necessarily follow from (Sel') and (Inv'), but they do follow from the explicit injectivity conditions (Inj_1) and (Inj_2).

Let M be a countable weak partial model of (Sel') $\cup$ (Inv') $\cup$ (Inj_1) $\cup$ (Inj_2), where c_M is defined for finitely many elements of A_M^n and each of s_{i_M} is defined for finitely many elements of A_M.

We define completions $\overline{c}$ of c_M and $\overline{s_i}$ of the s_{i_M}, considering the following cases:

i) $|A_M| = 1$,

ii) $|A_M| \geq 2$ and $n = 1$, and

iii) $|A_M| \geq 2$ and $n \geq 2$.

In case i), we can trivially complete any partial model to a total model by defining every possible function value to be equal to the single element.

In case ii), we complete M to a total model of the same size by first defining $\overline{c}(x) = y$ whenever either $c_M(x) = y$ or $s_{1_M}(y) = x$, and $\overline{s}(x) = y$ whenever either $s_{1_M}(x) = y$ or $c_M(y) = x$. (In contrast to Evans partial models, we cannot expect one to be defined if the other is.) If the resulting functions are not total, let $A_1 \subseteq A_M$ be the set of elements for which $\overline{c}$ is not defined, and $A_2 \subseteq A_M$ be the set of elements for which $\overline{s_1}$ is not defined. By definition of $\overline{c}$ and $\overline{s_1}$, we have $|A_1| = |A_2|$. Let $f : A_1 \to A_2$ be an arbitrary bijection, and define $\overline{c}(x) = f(x)$ for all $x \in A_1$, and $\overline{s_1}(x) = f^{-1}(x)$ for all $x \in A_2$. The resulting functions $\overline{c}$ and $\overline{s_1}$ are complete on A, and direct inverses, which means they must both be injective. Thus, they satisfy (Sel') $\cup$ (Inv') $\cup$ (Inj_1) $\cup$ (Inj_2).

In case iii), existence of more than one element and injectivity of an n-ary constructor implies that in any total model M' of $\mathcal{T}$, $A_{M'}$ must have infinitely many elements. Thus, if A_M is finite, let $\overline{A} \supseteq A_M$ be a countably infinite set of elements. In order to find completions of c_M and the s_{i_M}, we define a bijection $\overline{s} : \overline{A} \to \overline{A}^n$. Since $\overline{A}$ is countable, there exists a bijection $i : \mathbb{N} \to \overline{A}$. Let $<$ be the well-founded total order on $\overline{A}$ induced by this bijection and the usual order on $\mathbb{N}$. Let $\mathsf{Dom}(c_M) = \{\ a \in A_M^n \mid c_M(a) \text{ defined } \}$ and $\mathsf{Codom}(c_M) = \{\ a \in A_M \mid c_M(b) = a \text{ defined for some } b \in A_M^n\ \}$. We define $\overline{s}(i(m))$ inductively for all $m \in \mathbb{N}$:

Case $m = 1$: Assume $\overline{s}$ is not defined for any $a \in \overline{A}$. If $i(1) \in \mathsf{Codom}(c_M)$, then there is $a \in \mathsf{Dom}(c_M)$ such that $c_M(a) = i(1)$. Because of injectivity of c_M (on its domain of definition), this a is unique. Define $\overline{s}(i(1)) = a$. Otherwise, define $\overline{s}(i(1)) = (a_1, \ldots, a_n)$, where $a_i = s_{i_M}(i(1))$, if defined. For i with $s_{i_M}(i(1))$ undefined, choose a_i such that $(a_1, \ldots, a_n)$ is the smallest element (wrt. to the order $<$) of $\overline{A}^n$ with $(a_1, \ldots, a_n) \notin \mathsf{Dom}(c_M)$. (Again, we cannot assume that the s_{i_M} are defined iff c_M is defined for weak partial models.)

Case $m \to m + 1$: Assume $\overline{s}$ is defined for $\{\ i(1), \ldots, i(m)\ \}$ such that $\overline{s} : \{\ i(1), \ldots, i(m)\ \} \to \overline{A}^n$ is injective. Define $\overline{s}(i(m+1))$ as above, except that in case $i(m+1) \notin \mathsf{Codom}(c_M)$, we require that $(a_1, \ldots, a_n) \notin \mathsf{Dom}(c_M) \cup \{\ i(1), \ldots, i(m)\ \}$. Injectivity of $\overline{s} : \{\ i(1), \ldots, i(m+1)\ \} \to \overline{A}^n$ is immediate.

Since we always choose the smallest possible element from $\overline{A}^n$ wrt. $<$ (and $<$ is well-founded), the resulting function will also be surjective, and thus bijective.

Now, define $\overline{c} = \overline{s}^{-1}$, and $\overline{s_i}(x) = x_i$ whenever $\overline{s}(x) = (x_1, \ldots, x_n)$. By definition of $\overline{s}$, these functions are completions of c_M and the s_{i_M}, respectively. Since $\overline{c}$ and $\overline{s}$ are inverse, the completions satisfy $(\mathsf{Sel}') \cup (\mathsf{Inv}') \cup (\mathsf{Inj}_1) \cup (\mathsf{Inj}_2)$.

Thus, let the structure N be such that $A_N = \overline{A}$, $c_N(x) = \overline{c}(x)$ and $s_{i_N}(x) = \overline{s_i}(x)$ for all x, i. N is a total structure and a model of $(\mathsf{Sel}') \cup (\mathsf{Inv}') \cup (\mathsf{Inj}_1) \cup (\mathsf{Inj}_2)$. A weak embedding from U_M into U_N is the identity function. Thus, the theory satisfies $(\mathsf{Emb}^{\mathsf{c}})$. □

Example 3.44 (Binary trees as local theories)**.** We can also give a local axiomatization of binary trees:

$$
\begin{array}{ll}
(\mathsf{Sel}) & \left\{ \begin{array}{l} x = \mathit{node}(x_1, x_2) \rightarrow \mathit{left}(x) = x_1 \\ x = \mathit{node}(x_1, x_2) \rightarrow \mathit{right}(x) = x_2 \end{array} \right. \\
(\mathsf{Inv}) & \mathit{left}(x) = x_1 \wedge \mathit{right}(x) = x_2 \rightarrow \mathit{node}(x_1, x_2) = x \\
(\mathsf{Inj}_{\mathsf{node}}) & \left\{ \begin{array}{l} \mathit{node}(x_1, x_2) = \mathit{node}(y_1, y_2) \rightarrow x_1 = y_1 \\ \mathit{node}(x_1, x_2) = \mathit{node}(y_1, y_2) \rightarrow x_2 = y_2 \end{array} \right. \\
(\mathsf{Inj}_{\mathsf{sel}}) & \mathit{left}(x) = \mathit{left}(y) \wedge \mathit{right}(x) = \mathit{right}(y) \rightarrow x = y
\end{array}
$$

By Theorem 3.43, the theory of binary trees above is local.

Example 3.45 (Modeling Updates of Data Structures)**.** Using the piecewise boundedness axioms from Section 3.4, we can model updates of data structures. Consider a theory of binary trees, as defined in Example 3.42 or 3.44. Assume that *size* is a function measuring the size of tree terms. Then the following axioms define an update that modifies the tree such that for every parent node, the larger child tree will be on the left after the update:

$$
\begin{array}{l}
\mathit{size}(\mathit{left}(x)) \leq \mathit{size}(\mathit{right}(x)) \rightarrow \mathit{left}'(x) = \mathit{right}(x) \\
\mathit{size}(\mathit{left}(x)) \leq \mathit{size}(\mathit{right}(x)) \rightarrow \mathit{right}'(x) = \mathit{left}(x) \\
\neg(\mathit{size}(\mathit{left}(x)) \leq \mathit{size}(\mathit{right}(x))) \rightarrow \mathit{left}'(x) = \mathit{left}(x) \\
\neg(\mathit{size}(\mathit{left}(x)) \leq \mathit{size}(\mathit{right}(x))) \rightarrow \mathit{right}'(x) = \mathit{right}(x)
\end{array}
$$

Using such local update axioms, we can apply hierarchical reasoning to answer queries about the modified tree structure by instantiating the axioms above and then reducing the problem to the base theory.

3.9 Cardinality Functions for Boolean Algebra

In this section we consider the extension of the combined theory of Boolean algebras and Presburger arithmetic with a bridging function that allows us to reason about cardinalities of sets. In related work, this theory (including the extension function) has been called Boolean Algebra with Presburger Arithmetic (BAPA) [42, 44]. Constraints over such bridging functions that can model cardinalities and other numerical features of sets have first been shown to be decidable by Ohlbach and Köhler [50], and Ohlbach later generalized this work to other set description logics [49].

As in Section 3.8, we do not give new decidability results or a more efficient decision procedure here. Our contribution is the establishment of a locality result, which we plan to extend in future work.

3.9.1 Ψ-Locality of Cardinality Functions

We want to show that cardinality functions over sets can be described as Ψ-local theory extensions. More specifically, we consider as base theory $\mathcal{T}_0$ the disjoint combination of Presburger arithmetic $\mathcal{T}_{\mathbb{N}}$ with sort $\mathbb{N}$ and the theory of Boolean algebra $\mathcal{T}_B$ with sort B, constants $\varnothing$ (the empty set) and $\mathcal{U}$ (the universe, i.e. a set that contains all sets in this algebra) of sort B, as well as function symbols $\cap$ of type $B \times B \to B$ (set intersection), $\cup$ of type $B \times B \to B$ (set union), and ^{c} of type $B \to B$ (set complement). We want to extend this combination with an extension symbol $|\cdot|$ of type $B \to \mathbb{N}$, modelling cardinality of sets.

Define

$$\mathcal{K} = \left\{ \begin{array}{l} x_1 \cap x_2 = \varnothing \;\to\; |x_1 \cup x_2| = |x_1| + |x_2|, \\ |\varnothing| = 0, \\ |x| = 0 \;\to\; x = \varnothing \end{array} \right\}.$$

We want to find a function Ψ such that for any set of ground clauses G in the extended signature, we have

$$\mathcal{T}_0 \cup \mathcal{K} \cup G \models \square \;\Leftrightarrow\; \mathcal{T}_0 \cup \mathcal{K}^{\Psi}[G] \cup G \models \square.$$

To this end, consider a set of ground clauses G. Let $T = \mathsf{st}(G)$ and let $b_1, \ldots, b_n$ be the constants of sort B in T. Define the set of terms

$$B(T) = \{\; b_1^{c_1} \cap \ldots \cap b_n^{c_n} \;\mid\; c_i \in \{1, c\} \;\},$$

where $b_i^1 = b_i$. $B(T)$ consists of 2^n elements such that $\mathcal{T}_B \models e_1 \cap e_2 = \varnothing$ for all $e_1, e_2 \in B(T)$ with $e_1 \neq e_2$ and $\mathcal{T}_B \models (\bigcup_{e \in B(T)} e) = \mathcal{U}$. Furthermore, for every term t of sort B in T we have

$$\mathcal{T}_B \models t = \bigcup_{e \in B(t)} e,$$

with

$$B(t) = \{\; e \in B(T) \;\mid\; \mathcal{T}_B \models e \subseteq t \;\}.$$

This means that $B(T)$ is a decomposition of $\mathcal{U}$ into atomic parts wrt. the given set of terms T, corresponding to the partitioning of $\mathcal{U}$ into Venn regions wrt. the given constants $b_1, \ldots, b_n$ of sort B.

As these atomic parts are all mutually disjoint, the cardinality of every term t of sort B can be defined by the sum of the cardinalities of its atomic subsets, i.e. if $B(t) = \{\; e_1, \ldots, e_m \;\}$, then $|t| = |e_1| + \ldots + |e_m|$ should hold.

To achieve this, we will require that for every $t \in T$ with $B(t) = \{\; e_1, \ldots, e_m \;\}$, $\Psi(T)$ contains $|e_i|$ for $1 \leq i \leq m$, and a *path* of cardinality terms $\mathsf{Path}(t) = \{\; |e_1 \cup e_2|, |e_1 \cup e_2 \cup e_3|, \ldots, |e_1 \cup \ldots \cup e_m| \;\}$.

We claim that the following function satisfies our requirement from above:

$$\begin{array}{rl} \Psi(T) = & T \cup \mathsf{st}(\mathcal{K}) \\ & \cup \{\; |e| \;\mid\; e \in B(t) \text{ for some } |t| \in T \;\} \\ & \cup \{\; \mathsf{Path}(t) \;\mid\; |t| \in T \;\} \end{array}$$

Theorem 3.46. *Let $\mathcal{T}_0$, $\mathcal{K}$ and Ψ as defined above. Then the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is a Ψ-local extension.*

Proof: We show locality directly by proving that for an arbitrary set of ground clauses G, we have $\mathcal{T}_0 \cup \mathcal{K}^\Psi[G] \cup G \models \Box \Leftrightarrow \mathcal{T}_0 \cup \mathcal{K} \cup G \models \Box$.
$\Rightarrow$: obvious.
$\Leftarrow$: Assume that $\mathcal{T}_0 \cup \mathcal{K} \cup G \models \Box$. In order to obtain a contradiction, assume that there exists a model M of $\mathcal{T}_0 \cup \mathcal{K}^\Psi[G] \cup G$. Consider the partial structure P where B_P is the Boolean subalgebra of B_M generated by $\{e_M \mid e \in B(T)\}$, $\mathbb{N}_P = \mathbb{N}_M$, predicates and functions from $\mathcal{T}_0$ are the restrictions of those in M to B_P and $\mathbb{N}_P$, and $|b|_P$ is defined for $b \in B_P$ iff there is a term $|t| \in \Psi(\mathsf{st}(G))$ with $t_P = b$. Then P is a weak partial model of $\mathcal{K}^\Psi[G] \cup G$. In the following we prove that it can be completed to a total model of $\mathcal{T}_0 \cup \mathcal{K} \cup G$, contradicting our first assumption.

By construction of Ψ, for every term t such that $|t_P|_P$ is defined, $|e_P|_P$ is defined for all $e \in B(t)$. Furthermore, $|t'_P|_P$ is defined for all $|t'| \in \mathsf{Path}(t)$, which implies (together with the first axiom of $\mathcal{K}$ and the fact that elements of $B(t)$ are disjoint) that $|t_P|_P = \sum_{e \in B(t)} |e_P|_P$. Finally, $|\varnothing_P|_P = 0_P$.

Now we can recursively define a completion $\overline{|\cdot|}$ of $|\cdot|_P$, where

$$\overline{|b|} = \begin{cases} |b|_P, & \text{if defined} \\ 1_P, & \text{if } b = e_P \text{ for some } e \in B(T), |e_P|_P \text{ undefined} \\ \sum_{e \in B(t)} \overline{|e_P|}, & \text{if } b = t_P \text{ for some } t \notin B(T) \text{ and } |t_P|_P \text{ undefined} \end{cases}$$

Note that in the second case we have $b \neq \varnothing_P$, as $|\varnothing_P|_P$ must be defined. Furthermore, $\overline{|b|}$ is well-defined in the last case, even if there are different terms t, t' with $b = t_P = t'_P$: if $t_P = t'_P$ then $e_P \subseteq t_P$ if and only if $e_P \subseteq t'_P$, for all $e \in B(T)$, which implies $B(t) = B(t')$.

Thus, we have $\overline{|t_P|} = \sum_{e \in B(t)} \overline{|e_P|}$ for non-atomic terms t. Since every element $b \in B_P$ can be represented as a finite union of atomic elements, we can show that the first axiom of $\mathcal{K}$ is satisfied wrt. $\overline{|\cdot|}$ by a simple inductive argument:

i) If b, b' are such that $b = e_P, b' = e'_P$ for $e, e' \in (B(T) \cup \varnothing)$, then

$$\overline{|b \cup b'|} = \sum_{e \in B(b \cup b')} \overline{|e_P|} = \overline{|b|} + \overline{|b'|}.$$

ii) Else assume that $b \cap b' = \varnothing$ and as induction hypotheses

$$\overline{|b|} = \sum_{e \in B(b)} \overline{|e_P|} \text{ and } \overline{|b'|} = \sum_{e \in B(b')} \overline{|e_P|}.$$

Then

$$\begin{aligned} \overline{|b \cup b'|} &= \sum_{e \in B(b \cup b')} \overline{|e_P|} \\ &= \sum_{e \in B(b) \cup B(b')} \overline{|e_P|} \\ &= \sum_{e \in B(b)} \overline{|e_P|} + \sum_{e \in B(b')} \overline{|e_P|} \\ &= \overline{|b|} + \overline{|b'|}. \end{aligned}$$

It is easy to see that $\overline{|\cdot|}$ also satisfies the third axiom in $\mathcal{K}$ for all atomic elements, and by the recursive definition also for all $b \in B_P$. Furthermore, by definition of P we have $t_P = t_M$ for every term t in G. Thus, the total structure based on P, with the total function $\overline{|\cdot|}$ instead of $|\cdot|_P$, is a total model of $\mathcal{T}_0 \cup \mathcal{K} \cup G$, contradicting our assumption. $\Box$

Complexity Analysis. First, we analyze the number of terms in $\Psi(T)$ for a given T. The number of atomic subsets e_i for which $|e_i|$ is generated is in the worst case exponential in the number n of constants b_i. For every extension term $|t|$ in T, we add a path of extension terms, in the worst case containing 2^n terms. If l is the number of extension terms in T, then $l \cdot 2^n$ is an upper bound for the number of terms in all these paths.[5] Thus, overall we have

$$|\Psi(T)| \leq |T| + |\mathsf{st}(\mathcal{K})| + (l+1) \cdot 2^n.$$

As both l and n are bounded by $|T|$ (and $|\mathsf{st}(\mathcal{K})| = 1$), the number of extension terms in $\Psi(T)$ is in $2^{O(|T|)}$.

Since there are at most two variables in every axiom of $\mathcal{K}$, the number of axiom instances that have to be generated for a given T is quadratic in the number of extension terms in $\Psi(T)$.

[5]Note that when only counting the number of terms, we neglect the cost of $l \cdot 2^n$ inclusion tests $e \subseteq t$. Since we can assume each test to be (non-deterministic) polynomial in the size of t, this will not change the overall complexity.

Chapter 4

Incremental Local Reasoning

In Section 2.3 we have introduced *local reasoning*, which gives us a decision procedure for ground satisfiability problems in certain local extensions of theories. Theoretically, this means that the procedure is guaranteed to terminate on arbitrary satisfiable and unsatisfiable inputs, given sufficient resources. In practice, generating all instances at once can be inefficient, especially for large verification problems where unsatisfiability often only depends on a small part of the formula. Therefore, we are looking for methods to treat local theory extensions more efficiently.

In this chapter we introduce methods to generate the needed instances incrementally. This is done 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 [21, 22, 23]: we keep a candidate model that satisfies the ground part of the current problem, and only add axiom instances that evolve this candidate model, either strengthening or refuting it. In recent years, instantiation-based methods have shown to be very efficient for solving effectively finite instantiation problems in pure FOL: from 2003 to 2009, the winner of the annual automated theorem proving system competition [63] in the EPR (effectively propositional) division has been an implementation of an instantiation-based approach. We want to transfer ideas from instantiation-based reasoning into a framework with background theories and effectively finite instantiation problems: hierarchical reasoning in local theory extensions.

In Section 4.1, we briefly review the original instance generation method by Ganzinger and Korovin. Section 4.2 introduces *local instance generation*, an adaption of this method to the case of local theory extensions. For chains of local theory extensions, we need a more sophisticated approach, which we will introduce in Section 4.3. We give experimental results in Section 4.4, showing that the incremental approaches are superior to the standard approach in many cases. The main results of this chapter have been published in 2008 [37] and 2009 [38].

4.1 Instance Generation and its Refinements

Resolution-based Instance Generation (*IG*) has been introduced by Ganzinger and Korovin in 2003 [21] as a theorem proving method for first-order logic without equality, and has since been refined to equational reasoning [22] and general theory reasoning [23]. The calculus *IG* 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]

We will need the following additional definitions:

Definition 4.1 (Variable Renaming)**.** A *variable renaming* is an injective substitution mapping variables to variables.

Definition 4.2 (Variant)**.** Two clauses C_1 and C_2 are *variants* of each other if there is a variable renaming σ such that $C_1\sigma = C_2$.

Definition 4.3 (Generalization)**.** If C_2 is an instance of C_1, then C_1 is a *generalization* of C_2.

Definition 4.4 (Most specific generalization)**.** Let F be a set of clauses. C_1 is a *most specific generalization of C_2 with respect to F* if C_1 is a generalization of C_2 and for every $C_3 \in F$ such that C_3 is both an instance of C_1 and a generalization of C_2, C_3 is a variant of C_1.

Definition 4.5 ((Partial) Herbrand interpretation)**.** A *partial Herbrand interpretation* for a given theory $\mathcal{T}$ with signature Π is a set of ground Π-literals $\mathcal{I}$ such that $\mathcal{I} \cup \mathcal{T}$ is consistent. It is a *total Herbrand interpretation* if for every ground Π-atom A, $\mathcal{I}$ contains either A or $\neg A$.

Definition 4.6 ((Partial) Herbrand model)**.** A (partial) Herbrand interpretation $\mathcal{I}$ (for a given theory $\mathcal{T}$) is a *(partial) Herbrand model* of a set of clauses F (notation: $\mathcal{I} \models F$) iff for every ground instance $C\sigma$ of a clause $C \in F$ there exists a literal $L \in C\sigma$ such that $L \in \mathcal{I}$.

From Herbrand's Theorem it follows that, for a universal theory $\mathcal{T}$ and a set of clauses F, $\mathcal{T} \cup F$ is satisfiable iff there exists a (partial) Herbrand model for $\mathcal{T}$ and F. We assume in the rest of Section 4.1 that our background theory $\mathcal{T}$ is universal.

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 $\mathcal{T} \cup F\bot \models \Box$, the procedure terminates and states unsatisfiability of the input. Otherwise, generate a partial Herbrand model $\mathcal{M}$ for $\mathcal{T}$ and $F\bot$.

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

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

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

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

Note that several variants of the same clause $C \in F$ can be used as input, and thus several instances of the same clause can be generated in the same inference. We consider a clause C *redundant with respect to* F if there is a clause $D \in F$ that is a variant of C.[2] If all inferences based on sel only produce instances that 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 satisfiability of the input. Otherwise, after an inference step that adds at least one non-redundant clause to F, we go back to **(1)**.

Example 4.7 (taken from [22])**.** Let $\mathcal{T}$ be the empty theory, 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 clauses in F 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. For a given theory, a procedure that finds such a substitution if one exists is called *answer-complete*. E.g., paramodulation is answer-complete for the empty theory.[3] In general, answer computation is undecidable.

Correctness of the IG Calculus

Definition 4.8 (*IG*-derivation)**.** An *IG-derivation* is a sequence of sets of clauses $F_0 \vdash F_1 \vdash \ldots \vdash F_n$ such that for $0 \leq i \leq n-1$, $\mathcal{T} \cup F_i\bot$ is satisfiable and F_{i+1} is the result of an *IG*-inference on F_i.

Definition 4.9 (Persisting inference)**.** An *IG*-inference is called *persisting* if in an infinite *IG*-derivation the conditions for its applicability are satisfied infinitely often.

The following results have been proved by Ganzinger and Korovin [21, 22, 23]:

Theorem 4.10 (Soundness of IG)**.** *Let $F_0 \vdash F_1 \vdash \ldots \vdash F_n$ be an IG-derivation. If $\mathcal{T} \cup F_n\bot \models \Box$, then $\mathcal{T} \cup F_0 \models \Box$.*

[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.

[3] In first-order logic without equality, unification is answer-complete for the empty theory.

Theorem 4.11 (Completeness of IG)**.** *Let $F_0 \vdash F_1 \vdash \ldots \vdash F_n$ be an IG-derivation. If the background theory $\mathcal{T}$ has an answer-complete procedure for finding unsatisfiable instances, and $\mathcal{T} \cup F_n$ is satisfiable and saturated under IG, then $\mathcal{T} \cup F_0$ is satisfiable.*

Theorem 4.12 (Termination of IG)**.** *For a set of clauses F_0 such that $\mathcal{T} \cup F_0$ is unsatisfiable, IG will always terminate after a finite derivation $F_0 \vdash F_1 \vdash \ldots \vdash F_n$ if every persisting inference is eventually taken, and if the background theory $\mathcal{T}$ is universal and has an answer-complete procedure for finding unsatisfiable instances.*

4.2 Combining Instance Generation and Local Reasoning

In this Section, we introduce *local instance generation*, an adaption of the instance generation approach to the framework of local reasoning. We present the modified instantiation procedure in Section 4.2.1 and give some examples how local reasoning can benefit from this approach in Section 4.2.2. Finally, we show that the method is sound, complete and terminating in Section 4.2.3.

4.2.1 Local Instance Generation (LIG)

We present a refinement of *IG* that uses knowledge about locality in order to obtain a decision procedure. Assume that $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is a local theory extension, and there is a decision procedure for the universal fragment of $\mathcal{T}_0$ plus uninterpreted function symbols. Then, the *LIG*-procedure stores the extension axioms $\mathcal{K}$ and ground clauses G separately, and only the ground part determines the partial Herbrand model — selection on non-ground clauses is fixed by a heuristic. For reasoning on ground clauses we use an SMT solver that should allow incremental addition of constraints and must be able to return a model for a set of satisfiable clauses. Effectively, we obtain a procedure that solves the satisfiability problem of a ground formula G modulo an extended theory $\mathcal{T}_0 \cup \mathcal{K}$.

We present the ideas for standard locality[4], i.e. local extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ that satisfy property (Loc). Furthermore, we assume $\mathcal{K}$ is Σ_1-linear and that every variable in $\mathcal{K}$ has at least one appearance in an extension term in every clause, i.e. $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}$ is universally reducing.

In contrast to *IG*, we do not require $\mathcal{T}_0$ to be universal. We will show in Section 4.2.3 that this restriction is not needed to prove correctness of *LIG*.

Separation of non-ground clauses and ground clauses. In *LIG*, we keep working sets of non-ground clauses $\mathcal{K}$ and ground clauses G. For a set of clauses F, let $\mathsf{sel}(F) = \{\ \mathsf{sel}(C) \mid C \in F\ \}$. 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), and remains fixed throughout the saturation process. Selection on ground clauses in G will be made implicit by considering a partial Herbrand model $\mathcal{M}$ of $\mathcal{T}_0$ and G. Then, every literal in $\mathcal{M}$ can be considered to be in $\mathsf{sel}(G)$.

[4]Extensions to other notions of locality are possible, but will be left for future work.

Proof Procedure

Given a set of non-ground clauses $\mathcal{K}$ and a set of ground clauses G, *LIG* checks satisfiability of G modulo an extended theory $\mathcal{T}_0 \cup \mathcal{K}$ by interleaving satisfiability checks (modulo $\mathcal{T}_0$) with instantiation of clauses in $\mathcal{K}$. The following two steps are repeated until termination:

(1) Ground satisfiability check. We give G to an SMT solver for $\mathcal{T}_0$ plus uninterpreted functions. If $\mathcal{T}_0 \cup G \models \Box$, then the procedure terminates and states unsatisfiability of the input. Otherwise, the solver returns a model M such that $M \models \mathcal{T}_0 \cup G$. From M, compute a partial Herbrand model $\mathcal{M}$ of $\mathcal{T}_0$ and G.

(2) Instance generation. Instances are generated according to the following inference rule. We depict $\mathcal{M}$ as an additional premise of the inference:

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

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) $\mathcal{T}_0 \cup L_1\sigma \cup \ldots \cup L_n\sigma \cup \mathcal{M} \models \Box$

As in *IG*, several variants of a clause $C \in \mathcal{K}$ can be instantiated in one inference. Ground clauses $(L_i \vee D_i)\sigma$ are added to G, non-ground clauses (if any) are added to $\mathcal{K}$. 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 non-ground clauses as described above and return to **(1)**. Note that we do not need $\mathcal{M}$ anymore after the inference; a new partial Herbrand model will be computed in **(1)** if necessary.

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

Lemma 4.13. *Let* $\{ (L_1 \vee D_1), \ldots, (L_n \vee D_n) \}$ *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 every* $\mathcal{L}_i\sigma$ *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. Consider an arbitrary $\mathcal{L}_i$ as defined above. As $\mathcal{L}_i$ contains all variables of $(L_i \vee D_i)$ 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 can 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 $\mathcal{L} = \{\mathsf{sel}(C)\}$.

Sort information. Note that we do not have to add explicit sort information to the *LIG* inference rule, as the restriction to clauses in $\mathcal{K}[\mathcal{M}]$ 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 can eliminate premises (and conclusions) of an inference which are not needed to satisfy condition (iv) of the *LIG* rule. If $\mathcal{T}_0 \cup \{L_i\sigma\}_{1\leq i\leq n} \cup \mathcal{M} \models \square$, then a decision procedure for $\mathcal{T}_0$ 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 $\mathcal{T}_0 \cup \mathcal{L} \cup \mathcal{M} \models \square$. If we ignore premises and conclusions that do not contain a literal $L_i \in \mathcal{L}$, we still have a valid *LIG* inference.

In the following we will use *LIG* to denote the calculus that does not use unsatisfiable cores, and $\mathit{LIG}_{\mathsf{uc}}$ to denote the calculus that does.

4.2.2 Examples: Behavior of *LIG* and $\mathit{LIG}_{\mathsf{uc}}$

In the following examples, we compare the different approaches to local reasoning. We will call the standard method from Section 2.3 *eager instantiation*, as opposed to the incremental methods *LIG* (without unsatisfiable cores), and $\mathit{LIG}_{\mathsf{uc}}$ (with unsatisfiable cores). For all examples, consider the same background theory and extension as in Example 2.42 (on page 13), i.e. $\mathcal{T}_0$ has a partial ordering and $\mathcal{K} = \{\ x \leq y \rightarrow f(x) \leq f(y)\ \}$.

Example 4.14. Let $G = \{\ a \leq b,\ \neg(f(a) \leq f(b))\ \}$. Then, as seen in Example 2.42, 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 the eager instantiation.

LIG_{uc} searches for an unsatisfiable core in $\mathcal{L}$ and can 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 G only consists of unit clauses, 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 that 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 4.15. Let G consist of the ground clauses

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

For any n, this set of clauses is unsatisfiable, because the set $\{ C_1, D_1, E, F \}$ is already unsatisfiable in $\mathcal{T}_0 \cup \mathcal{K}$. 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 $\mathcal{M}$ can 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* will 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 that makes G unsatisfiable.

Thus, *LIG* proves $\mathcal{T}_0 \cup \mathcal{K} \cup G \models \square$ 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 4.16. Consider a modified version of Example 4.15: 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 $\mathcal{K}[G] \cup G$.

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

In *LIG*, the SMT solver generates a model M for G, from which we compute a partial Herbrand model $\mathcal{M}$. 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 $(2n+1)^2$ 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 that does not include the second literal of E, or until the set of instances is saturated with respect to the model, i.e. all 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* can 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 instance.

4.2.3 Correctness of *LIG* and LIG_{uc}

In the following, we show that *LIG* and LIG_{uc} are sound, complete and terminating.

Definition 4.17 (*LIG*-derivation)**.** 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 $\mathcal{T}_0$, and for $0 \leq i \leq n-1$, G^i is $\mathcal{T}_0$-satisfiable and $(\mathcal{K}^{i+1}, G^{i+1})$ is the result of an *LIG*-inference on $(\mathcal{K}^i, G^i)$.

An LIG_{uc}-derivation is defined in the obvious way.

Theorem 4.18 (Soundness)**.** *Let* $(\mathcal{K}^0, G^0) \vdash (\mathcal{K}^1, G^1) \vdash \ldots \vdash (\mathcal{K}^n, G^n)$ *be an LIG- or* LIG_{uc}*-derivation. If* $\mathcal{T}_0 \cup G^n \models \square$*, then* $\mathcal{T}_0 \cup \mathcal{K}^0 \cup G^0 \models \square$*.*

Proof: All elements of G^n are instances of clauses in $\mathcal{K}^0 \cup G^0$. Thus, their unsatisfiability (modulo $\mathcal{T}_0$) implies unsatisfiability of $\mathcal{K}^0 \cup G^0$. $\square$

Theorem 4.19 (Completeness of *LIG*)**.** *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* $\mathcal{T}_0 \cup G^n$ *is 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* $\mathcal{T}_0 \cup \mathcal{K}^0 \cup G^0$ *is satisfiable.*

Proof: Let $\mathcal{K}^0, G^0, \mathcal{K}^n, G^n$, sel and $\mathcal{M}$ as defined above. $\mathcal{M}$ is a partial Herbrand model of $\mathcal{T}_0$ and G^n, where $G^n \subseteq \mathcal{K}^0[G^0] \cup G^0$. We extend $\mathcal{M}$ to a partial 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 wrt. $\mathcal{T}_0$: 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 $\mathcal{M}$. As $\mathcal{M}'$ is not contradictory wrt. $\mathcal{T}_0$ and contains one literal out of each clause in $\mathcal{K}^0[G^0] \cup G^0$, we can conclude that $\mathcal{T}_0 \cup \mathcal{K}^0[G^0] \cup G^0$ is satisfiable. By locality, $\mathcal{T}_0 \cup \mathcal{K}^0 \cup G^0$ is satisfiable iff $\mathcal{T}_0 \cup \mathcal{K}^0[G^0] \cup G^0$ is satisfiable. □

Theorem 4.20 (Completeness of LIG_{uc})**.** *Let* $(\mathcal{K}^0, G^0) \vdash (\mathcal{K}^1, G^1) \vdash \ldots \vdash (\mathcal{K}^n, G^n)$ *be an* LIG_{uc}*-derivation. If* $\mathcal{T}_0 \cup G^n$ *is satisfiable and* $(\mathcal{K}^n, G^n)$ *is saturated under* LIG_{uc} *with respect to a given selection function* sel *and a partial Herbrand model* $\mathcal{M}$*, then* $\mathcal{T}_0 \cup \mathcal{K}^0 \cup G^0$ *is satisfiable.*

Proof: The proof is the same as for *LIG*, except that LIG_{uc} can ignore some premises if the selected literals do not appear in an unsatisfiable core:

Let $\mathcal{K}^0, G^0, \mathcal{K}^n, G^n$, sel and $\mathcal{M}$ as defined above. $\mathcal{M}$ is a partial Herbrand model of $\mathcal{T}_0$ and G^n, where $G^n \subseteq \mathcal{K}^0[G^0] \cup G^0$. We extend $\mathcal{M}$ to a partial 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 wrt. $\mathcal{T}_0$: if this was the case, there would be an unsatisfiable core of $\mathcal{M}'$, containing at least one literal $\mathsf{sel}(D_i)\sigma_i$ (since $\mathcal{M}$ itself is not contradictory). Thus, we would have an LIG_{uc} inference with a subset of 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_{uc} wrt. sel and $\mathcal{M}$. As $\mathcal{M}'$ is not contradictory wrt. $\mathcal{T}_0$ and contains one literal out of each clause in $\mathcal{K}^0[G^0] \cup G^0$, we can conclude that $\mathcal{T}_0 \cup \mathcal{K}^0[G^0] \cup G^0$ is satisfiable. By locality, $\mathcal{T}_0 \cup \mathcal{K}^0 \cup G^0$ is satisfiable iff $\mathcal{T}_0 \cup \mathcal{K}^0[G^0] \cup G^0$ is satisfiable. □

Theorem 4.21 (Termination)**.** *For any input* $(\mathcal{K}^0, G^0)$ *(where* $\mathcal{K}^0$ *and* G^0 *are as defined above), LIG and* LIG_{uc} *terminate after a finite number of inferences.*

Proof: Both *LIG*- and LIG_{uc}-inferences produce only clauses that 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 inferences. □

Note that the difference to Theorems 4.11 and 4.12 is not only that we terminate for any input, but we also dropped the additional requirements to the background theory. We do not need the background theory to be universal or to have answer-complete methods for finding unsatisfiable literal instances, because locality of the theory extension gives us a finite search space of instances, and thus more powerful completeness and termination arguments. In this way, we have not only specialized the original *IG*-procedure, but can now effectively apply it to background theories that were out of the scope of the original method.

4.3 Incremental Instance Generation for Chains of Extensions

In Section 2.6.1 we showed how local reasoning can be generalized to chains of extensions. In Section 4.2.1 we introduced *LIG*, an incremental method for solving satisfiability problems modulo a local extension of a background theory, and its refinement LIG_{uc} which uses unsatisfiable cores to further minimize the number of generated instances. For a chain of extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \ldots \subseteq \mathcal{T}_m$, *LIG* (and LIG_{uc}) cannot be used as such: if we want to do the reduction from $\mathcal{T}_m$ to $\mathcal{T}_{m-1}$ incrementally, the inference rule requires that satisfiability problems in $\mathcal{T}_{m-1}$ can already be solved. In order to do this, we would need to use the standard reduction for chains of extensions from $\mathcal{T}_{i-1}$ down to $\mathcal{T}_0$, or call *LIG* recursively. We want to do neither of both.

Instead, we will introduce a refinement LIG^* of the *LIG* procedure that can handle chains of extensions directly. This requires a more sophisticated inference rule and infrastructure, but we are convinced that reasoning in chains of extensions can benefit very much from incremental generation of instances, since the set of ground clauses G_j grows polynomially with every reduction.

As for *LIG*, we will first introduce the method (Section 4.3.1), then give some examples (Section 4.3.2) and finally prove its correctness (Section 4.3.3).

4.3.1 Chain-Local Instance Generation (LIG^*)

We present the procedure LIG^*, which solves the satisfiability problem of a set of ground clauses G modulo a repeatedly extended theory $\mathcal{T}_0 \cup \mathcal{K}_1 \cup \ldots \cup \mathcal{K}_n$. As in Section 4.2.1, we only support standard locality, i.e. chains of local extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1 \subseteq \ldots \subseteq \mathcal{T}_0 \cup \mathcal{K}_1 \cup \ldots \cup \mathcal{K}_n$ that satisfy property (Loc) for every extension. Furthermore, we assume that all $\mathcal{K}_i$ are Σ_i-linear and that variables in the $\mathcal{K}_i$ have at least one appearance in an extension term in every clause, enforcing that $\mathcal{K}_i[G]$ is a set of ground clauses for every i and every set of ground clauses G.

LIG^* differs from *LIG* mainly in two points: First, for m successive extensions, non-ground clauses need to be kept in m different sets $\mathcal{K}_1, \ldots, \mathcal{K}_m$. Second, the search space for possible instances is restricted to the following sets of instances (as defined in Section 2.6.1):

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

Proof Procedure

For a given input $(\mathcal{K}_1, \ldots, \mathcal{K}_m, G)$, *LIG** keeps non-ground clauses 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 for $\mathcal{T}_0$ plus uninterpreted functions. If $\mathcal{T}_0 \cup G \models \Box$, then the procedure terminates and returns unsatisfiable. Otherwise, the solver returns a model M such that $M \models \mathcal{T}_0 \cup G$. As before, generate from M a partial Herbrand model $\mathcal{M}$ of $\mathcal{T}_0$ and G.

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

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

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 $\mathcal{K}_j[\bigcup_{j<k\leq m+1} \mathcal{I}_k]$, and

(iv) $\mathcal{T}_0 \cup L_1\sigma \cup \ldots \cup L_n\sigma \cup \mathcal{M} \models \Box$

Again, several variants of a clause can 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 non-ground clauses as described above and return to **(1)**.

Search for substitutions. Similarly to the case of *LIG*, the search for substitutions can be restricted to a finite set. We provide a lifting of Lemma 4.13:

Lemma 4.22. *Let* $\{ (L_1 \vee D_1), \ldots, (L_n \vee D_n) \}$ *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 *that 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 for any* $(L_i \vee D_i) \in \mathcal{K}_j$*, all ground extension terms in* $\mathcal{L}_i\sigma$ *are in* $\mathsf{st}(\bigcup_{j<k\leq m+1} \mathcal{I}_k)$*.*

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

Now suppose (2) holds. Because every $\mathcal{K}_j$ 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 $\mathsf{st}(\bigcup_{j<k\leq m+1}\mathcal{I}_k)$, so must be all ground extension terms in $(L_i \vee D_i)\sigma$. By definition of $\mathcal{K}_j[\bigcup_{j<k\leq m+1}\mathcal{I}_k]$, every $(L_i \vee D_i)\sigma$ is a generalization of a clause in $\mathcal{K}_j[\bigcup_{j<k\leq m+1}\mathcal{I}_k]$, satisfying (iii). □

Thus, as before we can watch for every non-ground clause C a set of literals $\mathcal{L} \subseteq C$ to ensure conditions (ii) and (iii) of the inference rule, and only need to consider substitutions σ that map variables of $\mathsf{sel}(C)$ to ground terms such that resulting ground extension terms are in a given set of terms. In this case we need more effort to find this set of terms, however: for clauses $C \in \mathcal{K}_j$, we need to compute the set $\mathsf{st}(\bigcup_{j<k\leq m+1}\mathcal{I}_k)$.

Special cases. We have the same special cases as before. For $C \in \mathcal{K}_j$:

- 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 in extension terms (of extension $\mathcal{K}_j$) in $\mathsf{sel}(C)$, then it is sufficient to consider $\mathcal{L} = \{\mathsf{sel}(C)\}$.

Unsatisfiable cores. Unsatisfiable cores can be computed and used as before, eliminating unnecessary premises and conclusions. Similar to the distinction between LIG and LIG_{uc}, we will use LIG^* to denote the calculus for chains of extensions that does not use unsatisfiable cores, and LIG^*_{uc} to denote the calculus that does.

4.3.2 Examples: Behavior of LIG^* and LIG^*_{uc}

Based on Theorem 3.39, we can extend the examples given for LIG to chains of extensions: consider again a base theory $\mathcal{T}_0$ with a partial ordering, and let

$$\begin{aligned}\mathcal{K}_1 = &\ \{\ x \leq y \rightarrow f_1(x) \leq f_1(y)\ \},\\ \mathcal{K}_2 = &\ \{\ f_1(x) \leq f_1(y) \rightarrow f_2(x) \leq f_2(y)\ \},\\ \mathcal{K}_3 = &\ \{\ f_2(x) \leq f_2(y) \rightarrow f_3(x) \leq f_3(y)\ \}.\end{aligned}$$

If we define $\mathcal{T}_1 = \mathcal{T}_0 \cup \mathcal{K}_1$, $\mathcal{T}_2 = \mathcal{T}_1 \cup \mathcal{K}_2$ and $\mathcal{T}_3 = \mathcal{T}_2 \cup \mathcal{K}_3$, then by locality of $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1$ and by Theorem 3.39, $\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \mathcal{T}_2 \subseteq \mathcal{T}_3$ is a chain of local theory extensions.

In all three axioms, $\mathsf{sel}(\mathcal{K}_i)$ is the last literal, since it contains both variables below the extension symbol of the respective axiom.

Example 4.23. If we use $G = \{\ a \leq b,\ \neg(f_3(a) \leq f_3(b))\ \}$, we have essentially the same result as in Example 4.14: due to the missing boolean structure of G, for any model $\mathcal{M}$ of G we have $\mathcal{K}_3[\mathcal{M}] = \mathcal{K}_3[G]$, and thus $\mathcal{K}_2[\mathcal{K}_3[\mathcal{M}] \cup G] =$

$\mathcal{K}_2[\mathcal{K}_3[G] \cup G]$ and $\mathcal{K}_1[\mathcal{K}_2[\mathcal{K}_3[\mathcal{M}] \cup G] \cup G] = \mathcal{K}_1[\mathcal{K}_2[\mathcal{K}_3[G] \cup G] \cup G]$. Thus, an LIG^*-procedure that takes maximal inferences will produce exactly the same instances as the eager instantiation:

$$\begin{aligned}
\mathcal{K}_3[G] = \{ \quad & f_2(a) \leq f_2(a) \rightarrow f_3(a) \leq f_3(a) \\
& f_2(a) \leq f_2(b) \rightarrow f_3(a) \leq f_3(b) \\
& f_2(b) \leq f_2(a) \rightarrow f_3(b) \leq f_3(a) \\
& f_2(b) \leq f_2(b) \rightarrow f_3(b) \leq f_3(b) \ \} \\
\mathcal{K}_2[G_2] = \{ \quad & f_1(a) \leq f_1(a) \rightarrow f_2(a) \leq f_2(a) \\
& f_1(a) \leq f_1(b) \rightarrow f_2(a) \leq f_2(b) \\
& f_1(b) \leq f_1(a) \rightarrow f_2(b) \leq f_2(a) \\
& f_1(b) \leq f_1(b) \rightarrow f_2(b) \leq f_2(b) \ \} \\
\mathcal{K}_1[G_1] = \{ \quad & a \leq a \rightarrow f_1(a) \leq f_1(a) \\
& a \leq b \rightarrow f_1(a) \leq f_1(b) \\
& b \leq a \rightarrow f_1(b) \leq f_1(a) \\
& b \leq b \rightarrow f_1(b) \leq f_1(b) \ \}
\end{aligned}$$

LIG^*_{uc} will in a first step find out that the instance $f_3(a) \leq f_3(b)$ of the selected literal in $\mathcal{K}_3$ is unsatisfiable in $\mathcal{M}$, producing the clause instance $f_2(a) \leq f_2(b) \rightarrow f_3(a) \leq f_3(b)$. A Herbrand model of this instance and G must contain $\neg(f_2(a) \leq f_2(b))$, so in the next step the procedure will find out that the instance $f_2(a) \leq f_2(b)$ of the selected literal in $\mathcal{K}_2$ is unsatisfiable in $\mathcal{M}'$, producing the clause instance $f_1(a) \leq f_1(b) \rightarrow f_2(a) \leq f_2(b)$. A Herbrand model of G and the two instances produced thus far must contain $\neg(f_1(a) \leq f_1(b))$, which then leads to generation of the instance $a \leq b \rightarrow f_1(a) \leq f_1(b)$ of $\mathcal{K}_1$. Together, G and these three instances are unsatisfiable.

Thus, for this example both the eager instantiation and LIG^* with maximal inferences will produce 12 instances, while LIG^*_{uc} would need three steps to produce exactly those 3 instances that are necessary for proving unsatisfiability of the set of ground clauses.

We can also extend Example 4.15 to the case of LIG^*. In this case, we assume that there is not only some redundant information on f_3, but also on f_2 and f_1.

Example 4.24. Let G consist of the ground clauses

$$\begin{aligned}
&(C_i) && a_i \leq b_i \ \vee \ c_i \leq d_i, && \text{for } 1 \leq i \leq n \\
&(D_i) && f_3(a_i) \leq f_3(b_i) \ \vee \ f_3(c_i) \leq f_3(d_i), && \text{for } 1 \leq i \leq n \\
&(E) && \neg(a_1 \leq b_1) \ \vee \ \neg(f_3(a_1) \leq f_3(b_1)) \\
&(F) && \neg(c_1 \leq d_1) \ \vee \ \neg(f_3(c_1) \leq f_3(d_1)) \\
&(G) && f_2(e1) \leq f_2(e2) \ \vee \ f_2(e3) \leq f_2(e4)) \\
&(H) && f_1(e5) \leq f_1(e6) \ \vee \ f_1(e7) \leq f_1(e8))
\end{aligned}$$

That is, compared to Example 4.15 we replace f by f_3 and add clauses G and H with redundant information on f_2 and f_1, respectively. Note that we will not add clauses containing f_2 or f_1 with increasing size of our parameter, but still the existing ground terms will lead to an increasing blow-up.

As before, this set of clauses is unsatisfiable for every n, because the set $\{ C_1, D_1, E, F \}$ is already unsatisfiable in $\mathcal{T}_3$. The instances of the axioms needed to prove this are

$$
\begin{aligned}
&f_2(a_1) \leq f_2(b_1) \rightarrow f_3(a_1) \leq f_3(b_1),\\
&f_2(c_1) \leq f_2(d_1) \rightarrow f_3(c_1) \leq f_3(d_1),\\
&f_1(a_1) \leq f_1(b_1) \rightarrow f_2(a_1) \leq f_2(b_1),\\
&f_1(c_1) \leq f_1(d_1) \rightarrow f_2(c_1) \leq f_2(d_1),\\
&a_1 \leq b_1 \rightarrow f_1(a_1) \leq f_1(b_1),\\
&c_1 \leq d_1 \rightarrow f_1(c_1) \leq f_1(d_1).
\end{aligned}
$$

Let us compare how the different approaches to local reasoning treat this problem:

The eager approach generates $\mathcal{K}_3[G]$, which consists of all instances of $\mathcal{K}_3$ with all combinations of substituting x and y with the constants a_i, b_i, c_i, d_i, i.e. $\mid \mathcal{K}_3[G] \mid = (4n)^2$. With the terms in $\mathcal{K}_3[G] \cup G$, we have to instantiate x and y in $\mathcal{K}_2$ by the a_i, b_i, c_i, d_i, as well as e_1, e_2, e_3, e_4. So here we have $\mid \mathcal{K}_2[G_2] \mid = (4n+4)^2$. Finally, for $\mathcal{K}_1$ we need to instantiate x and y with all combinations of the a_i, b_i, c_i, d_i and $e_i, 1 \leq i \leq 8$. Thus, $\mid \mathcal{K}_1[G_1] \mid = (4n+8)^2$.

In LIG^*, $\mathcal{M}$ can contain an arbitrary literal out of each ground 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. Furthermore, $\mathcal{M}$ will contain one literal for each of G and H.

For the extension terms with respect to $\mathcal{K}_3$, the situation is as in Example 4.15, i.e. we have $2n+2$ f_3-terms in $\mathsf{st}(\mathcal{M})$. The set $\mathsf{sel}(\mathcal{K}_3)[\mathcal{M}] \cup \mathcal{M}$ then contains f_2-terms with the same instantiation as the f_3-terms in $\mathcal{M}$, plus the additional f_2-terms from one literal in G, i.e. altogether we have $2n+4$ f_2-terms. Finally, $\mathcal{I}_2 = \mathcal{I}_3 \cup \mathcal{M}$ contains f_1-terms with the same instantiation as the f_2-terms in $\mathcal{I}_3 \cup \mathcal{M}$, plus the f_1-terms from one literal in H, i.e. $2n+6$ f_1-terms altogether. Thus, a maximal LIG^*-inference will generate $(2n+2)^2$ instances of $\mathcal{K}_3$, $(2n+4)^2$ instances of $\mathcal{K}_2$ and $(2n+6)^2$ instances of $\mathcal{K}_3$. All of the instances needed to prove unsatisfiability are necessarily generated in this step, and thus the procedure terminates after checking satisfiability of the resulting instances together with G.

LIG^*_{uc} will search for an unsatisfiable core of the corresponding $2n+2$ instances of $\mathsf{sel}(\mathcal{K}_3)$, $2n+4$ instances of $\mathsf{sel}(\mathcal{K}_2)$, and $2n+6$ instances of $\mathsf{sel}(\mathcal{K}_1)$. The Herbrand model $\mathcal{M}$ contains either $\neg(f_3(a_1) \leq f_3(b_1))$ or $\neg(f_3(c_1) \leq f_3(d_1))$, and the corresponding positive literal is an unsatisfiable core. After generating the according instance of $\mathcal{K}_3$, a Herbrand model of the resulting set must contain at least one of the three following literals: $\neg(f_2(a_1) \leq f_2(b_1))$, $\neg(f_2(a_1) \leq f_2(b_1))$ or the literal from above which was not in the previous model. Thus, we will again find a single instance of either $\mathsf{sel}(\mathcal{K}_3)$ or $\mathsf{sel}(\mathcal{K}_2)$ which is unsatisfiable in the new model, and generate the according axiom instance. It is not difficult to see that this process will continue until after 6 steps, all 6 of the instances necessary for proving unsatisfiability of G in $\mathcal{T}_3$ have been produced.

Thus, LIG^* with maximal inferences proves $\mathcal{T}_3 \cup G \models \square$ by generating $(2n+2)^2 + (2n+4)^2 + (2n+6)^2$ instances in one step, and LIG^*_{uc} needs to

generate only 6 instances in 2 steps. Recall that in contrast to this, eager instantiation generates $(4n+8)^2$ instances.

We will not give in detail an example that corresponds to Example 4.16, since the behavior of LIG^* compared to the eager method is exactly what can be expected from Examples 4.16 and 4.24: while the eager method generates the full set of instances, behavior of LIG^* with maximal inferences depends very much on the models delivered by the SMT solver. It will terminate as soon as it obtains a model not containing $\neg(f_3(a_1) \leq f_3(b_1))$, the only negative literal with extension symbols in the problem. This can happen without generating any instances, or never at all. In the latter case, LIG^* will only terminate after generating the full set of instances. LIG^*_{uc} can also terminate when we obtain a model that does not contain $\neg(f_3(a_1) \leq f_3(b_1))$, or when we cannot generate any of the six instances produced in Example 4.24. Since G in this case does not contain $\neg(f_3(c_1) \leq f_3(d_1))$, only three out of those six can be generated, and the procedure will terminate after at most 3 steps.

4.3.3 Correctness of LIG^* and LIG^*_{uc}

In the following we state that LIG^* and LIG^*_{uc} are sound, complete, and terminating. The proofs are direct liftings of the corresponding proofs in Section 4.2.3, with $\mathcal{K}_1 \cup \ldots \cup \mathcal{K}_m$ replacing $\mathcal{K}$ and G_0 (as defined in Sections 4.3.1 and 2.6.1) replacing $\mathcal{K}[G]$.

Definition 4.25 (LIG^*-derivation)**.** Consider a base theory $\mathcal{T}_0$ and a chain of extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \ldots \subseteq \mathcal{T}_m$, with $\mathcal{T}_{j+1} = \mathcal{T}_j \cup \mathcal{K}^0_{j+1}$. 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 $0 \leq j \leq n-1$, $\mathcal{T}_j \subseteq \mathcal{T}_j \cup \mathcal{K}^0_{j+1}$ 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 $(\mathcal{K}^i_1, \ldots, \mathcal{K}^i_m, G^i)$.

An LIG^*_{uc}-derivation is defined in the obvious way.

Theorem 4.26 (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^*- or LIG^*_{uc}-derivation. If $\mathcal{T}_0 \cup G^n \models \Box$, then $\mathcal{T}_m \cup G^0 \models \Box$.*

Proof: All elements of G^n are instances of clauses in $\mathcal{K}^0_1 \cup \ldots \cup \mathcal{K}^0_m \cup G^0$. Thus, their unsatisfiability implies unsatisfiability of $\mathcal{K}^0_1 \cup \ldots \cup \mathcal{K}^0_m \cup G^0$. □

Theorem 4.27 (Completeness of LIG^*)**.** *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 is $\mathcal{T}_0$-satisfiable and $(\mathcal{K}^n_1, \ldots, \mathcal{K}^n_m, G^n)$ is saturated under LIG^* with respect to a given selection function* sel *and a partial Herbrand model $\mathcal{M}$ of G^n, then $\mathcal{T}_m \cup G^0$ is satisfiable.*

Proof: Let $\mathcal{K}^0_1, \ldots, \mathcal{K}^0_m, G^0, \mathcal{K}^n_1, \ldots, \mathcal{K}^n_m, G^n, \mathsf{sel}$ and $\mathcal{M}$ as defined above, and let G_0 be the search space of local instances defined by $\mathcal{K}^0_1, \ldots, \mathcal{K}^0_m$ and G^0. $\mathcal{M}$ is a partial Herbrand model of $\mathcal{T}_0$ and G^n, where $G^n \subseteq G_0$. We extend $\mathcal{M}$ to a partial Herbrand model of $\mathcal{T}_0$ and G_0 in the following way: for every $C_i \in G_0 \setminus G^n$ we can find a non-ground clause $D_i \in (\mathcal{K}^n_1 \cup \ldots \cup \mathcal{K}^n_m)$ s.t. $D_i\sigma_i = C_i$ and D_i is a most specific generalization of C_i wrt. $(\mathcal{K}^n_1 \cup \ldots \cup \mathcal{K}^n_m)$. 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 G_0. Furthermore, $\mathcal{M}'$ cannot be contradictory

wrt. $\mathcal{T}_0$: 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. $(\mathcal{K}_1^n \cup \ldots \cup \mathcal{K}_m^n)$, the produced instances are neither in G^n nor $(\mathcal{K}_1^n \cup \ldots \cup \mathcal{K}_m^n)$, thus contradicting our assumption that $(\mathcal{K}_1^n, \ldots, \mathcal{K}_m^n, G^n)$ is saturated under LIG^* wrt. sel and $\mathcal{M}$. As $\mathcal{M}'$ is not contradictory wrt. $\mathcal{T}_0$ and contains one literal out of each clause in G_0, we can conclude that $\mathcal{T}_0 \cup G_0$ is satisfiable. By locality, $\mathcal{T}_0 \cup \mathcal{K}_1^0 n \cup \ldots \cup \mathcal{K}_m^0 \cup G^0$ is satisfiable iff $\mathcal{T}_0 \cup G_0$ is satisfiable. □

Theorem 4.28 (Completeness of LIG^*_{uc})**.** *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^*_{uc}*-derivation. If* G^n *is* $\mathcal{T}_0$*-satisfiable and* $(\mathcal{K}_1^n, \ldots, \mathcal{K}_m^n, G^n)$ *is saturated under* LIG^*_{uc} *with respect to a given selection function* sel *and a partial Herbrand model* $\mathcal{M}$ *of* G^n*, then* $\mathcal{T}_m \cup G^0$ *is satisfiable.*

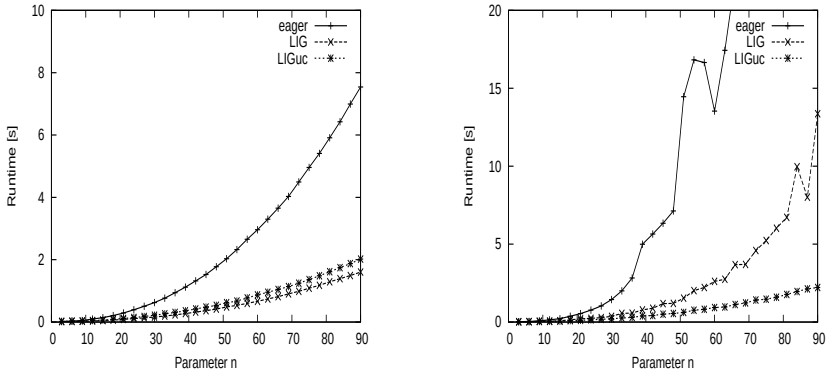
Proof: Let $\mathcal{K}_1^0, \ldots, \mathcal{K}_m^0, G^0, \mathcal{K}_1^n, \ldots, \mathcal{K}_m^n, G^n, \mathsf{sel}$ and $\mathcal{M}$ as defined above, and let G_0 be the search space of local instances defined by $\mathcal{K}_1^0, \ldots, \mathcal{K}_m^0$ and G^0. $\mathcal{M}$ is a partial Herbrand model of $\mathcal{T}_0$ and G^n, where $G^n \subseteq G_0$. We extend $\mathcal{M}$ to a partial Herbrand model of $\mathcal{T}_0$ and G_0 in the following way: for every $C_i \in G_0 \setminus G^n$ we can find a non-ground clause $D_i \in (\mathcal{K}_1^n \cup \ldots \cup \mathcal{K}_m^n)$ s.t. $D_i\sigma_i = C_i$ and D_i is a most specific generalization of C_i wrt. $(\mathcal{K}_1^n \cup \ldots \cup \mathcal{K}_m^n)$. 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 G_0. Furthermore, $\mathcal{M}'$ cannot be contradictory wrt. $\mathcal{T}_0$: if this was the case, we would have an unsatisfiable core of $\mathcal{M}'$, containing at least one of the $\mathsf{sel}(D_i)\sigma_i$ (since $\mathcal{M}$ itself is not contradictory). Thus, we would have an LIG^*_{uc} inference with a subset of 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. $(\mathcal{K}_1^n \cup \ldots \cup \mathcal{K}_m^n)$, the produced instances are neither in G^n nor $(\mathcal{K}_1^n \cup \ldots \cup \mathcal{K}_m^n)$, thus contradicting our assumption that $(\mathcal{K}_1^n, \ldots, \mathcal{K}_m^n, G^n)$ is saturated under LIG^*_{uc} wrt. sel and $\mathcal{M}$. As $\mathcal{M}'$ is not contradictory wrt. $\mathcal{T}_0$ and contains one literal out of each clause in G_0, we can conclude that $\mathcal{T}_0 \cup G_0$ is satisfiable. By locality, $\mathcal{T}_0 \cup \mathcal{K}_1^0 n \cup \ldots \cup \mathcal{K}_m^0 \cup G^0$ is satisfiable iff $\mathcal{T}_0 \cup G_0$ is satisfiable. □

Theorem 4.29 (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^* *and* LIG^*_{uc} *terminate after a finite number of inferences.*

Proof: LIG^*- and LIG^*_{uc}-inferences produce only clauses that are both instances of a clause in one of the $\mathcal{K}_j$ and generalizations of clauses in $\mathcal{K}_j[\bigcup_{j<k\leq m+1} \mathcal{I}_k]$, where for every sel, $\mathcal{K}_j[\bigcup_{j<k\leq m+1} \mathcal{I}_k] \subseteq \mathcal{K}_j[G_j]$. For a given input $(\mathcal{K}_1, \ldots, \mathcal{K}_n, G)$, only finitely many clauses are both instances of clauses in one of the $\mathcal{K}_j$ and generalizations of clauses in $\mathcal{K}_j[G_j]$, and thus there are only finitely many possible LIG^*- and LIG^*_{uc}-inferences. □

4.4 Implementation: iLoRe

We implemented the incremental approaches LIG, LIG_{uc} and LIG^* on top of an existing OCaml implementation of the eager approach. The iLoRe tool uses Z3 [11] as a black-box SMT solver, because Z3 provides an OCaml API as well as the necessary functionality (model generation, incremental addition of



(a) Example 4.15 (b) Example 4.16

Figure 4.1: Runtime comparison of the three approaches on parameterized examples

constraints and computation of unsatisfiable cores). For our benchmark tests we used a machine with Intel Xeon CPU at 3.16 GHz and 16 GB of RAM.

Runtime Comparison. We tested our implementation on four sets of benchmarks. The implementation of LIG_{uc} is still preliminary, therefore it is only included in two of the four comparisons below.

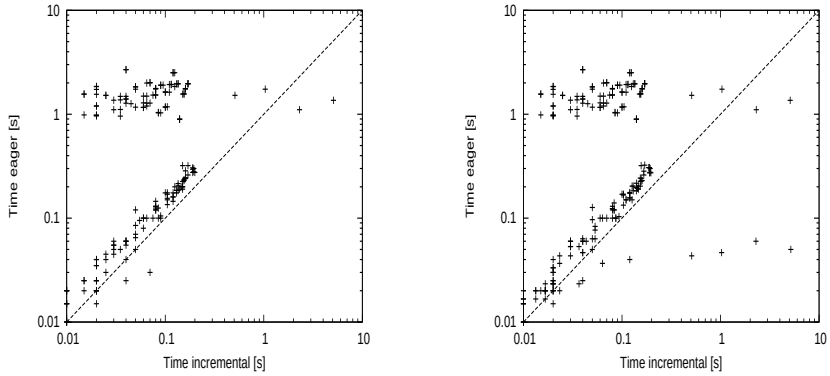
In Figure 4.1 (a) we see runtimes of all three approaches on the parameterized problem from Example 4.15. The incremental approaches outperform the eager instantiation substantially, while LIG_{uc} is a bit slower than LIG. Figure 4.1 (b) shows runtimes for Example 4.16: here, LIG_{uc} is significantly faster than LIG.

In Figure 4.2, we see runtimes of LIG and the eager approach on a set of verification benchmarks taken from SMT-LIB, an independent benchmark library for SMT problems. More specifically, we compared runtimes of the two approaches on modified versions of all problems from the `QF_UFLIA/wisas` section of SMT-LIB. The problems in this section were originally benchmarks of the Wisconsin Safety Analyzer. They are originally in the universal fragment of $\mathcal{T}_{\mathbb{Z}}$ with additional free function symbols.

We checked satisfiability of these benchmarks in two extended theories, which assume one of the functions to be monotone or injective, respectively. All of the problems turned out to be unsatisfiable in both of the extended theories. Figure 4.2 (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 almost all of the problems.

For the set of all benchmarks from the `QF_UFLIA/wisas` section of SMT-LIB, we had mixed results (see Figure 4.2 (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.

Memory Consumption. In Table 4.4, we compare memory consumption of



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

Figure 4.2: Runtime comparison of *LIG* and eager approach on verification problems

the different approaches. The table shows the memory in use at exit, as reported by the Valgrind tool. We have chosen a number of problems that we also used for timing benchmarks (with increasing complexity, otherwise random). In the table, "Blowup_n" denotes the parameterized example with parameter n. The problems starting with "xs" are those taken from the SMT-LIB, "(+inj)" means that we added an injectivity axiom to the original problem, "(+mon)" means we added a monotonicity axiom. As for the time comparison, we cannot include LIG_{uc} in the comparison for the SMT-LIB examples.

The results are not very surprising: the incremental approaches need less memory than the eager approach, and the version with unsatisfiable core (where applicable) needs less than the one with maximal inferences.

4.5 Results

In this chapter we have introduced several methods for incremental generation of the instances that make up the search space of local reasoning. We have given examples where incremental approaches can terminate after traversing only a small subset of the search space, and have demonstrated with an implementation that this theoretical benefit is also reflected in a better time- and space-efficiency.

Our tests with independent verification-related benchmarks should be a good indication that efficiency is not only considerably increased for certain toy examples, but also for a significant class of real-world applications.

Table 4.1: Comparison: Memory in use at exit (in MB)

Problem	eager	*LIG*	LIG_{uc}
Blowup_3	1.466	1.364	1.287
Blowup_15	7.987	3.033	1.682
Blowup_75	171.442	47.762	6.150
xs_7_7 (+inj)	1.729	1.591	-
xs_7_7 (+mon)	1.777	1.679	-
xs_10_10 (+inj)	4.901	1.781	-
xs_10_10 (+mon)	2.125	1.839	-
xs_15_20 (+inj)	4.857	2.098	-
xs_15_20 (+mon)	2.647	2.245	-
xs_21_31 (+inj)	4.993	2.612	-
xs_21_31 (+mon)	3.683	3.077	-
xs_27_47 (+inj)	5.100	3.552	-
xs_27_47 (+mon)	4.372	4.153	-
xs_34_44 (+inj)	6.490	4.358	-
xs_34_44 (+mon)	5.978	5.252	-

Chapter 5

Application: Verification of Parameterized Systems

In Chapters 2 and 3 we have introduced the approach of local reasoning, as well as many examples of local theory extensions. So far, we have not given detailed examples of interesting applications.

In this chapter we show how local reasoning can be used to solve a class of verification problems for parameterized systems. In Section 5.1, we define the notation of our formal models and verification problems and mention a method how to obtain such formal system models from a suitable specification language. In Section 5.2, we show that local reasoning gives us decision procedures for a certain class of systems and properties, provided that they can be expressed as chains of local theory extensions.

In the rest of Chapter 5, we demonstrate that interesting systems can be modeled in this class by describing in detail a case study taken from the European Train Control System (ETCS) standard. In Section 5.3, we introduce the case study informally. In Section 5.4 we present in detail a simple model of the case study, including verification tasks for proving a safety property and how we can solve these problems with local reasoning. We present different extensions of this model with increasing complexity in Section 5.5.

The main results of this chapter have been published in 2006 [39] and 2007 [40, 17].

5.1 Formal Specification and Verification

A major application area of automated reasoning is the verification of systems. In order to apply automated reasoning methods to a system, we need a formal system specification, called a *system model*, and verification methods for proving properties of such system models.

In this section we introduce *transition constraint systems*, which we use as system models. We explain how safety properties of these systems can be verified by *invariant checking* and *bounded model checking*.

5.1.1 Formal System Model

Definition 5.1 (Transition constraint system)**.** Let $\Pi_0 = (S, \Sigma_0, \mathsf{Pred})$ be a signature and $\mathcal{T}_0$ a Π_0-theory. Let V be a set of constant symbols and Σ a set of non-constant function symbols, such that V, Σ and Σ_0 are pairwise disjoint.

Let $V' = \{\ v' \mid v \in V\ \}$ and $\Sigma' = \{\ f' \mid f \in \Sigma\ \}$, and define $\Pi = (S, \Sigma_0 \cup \Sigma \cup V, \mathsf{Pred})$ and $\Pi' = (S, \Sigma_0 \cup \Sigma \cup \Sigma' \cup V \cup V', \mathsf{Pred})$.

Then, a *transition constraint system* (TCS) (with background theory $\mathcal{T}_0$) consists of:

- the set of constant symbols V, called *system variables*,
- the set of function symbols Σ, called *system functions*,
- a Π-formula (Init), called the *initial condition*, and
- a Π'-formula (Update), called the *transition relation*.

The system variables V model properties of the overall system. System functions Σ model either properties of the overall system which depend on some other values, or local properties of a parameterized number of components of the system. In the latter case, $f(i)$ can be seen as a local variable f of some component i.

Definition 5.2 (System state)**.** For a given TCS, a *system state* is a Π-structure M such that $M \models \mathcal{T}_0$, i.e. a structure that is a model of the background theory and assigns values to system variables and system functions.[1]

The initial condition (Init) describes the set of initial state of the system, using symbols in the given background signature Π_0 as well as system variables V and system functions Σ:

Definition 5.3 (Initial state)**.** For a given TCS, an *initial state* is a system state M such that $M \models (\mathsf{Init})$.

The transition relation (Update) describes the relation between system variables and system functions before and after a transition of the system, using signature Π'. V and Σ refer to the system state before the transition, V' and Σ' to the state after the transition:

Definition 5.4 (Pre-state, post-state, successor)**.** For a Π'-structure N, let its *pre-state* be the reduct of N to the signature Π. We obtain its *post-state* by considering the reduct of N to the signature $(S, \Sigma_0 \cup \Sigma' \cup V', \mathsf{Pred})$, and renaming constant symbols $v' \in V$ to v, and function symbols $f' \in \Sigma'$ to f (such that the result is again a Π-structure).

A system state M' is a *successor* of a system state M if there exists a Π'-structure N such that M is the pre-state of N, M' is the post-state of N and $N \models (\mathsf{Update})$.

[1] It may seem unusual to the reader that we include the structure for the background theory in the system state. Usually, the system state would be defined as a valuation of the symbols in V and Σ, based on a model of the background theory. Logically, the two possible definitions are equivalent, since the reduct of any Π-model of $\mathcal{T}_0$ to Π_0 is a Π_0-model of $\mathcal{T}_0$. The definition we chose is more convenient for what follows.

Definition 5.5 (Reachable state)**.** A system state M_n is *reachable* if there is a sequence of system states $M_1, \ldots, M_n$ such that M_1 is an initial state and M_{i+1} is a successor of M_i for $1 \leq i \leq n-1$.

Definition 5.6 (Safety property, unsafe state, safety)**.** A *safety property* of a TCS T is a Π-formula. For a given safety property (Safe), an *unsafe state* is a system state M with $M \models \neg(\mathsf{Safe})$. A TCS T is *safe* if no unsafe states are reachable.

We present an approach for proving safety of TCSs in Section 5.1.2.

5.1.2 Invariant Checking of TCSs

A method for proving safety of systems is *invariant checking*. The idea is to find an invariant of the system that implies the safety property.

Definition 5.7 (Invariant, inductive invariant, safety invariant)**.** A Π-formula (Inv) is an *invariant* of a TCS $T = (V, \Sigma, (\mathsf{Init}), (\mathsf{Update}))$ with background theory $\mathcal{T}_0$ if $M \models (\mathsf{Inv})$ for all reachable system states M.

An invariant (Inv) is *inductive* if $\mathcal{T}_0 \cup (\mathsf{Inv}) \cup (\mathsf{Update}) \models (\mathsf{Inv}')$, where (Inv') is obtained from (Inv) by replacing every variable $v \in V$ with v' and every function symbol $f \in \Sigma$ with f'.

An invariant (Inv) is a *safety invariant* for a given safety property (Safe) if $\mathcal{T}_0 \cup (\mathsf{Inv}) \models (\mathsf{Safe})$

For verification of safety properties, we are interested in inductive safety invariants, because they allow us to reduce verification problems to first-order satisfiability problems.

Invariant checking. Consider a TCS $T = (P, V, \Sigma, (\mathsf{Init}), (\mathsf{Update}))$ with background theory $\mathcal{T}_0$. To show that a Π-formula (Inv) is an inductive safety invariant of T for a given safety property (Safe), we need to prove

(1) $\mathcal{T}_0 \cup (\mathsf{Inv}) \models (\mathsf{Safe})$,

(2) $\mathcal{T}_0 \cup (\mathsf{Init}) \models (\mathsf{Inv})$, and

(3) $\mathcal{T}_0 \cup (\mathsf{Inv}) \cup (\mathsf{Update}) \models (\mathsf{Inv}')$.

Note that these satisfiability problems can be arbitrarily hard, depending on the background theory $\mathcal{T}_0$ and formulas $(\mathsf{Init}), (\mathsf{Update}), (\mathsf{Safe})$ and (Inv). In Section 5.2 we will introduce restrictions on these formulas and the background theory that allow us to decide the given problems with the local reasoning approach.

Remark. In many cases we can simplify the verification conditions above by explicitly separating constant and function symbols that do not change their valuation during an execution of the system from those that do change their valuation.

Let $P \subseteq V$ be the set of system variables which are kept fixed by (Update), i.e. $\mathcal{T}_0 \cup (\mathsf{Update}) \models v = v'$ for all $v \in P$. We call them *system parameters*. Similarly, let $\Sigma_P \subseteq \Sigma$ be the set of function symbols which are kept fixed by (Update), and let $\Pi_P = (S, \Sigma_0 \cup P \cup \Sigma_P, \mathsf{Pred})$. Then, let $(\mathsf{Init}) = (\mathsf{Global}) \cup$

(Init_V), where (Global) is the maximal subset of (Init) which contains only Π_P-formulas, and $(\mathsf{Global}) \cap (\mathsf{Init}_V) = \varnothing$. We call (Global) the *global constraints* of T. Since the valuation of all symbols in (Global) is fixed during an execution of the system, we can assume that (Global) holds in every reachable state of the system. Therefore, we do not need to introduce primed variants of these symbols for the post-state of a transition.

With the separation explained above, the invariant checking problem is equivalent to proving

(1) $\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Inv}) \models (\mathsf{Safe})$,

(2) $\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Init}_V) \models (\mathsf{Inv})$, and

(3) $\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Inv}) \cup (\mathsf{Update}) \models (\mathsf{Inv}')$,

where (Update) and (Inv') are not in Π', but in $\Pi'_V = (S, \Sigma_0 \cup \Sigma \cup (V' \setminus P') \cup (\Sigma' \setminus \Sigma'_P))$, i.e. system parameters $v \in P$ and system functions $f \in \Sigma_P$ only appear in their original form, not as primed variants v', f'.

Invariant checking crucially depends on the presence or the ability to find an inductive safety invariant. If for a given candidate invariant (Inv_1) we can prove (1) and (2), but not (3), this does not mean that the system is not safe: there may be a formula (Inv_2) with $\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Inv}_2) \models (\mathsf{Inv}_1)$, and such that (Inv_2) is inductive. Finding such inductive invariants is an own branch of research which we do not consider here. The problem is in general undecidable, but if we cannot come up with an invariant we can still resort to bounded model checking, described in Section 5.1.3.

5.1.3 Bounded Model Checking of TCSs

The invariant checking approach from Section 5.1.2 fails if we cannot find an inductive safety invariant for the desired safety property (Safe). The idea of *bounded model checking* (BMC) is to prove safety only for a finite number of transition steps.

Bounded model checking. Consider a TCS $T = (V, \Sigma, (\mathsf{Init}), (\mathsf{Update}))$ with background theory $\mathcal{T}_0$.

For a given formula F and $i \in \mathbb{N}$, let F_i be the result of replacing in F all variables $v \in V$ by v_{i-1}, all $v' \in V'$ by v_i, all functions $f \in \Sigma$ by f_{i-1}, and all $f' \in \Sigma'$ by f_i.

To show that the system T cannot reach an unsafe state within n transition steps, we need to prove

$$\mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \textstyle\bigcup_{i=1}^{k} (\mathsf{Update}_i) \models (\mathsf{Safe}_k),$$

for $1 \leq k \leq n$.

While this approach does not depend on an inductive invariant, the satisfiability problems can also be arbitrarily hard in this case, depending on the background theory $\mathcal{T}_0$ and formulas $(\mathsf{Init}), (\mathsf{Update})$ and (Safe), as well as the bound n. In Section 5.2, we will introduce restrictions on these formulas and the background theory that allow us to decide the given problems with the local reasoning approach (for any given n).

Remark. Similar to the invariant checking approach, we can simplify the proof tasks for BMC by separating symbols that change their valuation during an execution of the system from those that do not change. The BMC problem is then equivalent to

$$\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{k} (\mathsf{Update}_i) \models (\mathsf{Safe}_k),$$

for $1 \leq k \leq n$, where $(\mathsf{Init}_0), (\mathsf{Update}_i)$ and (Safe_k) only contain variants with subscript of those symbols that change their valuation during an execution of the system.

5.1.4 Translation from a Formal Specification Language

Complex systems consisting of several interacting components, like the examples we will present in Sections 5.4 and 5.5, arise in a natural way in a wide range of applications. In order to make automatic verification of such systems feasible in industrial application, one needs not only the verification techniques we introduced thus far, but also a suitable *specification language*.

We want to mention briefly the specification language we used for modeling such complex systems [17]. The detailed approach of specification in this language and translation of specifications into TCSs is not part of this thesis.

Combination of processes, data and time. In the specification of complex systems, one needs to take several aspects into account: On an abstract level, the control flow of such systems can be divided into processes, which are sequences of high-level actions. These sequences are subject to certain restrictions: certain actions must be preceded or followed by other actions, and different processes may have to synchronize on some of their actions. Then, for every action of the system, we can define what are the preconditions on the values of system variables and system functions, as well as the resulting changes on their values. Finally, actions of the system can be subject to timing restrictions.

In our work, we used a combined language CSP-OZ-DC (COD) [34, 32] that allows us to specify all of these aspects separately. Communicating Sequential Processes (CSP) are used to model the high-level control flow, Object-Z (OZ) for modeling valuations of system variables and functions, and the Duration Calculus (DC) for timing restrictions.

Translation and Verification of COD specifications. Existing verification approaches for COD [33, 18] are only able to deal with simple background theories. We extended these approaches to complex theories, which allows us to verify systems with a parametric number of components and complex data types.

Verification of COD specifications in these previous approaches is based on a translation into Phase Event Automata. We showed that these can easily be converted into TCSs as introduced in Section 5.1.1. Furthermore, we extended the approach to deal with timing parameters in the OZ part, and background theories more complex than pure linear arithmetic.

5.2 Local Reasoning in Verification of Parameterized Systems

In this section we show under which assumptions on a TCS T, the background theory $\mathcal{T}_0$ and the formulas (Safe) and (Inv) we can decide the satisfiability problems resulting from invariant checking and bounded model checking of TCSs.

Deciding invariant checking problems with local reasoning. For invariant checking, we need to solve the following satisfiability problems:

(1) $\mathcal{T}_0 \cup (\mathsf{Inv}) \models (\mathsf{Safe})$,

(2) $\mathcal{T}_0 \cup (\mathsf{Init}) \models (\mathsf{Inv})$, and

(3) $\mathcal{T}_0 \cup (\mathsf{Inv}) \cup (\mathsf{Update}) \models (\mathsf{Inv}')$.

For solving (1), consider $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Inv})$ as a theory extension. By Theorem 2.41, we can decide the universal fragment of $\mathcal{T}_0 \cup (\mathsf{Inv})$ if either the extension is $\forall\exists$-reducing and the $\forall\exists$-fragment of $\mathcal{T}_0$ is decidable, or if the extension is universally reducing and the universal fragment of $\mathcal{T}_0$ is decidable. If we can decide the universal fragment of $\mathcal{T}_0 \cup (\mathsf{Inv})$ and (Safe) is universal, then we can decide (1).

For solving (2), consider $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Init})$ as a theory extension. With the same argument as above, we can decide the universal fragment of $\mathcal{T}_0 \cup (\mathsf{Init})$ if either the extension is $\forall\exists$-reducing and the $\forall\exists$-fragment of $\mathcal{T}_0$ is decidable, or if the extension is universally reducing and the universal fragment of $\mathcal{T}_0$ is decidable. If we can decide the universal fragment of $\mathcal{T}_0 \cup (\mathsf{Init})$ and (Inv) is universal, then we can decide (2).

Finally, for solving (3), assume that we have solved (1), i.e. we can decide the universal fragment of $\mathcal{T}_0 \cup (\mathsf{Inv})$. Then we can decide the universal fragment of $\mathcal{T}_0 \cup (\mathsf{Inv}) \cup (\mathsf{Update})$ if the extension $\mathcal{T}_0 \cup (\mathsf{Inv}) \subseteq \mathcal{T}_0 \cup (\mathsf{Inv}) \cup (\mathsf{Update})$ is universally reducing. If we can decide the universal fragment of $\mathcal{T}_0 \cup (\mathsf{Inv}) \cup (\mathsf{Update})$ and (Inv) is universal, then we can decide (3).

Thus, the following is a consequence of Theorem 2.41 and the considerations above:

Theorem 5.8. *Let $T = (V, \Sigma, (\mathsf{Init}), (\mathsf{Update}))$ be a TCS with a background theory $\mathcal{T}_0$ with signature Π_0, and let* (Safe) *and* (Inv) *be universal Π-formulas (Π as defined above). Assume that the theory extension $\mathcal{T}_0 \cup (\mathsf{Inv}) \subseteq \mathcal{T}_0 \cup (\mathsf{Inv}) \cup (\mathsf{Update})$ is universally reducing.*

i) If the theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Init})$ and $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Inv})$ are $\forall\exists$-reducing and the $\forall\exists$-fragment of $\mathcal{T}_0$ is decidable, then the invariant checking problem wrt. T, (Safe) *and* (Inv) *is decidable.*

ii) If the theory extensions $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Init})$ and $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Inv})$ are universally reducing and the universal fragment of $\mathcal{T}_0$ is decidable, then the invariant checking problem wrt. T, (Safe) *and* (Inv) *is decidable.*

Even if not all of the conditions above are satisfied, we may still be able to use local reasoning to solve the given satisfiability problems by considering partitionings of the formulas (Init), (Inv) and (Update) into smaller sets and reasoning about chains of extensions wrt. these partitionings.

E.g., if the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Init})$ is neither universally nor $\forall\exists$-reducing, we can consider a partitioning $\mathcal{K}_1^{\mathsf{Init}} \cup \ldots \cup \mathcal{K}_m^{\mathsf{Init}}$ of (Init) and the *corresponding chain of extensions*

$$\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup \mathcal{K}_1^{\mathsf{Init}} \subseteq \ldots \subseteq \mathcal{T}_0 \cup \bigcup_{i=1}^{m} \mathcal{K}_i^{\mathsf{Init}}.$$

By Theorem 2.55, the universal theory of $\mathcal{T}_0 \cup (\mathsf{Init})$ is decidable if either this chain of extensions is $\forall\exists$-reducing and the $\forall\exists$-fragment of $\mathcal{T}_0$ is decidable, or this chain of extensions is universally reducing and the universal fragment of $\mathcal{T}_0$ is decidable. By considering such partitionings for $(\mathsf{Init}), (\mathsf{Inv})$ and (Update) (and the corresponding chains of theory extensions), we obtain the following generalization of Theorem 5.8:

Theorem 5.9. *Let $T = (V, \Sigma, (\mathsf{Init}), (\mathsf{Update}))$ be a TCS with a background theory $\mathcal{T}_0$ with signature Π_0, and let (Safe) and (Inv) be universal Π-formulas (Π as defined above). Assume that there is a partitioning $\bigcup_{i=1}^{n} \mathcal{K}_i^{\mathsf{Update}}$ of (Update) such that the corresponding chain of extensions $\mathcal{T}_0$ to $\mathcal{T}_0 \cup (\mathsf{Update})$ is universally reducing.*

i) If there are partitionings $\bigcup_{i=1}^{l} \mathcal{K}_i^{\mathsf{Init}}$ of (Init) and $\bigcup_{i=1}^{m} \mathcal{K}_i^{\mathsf{Inv}}$ of (Inv) such that the corresponding chains of extensions from $\mathcal{T}_0$ to $\mathcal{T}_0 \cup (\mathsf{Init})$ and from $\mathcal{T}_0$ to $\mathcal{T}_0 \cup (\mathsf{Inv})$ are $\forall\exists$-reducing and the $\forall\exists$-fragment of $\mathcal{T}_0$ is decidable, then the invariant checking problem wrt. T, (Safe) and (Inv) is decidable.

ii) If there are partitionings $\bigcup_{i=1}^{l} \mathcal{K}_i^{\mathsf{Init}}$ of (Init) and $\bigcup_{i=1}^{m} \mathcal{K}_i^{\mathsf{Inv}}$ of (Inv) such that the corresponding chains of extensions from $\mathcal{T}_0$ to $\mathcal{T}_0 \cup (\mathsf{Init})$ and from $\mathcal{T}_0$ to $\mathcal{T}_0 \cup (\mathsf{Inv})$ are universally reducing and the universal fragment of $\mathcal{T}_0$ is decidable, then the invariant checking problem wrt. T, (Safe) and (Inv) is decidable.

Deciding BMC problems with local reasoning. For bounded model checking, we need to solve satisfiability problems

$$\mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{k} (\mathsf{Update}_i) \models (\mathsf{Safe}_k),$$

for $1 \leq k \leq n$.

To this end, consider $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Init}_0) \subseteq \mathcal{T}_0 \cup (\mathsf{Init}_0) \cup (\mathsf{Update}_1) \subseteq \ldots \subseteq \mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{k} (\mathsf{Update}_i)$ as a chain of theory extensions. By Theorem 2.55, we can decide the universal fragment of $\mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{k} (\mathsf{Update}_i)$ if either this chain is $\forall\exists$-reducing and the $\forall\exists$-fragment of $\mathcal{T}_0$ is decidable, or this chain is universally reducing and the universal fragment of $\mathcal{T}_0$ is decidable.

If we can decide the universal fragment of $\mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{k} (\mathsf{Update}_i)$ for every k and (Safe) is universal (and thus (Safe_k) is universal for every k), then we can decide the given satisfiability problem for any k.

Again, we formulate the consequence of these considerations and Theorems 2.41 and 2.55:

Theorem 5.10. *Let $T = (V, \Sigma, (\mathsf{Init}), (\mathsf{Update}))$ be a TCS with a background theory $\mathcal{T}_0$ with signature Π_0, and let (Safe) be a universal Π-formula (Π as defined above). Assume that the extensions $\mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{j-1} (\mathsf{Update}_i) \subseteq \mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{j} (\mathsf{Update}_i)$, for every j, are universally reducing.*

i) If the theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Init}_0)$ is $\forall\exists$-reducing and the $\forall\exists$-fragment of $\mathcal{T}_0$ is decidable, then the bounded model checking problem wrt. T and (Safe) is decidable for any k.

ii) If the theory extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Init}_0)$ is universally reducing and the universal fragment of $\mathcal{T}_0$ is decidable, then the bounded model checking problem wrt. T and (Safe) is decidable for any k.

If the premises of this theorem are not satisfied, like in the case of invariant checking, we may find partitionings of (Init_0) and (Update_i) such that the corresponding chain of extensions from $\mathcal{T}_0$ to $\mathcal{T}_0 \cup (\mathsf{Init}_0)$ is universally or $\forall\exists$-reducing and the corresponding chains of extensions from $\mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{j-1}(\mathsf{Update}_i)$ to $\mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{j}(\mathsf{Update}_i)$ (for $1 \leq j \leq k$) are universally reducing.

Theorem 5.11. *Let $T = (V, \Sigma, (\mathsf{Init}), (\mathsf{Update}))$ be a TCS with a background theory $\mathcal{T}_0$ with signature Π_0, and let (Safe) be a universal Π-formula (Π as defined above). Assume that there is a partitioning of (Update_i) such that the corresponding chain of extensions from $\mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{j-1}(\mathsf{Update}_i)$ to $\mathcal{T}_0 \cup (\mathsf{Init}_0) \cup \bigcup_{i=1}^{j}(\mathsf{Update}_i)$, for every $j \geq 1$, is universally reducing.*

i) If there is a partitioning of (Init_0) such that the corresponding chain of extensions from $\mathcal{T}_0$ to $\mathcal{T}_0 \cup (\mathsf{Init}_0)$ is $\forall\exists$-reducing and the $\forall$ exists-fragment of $\mathcal{T}_0$ is decidable, then the bounded model checking problem wrt. T and (Safe) is decidable for any k.

ii) If there is a partitioning of (Init_0) such that the corresponding chain of extensions from $\mathcal{T}_0$ to $\mathcal{T}_0 \cup (\mathsf{Init}_0)$ is universally reducing and the universal fragment of $\mathcal{T}_0$ is decidable, then the bounded model checking problem wrt. T and (Safe) is decidable for any k.

Modeling systems with local theory extensions. Thus, the remaining problem for both approaches is to show that the resulting extensions or chains of extensions are universally or $\forall\exists$-reducing, or to find partitionings that satisfy these properties. The main problem is establishing locality of the extensions, which can be done in two ways: we can show locality of extensions by hand, as we did in Section 3, or we can try to model the system such that the resulting theory extensions are already known to be local.

In the next sections we show that the locality results given in Section 2.5 and Chapter 3 can be used as a toolbox to model interesting parameterized systems, and that our implementation of the local reasoning approach can be used to decide invariant checking and BMC problems for these systems automatically.

5.3 Description of the ETCS Case Study

In this Section we introduce informally a case study taken from the European Train Control System standard (cf. [16]).

The ETCS standard has been issued by the European Commission and is an international system that shall replace traditional, national train control systems in the future. The main goals are to ensure cross-border interoperability and improve railway safety and track utilization. In the final ETCS implementation-level 3, existing national systems for detection of train speed, location, and

integrity will not be used anymore. Instead, values of these properties for a moving train are detected by the train's on-board ETCS unit in cooperation with a so-called radio block center (RBC), which controls the traffic in a well-defined area and grants movement authorities (MA) to trains.

Over a radio connection, the trains communicate information about their current position (and possibly other properties) to the RBC, which collects this information from all trains in its area. The collected information is used to issue MAs to all trains, and every single train then has the responsibility to ensure that it does not move beyond its MA. Together, this behavior should ensure that the overall system is well-behaved, i.e. the trains move in a reliable and safe way.

ETCS implementation-level 3 also considers emergency messages: their function is to ensure that in case of an accident, like a malfunction of a train or a blocked rail track, every train that is affected comes to a standstill before colliding with other trains. To this end, a train that detects an emergency can send a special emergency message to the RBC, which is then forwarded to every train approaching the danger position. The trains acknowledge the message and have to be able to brake to a standstill before reaching the position of another train or the danger point. If necessary, this includes automatic application of the emergency brakes if the driver of the train does not react in time.

In the following sections, we will introduce successively more complex models of an RBC-controlled rail track with a parametric number of trains. We start with a very simple model and show that we can extend the complexity of the system significantly, while still being able to reduce the problems of invariant checking and bounded model checking to decidable satisfiability problems. All the time, the number of trains as well as several other values of the system description will be considered as parameters. That is, we prove safety properties for all possible values of these parameters at once, or even deduce certain constraints on the parameters that guarantee safety.

5.4 Verification of a Simple Model

We start with a very simple model of the case study, where we abstract from the communication issues and assume that at given points in time the system has accurate knowledge about the positions of all trains. At these points, the positions are updated and the behavior of the trains until the next update is determined. We assume that all trains move according to the same rules, which determine the speed until the next update by comparing the distance to the preceding train to a given safety distance. For now, we do not consider the length of trains, i.e. they are just points on a line. The number of trains on that line is arbitrary, but remains fixed during a run of the system.

Convention. In this and the following sections, we use the separation of system variables and system functions into those which do change their valuation under (Update), and those that do not, as explained in the remark in Section 5.1.2 (and in Section 5.1.3 for BMC).

In Section 5.5 we will introduce several extensions of this model that retract some of these simplifications: we will consider trains that enter and leave the area controlled by the RBC, we will introduce a safety condition that takes

into account the length of trains and finally we will also allow non-deterministic emergencies in addition to the standard behavior of the system.

5.4.1 TCS of the Simple Model

We define a TCS with background theory $\mathcal{T}_0 = \mathcal{T}_{\mathbb{Z}} \cup \mathcal{T}_{\mathbb{R}}$, where $\mathbb{Z}$ is used as an index sort for our components[2]. For the simple model, the set of system parameters is $P = \{ \Delta_t, \mathsf{d}, \mathsf{min}, \mathsf{max}, \mathsf{n} \}$, where:

- $\Delta_t : \mathbb{R}$ is the amount of time between two system updates,
- $\mathsf{d} : \mathbb{R}$ is a safety distance,
- $\mathsf{min} : \mathbb{R}$ is the minimum speed of trains on this track segment,
- $\mathsf{max} : \mathbb{R}$ is the maximum speed of trains on the segment, and
- $\mathsf{n} : \mathbb{Z}$ is the number of trains on the segment.

In this system we do not have system variables which change their valuation, i.e. $V = P$. The set of system functions Σ consists of a single function symbol pos of type $\mathbb{Z} \to \mathbb{R}$, modeling positions of trains on the track. That is, $\mathsf{pos}(i)$ denotes the position of train i on the track, if $0 \leq i \leq n-1$. Σ_P is empty.

We impose the following global constraints on the system parameters:

$$\begin{aligned}(\mathsf{Global}) \qquad & \Delta_t > 0 \ \wedge\ 0 \leq \mathsf{min} \ \wedge\ \mathsf{min} \leq \mathsf{max} \ \wedge\ \mathsf{n} > 0 \\ \wedge\quad & \mathsf{d} \geq \Delta_t \cdot \mathsf{max} - \Delta_t \cdot \mathsf{min}\end{aligned}$$

In the initial condition of the system, we define that the positions of the trains are ordered:

$$(\mathsf{Init}_V) \quad \forall\, i, j : \mathbb{Z}.\quad 0 \leq i < j < \mathsf{n} \ \rightarrow\ \mathsf{pos}(i) > \mathsf{pos}(j)$$

Note that the ordering on trains requires the train with the lowest number to have the greatest position.

Finally, we specify $(\mathsf{Update}) = \bigwedge_{i=1}^{4} \mathsf{F}_i$, where

$$\begin{aligned}
&(\mathsf{F}_1) \quad \forall\, i : \mathbb{Z}.\quad i = 0 \ \rightarrow\ \mathsf{pos}(i) + \Delta_t \cdot \mathsf{min} \leq \mathsf{pos}'(i) \leq \mathsf{pos}(i) + \Delta_t \cdot \mathsf{max} \\
&(\mathsf{F}_2) \quad \forall\, i : \mathbb{Z}.\quad 0 < i < \mathsf{n} \ \wedge\ \mathsf{pos}(i-1) \leq 0 \ \rightarrow\ \mathsf{pos}'(i) = \mathsf{pos}(i) \\
&(\mathsf{F}_3) \quad \forall\, i : \mathbb{Z}.\quad 0 < i < \mathsf{n} \ \wedge\ \mathsf{pos}(i-1) > 0 \ \wedge\ \mathsf{pos}(i-1) - \mathsf{pos}(i) < \mathsf{d} \\
&\qquad\qquad\ \rightarrow\ \mathsf{pos}'(i) = \mathsf{pos}(i) + \Delta_t \cdot \mathsf{min} \\
&(\mathsf{F}_4) \quad \forall\, i : \mathbb{Z}.\quad 0 < i < \mathsf{n} \ \wedge\ \mathsf{pos}(i-1) > 0 \ \wedge\ \mathsf{pos}(i-1) - \mathsf{pos}(i) \geq \mathsf{d} \\
&\qquad\qquad\ \rightarrow\ \mathsf{pos}(i) + \Delta_t \cdot \mathsf{min} \leq \mathsf{pos}'(i) \leq \mathsf{pos}(i) + \Delta_t \cdot \mathsf{max}
\end{aligned}$$

The rules determine the relation between the position $\mathsf{pos}(i)$ before the update and $\mathsf{pos}'(i)$ after the update, for $0 \leq i < \mathsf{n}$.

The first rule states that the behavior of train number 0 is free: it can always choose a speed between min and max. The next three rules specify the behavior

[2] In fact, the index theory for trains need not be the integers. We could also use an (acyclic) list structure, or any other theory that has a non-dense strict ordering and can be combined with real arithmetic. We use $\mathbb{Z}$ because it is directly supported in every state of the art SMT solver.

of the other trains: F_2 states that for any train i, if the preceding train $i-1$ has not passed position 0 yet, then i will not move at all. This models that the positions smaller than 0 are a starting interval, from which trains will only emerge one by one. F_3 states that a train must move at minimum speed if the distance to the preceding train is smaller than d, and F_4 states that it can choose any speed between min and max otherwise.

For the simple model, we define $T_1 = (V, \Sigma, (\mathsf{Init}_V) \cup (\mathsf{Global}), (\mathsf{Update}))$ with the components defined above.

5.4.2 Invariant Checking for the Simple Model

For our case study, collision freeness is one of the most important safety properties. Since we consider a single track and model trains without length, we can assume that we have a collision-free system if the initial strict ordering on the train positions is preserved. I.e., our safety condition is the same ordering on positions of trains as in our initial condition:

$$(\mathsf{Safe}) = (\mathsf{Init}_V).$$

In the following, we prove that (Safe) is a safety invariant of T_1. According to the approach for invariant checking introduced in Section 5.1.2, proving that (Safe) is an inductive safety invariant of our system $T_1 = (V, \Sigma, (\mathsf{Init}_V) \cup (\mathsf{Global}), (\mathsf{Update}))$ amounts to proving

(1) $\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}) \models (\mathsf{Safe})$

(2) $\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Init}_V) \models (\mathsf{Safe})$, and

(3) $\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}) \cup (\mathsf{Update}) \models (\mathsf{Safe}')$.

Verification conditions (1) and (2) are trivial. Verification condition (3) is a satisfiability problem in $\mathcal{T}_0$ extended with sets of clauses $(\mathsf{Global}), (\mathsf{Safe})$ and (Update), and we will use local reasoning to solve it, as shown in Section 5.2.

Locality

To prove $\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}) \cup (\mathsf{Update}) \models (\mathsf{Safe}')$, we consider three successive extensions of $\mathcal{T}_0$:

(1) the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Global})$. We call the extended theory $\mathcal{T}_1$.

(2) the extension $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup (\mathsf{Safe})$ with a function symbol pos of type $\mathbb{Z} \to \mathbb{R}$. We call the extended theory $\mathcal{T}_2$.

(3) the extension $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Update})$ with a function symbol pos' of type $\mathbb{Z} \to \mathbb{R}$. We call the extended theory $\mathcal{T}_3$.

In order to decide whether $\mathcal{T}_3 \models (\mathsf{Safe})$ using the solution from Section 5.2, we need to show that all three extensions are either universally or $\forall\exists$-reducing. For the extension $\mathcal{T}_0 \subseteq \mathcal{T}_0 \cup (\mathsf{Global})$ this is trivial, since (Global) is ground and in the signature of $\mathcal{T}_0$, except for new constant symbols. We show locality of the other two extensions:

Corollary 5.12. *The extension $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup (\mathsf{Safe})$ is universally reducing.*

Proof: Locality of the extension $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup (\mathsf{Safe})$ is a direct consequence of Corollary 3.14. Our base theory $\mathcal{T}_\mathbb{Z} \cup \mathcal{T}_\mathbb{R} \cup (\mathsf{Global})$ satisfies the conditions of Theorem 3.9 (in our case A is $\mathbb{Z}$, $<_A$ is $>_\mathbb{Z}$, B is $\mathbb{R}$ and $<_B$ is $<_\mathbb{R}$).[3] Constants a_1 and a_2 are instantiated to -1 and n, respectively (since $0 \leq_\mathbb{Z} i$ is equivalent to $-1 <_\mathbb{Z} i$). Thus, $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup (\mathsf{Safe})$ is a local extension.

Since $(\mathsf{Safe})[G]$ is finite and ground for every set of ground clauses G and the universal fragment of $\mathcal{T}_1$ is decidable, the extension $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup (\mathsf{Safe})$ is universally reducing. □

Corollary 5.13. *The extension $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Update})$ is universally reducing.*

Proof: Locality is a consequence of Theorem 3.19. The base theory for this extension is $\mathcal{T}_2 = \mathcal{T}_1 \cup (\mathsf{Safe})$, the partial order in this case is $\leq_\mathbb{R}$. Clearly, the left-hand sides of the update rules are conjunctions of literals in the signature of $\mathcal{T}_2$. For the right-hand sides, $\mathsf{pos}'(i) = \mathsf{pos}(i)$ in rule F_2 is equivalent to $\mathsf{pos}(i) \leq \mathsf{pos}'(i) \leq \mathsf{pos}(i)$, and similarly for F_3. Then, if we denote the right-hand side of a rule as $t_1 \leq \mathsf{pos}'(i) \leq t_2$, we have that t_1 and t_2 are terms in the signature of $\mathcal{T}_2$, and $t_1 \leq t_2$ for all rules (because of our restrictions on the system variables).

Finally, we need to ensure that the left-hand sides of the rules are mutually exclusive in $\mathcal{T}_2$. This is clear because the first literal of F_1 contradicts the first literal of all other rules, the second literal of F_2 contradicts the second literal of F_3 and F_4, and the third literal of F_3 contradicts the third literal of F_4. Thus, $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Update})$ is a local extension.

Since $(\mathsf{Update})[G]$ is finite and ground for every set of ground clauses G $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Update})$ is universally reducing. □

Hierarchical Reasoning

Since we know that all of the extensions above are universally reducing, we can use local reasoning in chains of extensions (see Section 2.6.1) to reduce the given satisfiability problem to an equisatisfiable problem in the base theory.

We want to check whether or not $\mathcal{T}_3 \models (\mathsf{Safe})$, or equivalently

$$\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}) \cup (\mathsf{Update}) \cup \neg(\mathsf{Safe}') \models \square,$$

since $\mathcal{T}_3 = \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}) \cup (\mathsf{Update})$.

By the locality of $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Update})$, we have

$$\begin{aligned} & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}) \cup (\mathsf{Update}) \cup \neg(\mathsf{Safe}') \models \square \\ \Leftrightarrow \quad & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}) \cup (\mathsf{Update})[\neg(\mathsf{Safe}')] \cup \neg(\mathsf{Safe}') \models \square \end{aligned}$$

Since the extension is universally reducing, the set $(\mathsf{Update})[\neg(\mathsf{Safe}')]$ consists only of ground clauses. For the second reduction, let $G = (\mathsf{Update})[\neg(\mathsf{Safe}')] \cup \neg(\mathsf{Safe}')$. By locality of $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup (\mathsf{Safe})$, we have

$$\begin{aligned} & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}) \cup G \models \square \\ \Leftrightarrow \quad & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe})[G] \cup G \models \square \end{aligned}$$

[3] For better readability, we omit the subscripts of the orderings in the following whenever the considered orderings are clear from the sorts of the compared terms.

The result of this reduction is again a set of ground clauses. Since (Global) is also ground, we can already solve this satisfiability problem with a Nelson-Oppen combination of decision procedures for $\mathcal{T}_{\mathbb{R}}, \mathcal{T}_{\mathbb{Z}}$ and free function symbols. However, since the arithmetic constraints are non-linear, most actual SMT implementations are not able to handle them, so we have to linearize them by restricting either Δ_{t} or both min and max to fixed values in order to use this approach.

An alternative is to make two further reduction steps, removing function symbols pos and pos′ by Ackermann's reduction:

$$\begin{aligned} & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe})[G] \cup G \models \square \\ \Leftrightarrow \quad & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe})[G]' \cup G' \cup D \models \square \\ \Leftrightarrow \quad & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe})[G]' \cup G' \cup N \models \square, \end{aligned}$$

where in the first step $(\mathsf{Safe})[G]' \cup G'$ results from replacing in $(\mathsf{Safe})[G] \cup G$ all extension terms $\mathsf{pos}(t_i), \mathsf{pos}'(u_i)$ with fresh constant symbols c_i, d_i and D contains for every fresh constant its definition $c_i = \mathsf{pos}(t_i)$, $d_i = \mathsf{pos}'(d_i)$. Now, function symbols pos and pos′ only appear in D.

In the second step we remove D, and have to add instances of the congruence axiom to preserve equisatisfiability: for every pair of equations $c_i = \mathsf{pos}(t_i)$, $c_j = \mathsf{pos}(t_j)$ in D, N contains a clause $t_i = t_j \rightarrow c_i = c_j$, and similarly for pos′-terms.

The resulting set of clauses is ground and does not contain any free function symbols. As a result, it can be handled by some decision procedures for $\mathcal{T}_{\mathbb{R}}$ that cannot handle the previous problem because of the additional free function symbols, but do support non-linear real arithmetic, like the implementation of real and integer arithmetic in the computer algebra system REDLOG/REDUCE [13].

Also, in the absence of function symbols we can use quantifier elimination in the combined theory of reals and integers to deduce constraints on the system parameters that guarantee safety. We can e.g. remove some or all of the global constraints (Global) and try to deduce a relation between the system parameters that is sufficient for safety of the system.

In the following, we show the reductions from above in (almost) full detail.

Reductions in Detail

For this simple model, we want to show in detail how the sets of formulas are obtained that can finally be handed to a prover of the base theory.

We want to show that $\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}) \cup (\mathsf{Update}) \cup \neg(\mathsf{Safe}') \models \square$ by reduction to a ground problem in the base theory $\mathcal{T}_0$, where

$$\neg(\mathsf{Safe}') \quad 0 \leq a < b < \mathsf{n} \wedge \mathsf{pos}'(a) \leq \mathsf{pos}'(b)$$

is the negation of (Safe), skolemized by introduction of fresh constants a and b. To this end, we first reduce the problem to a satisfiability problem over $\mathcal{T}_2$.

Reduction from $\mathcal{T}_3$ to $\mathcal{T}_2$. In order to construct the set $(\mathsf{Update})[\neg(\mathsf{Safe}')]$, we check $\mathsf{st}((\mathsf{Update}) \cup \neg(\mathsf{Safe}'))$ for extension terms. Our extension symbol in this reduction is pos′, and $\mathsf{st}((\mathsf{Update}) \cup \neg(\mathsf{Safe}'))$ contains only two extension terms, $\mathsf{pos}'(a)$ and $\mathsf{pos}'(b)$. This means that $(\mathsf{Update})[\neg(\mathsf{Safe}')]$ contains two

$$
\begin{array}{ll}
(\mathsf{F}_1)[\mathsf{a}] & a = 0 \;\rightarrow\; \mathsf{pos}(a) + \Delta_\mathsf{t} \cdot \mathsf{min} \leq \mathsf{pos}'(a) \leq \mathsf{pos}(a) + \Delta_\mathsf{t} \cdot \mathsf{max} \\
(\mathsf{F}_2)[\mathsf{a}] & 0 < a < \mathsf{n} \;\wedge\; \mathsf{pos}(a-1) \leq 0 \;\rightarrow\; \mathsf{pos}'(a) = pos(a) \\
(\mathsf{F}_3)[\mathsf{a}] & 0 < a < \mathsf{n} \;\wedge\; \mathsf{pos}(a-1) > 0 \;\wedge\; \mathsf{pos}(a-1) - \mathsf{pos}(a) < \mathsf{d} \\
 & \rightarrow\; \mathsf{pos}'(a) = \mathsf{pos}(a) + \Delta_\mathsf{t} \cdot \mathsf{min} \\
(\mathsf{F}_4)[\mathsf{a}] & 0 < a < \mathsf{n} \;\wedge\; \mathsf{pos}(a-1) > 0 \;\wedge\; \mathsf{pos}(a-1) - \mathsf{pos}(a) \geq \mathsf{d} \\
 & \rightarrow\; \mathsf{pos}(a) + \Delta_\mathsf{t} \cdot \mathsf{min} \leq \mathsf{pos}'(a) \leq \mathsf{pos}(a) + \Delta_\mathsf{t} \cdot \mathsf{max} \\
(\mathsf{F}_1)[\mathsf{b}] & b = 0 \;\rightarrow\; \mathsf{pos}(b) + \Delta_\mathsf{t} \cdot \mathsf{min} \leq \mathsf{pos}'(b) \leq \mathsf{pos}(b) + \Delta_\mathsf{t} \cdot \mathsf{max} \\
(\mathsf{F}_2)[\mathsf{b}] & 0 < a < \mathsf{n} \;\wedge\; \mathsf{pos}(a-1) \leq 0 \;\rightarrow\; \mathsf{pos}'(a) = pos(a) \\
(\mathsf{F}_3)[\mathsf{b}] & 0 < b < \mathsf{n} \;\wedge\; \mathsf{pos}(b-1) > 0 \;\wedge\; \mathsf{pos}(b-1) - \mathsf{pos}(b) < \mathsf{d} \\
 & \rightarrow\; \mathsf{pos}'(b) = \mathsf{pos}(b) + \Delta_\mathsf{t} \cdot \mathsf{min} \\
(\mathsf{F}_4)[\mathsf{b}] & 0 < b < \mathsf{n} \;\wedge\; \mathsf{pos}(b-1) > 0 \;\wedge\; \mathsf{pos}(b-1) - \mathsf{pos}(b) \geq \mathsf{d} \\
 & \rightarrow\; \mathsf{pos}(b) + \Delta_\mathsf{t} \cdot \mathsf{min} \leq \mathsf{pos}'(b) \leq \mathsf{pos}(b) + \Delta_\mathsf{t} \cdot \mathsf{max}
\end{array}
$$

Figure 5.1: Axiom instances in $(\mathsf{Update})[\neg(\mathsf{Safe}')]$

instances of every clause in (Update): one with i instantiated to a, the other with i instantiated to b. $(\mathsf{Update})[\neg(\mathsf{Safe}')]$ can be seen in Figure 5.1.

To decide satisfiability of $\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}) \cup (\mathsf{Update})[\neg(\mathsf{Safe}')] \cup \neg(\mathsf{Safe})$, we have to do another reduction with respect to the extension $\mathcal{T}_1 \subseteq \mathcal{T}_2$.

Reduction from $\mathcal{T}_2$ to $\mathcal{T}_1$. Again, we check our axioms and ground goal for ground extension terms. For this reduction, we consider the set of ground clauses $G = (\mathsf{Update})[\neg(\mathsf{Safe}')] \cup \neg(\mathsf{Safe}')$, there is one axiom in (Safe) and our extension symbol is pos. The set of ground extension terms contained in $\mathsf{st}(G)$ is $\{\ \mathsf{pos}(a), \mathsf{pos}(a-1), \mathsf{pos}(b), \mathsf{pos}(b-1)\ \}$. The axiom in (Safe) contains two variables i, j, and therefore has to be instantiated with every pair of terms appearing below pos, giving us 16 instances. $(\mathsf{Safe})[G]$ can be seen in Figure 5.2.

As mentioned before, the resulting set of ground clauses $(\mathsf{Global}) \cup (\mathsf{Safe})[G] \cup G$ can now be handed to an SMT solver or a similar tool that decides $\mathcal{T}_0$ with additional free function symbols, possibly after fixing values of either Δ_t or min and max.

Alternatively, we make another reduction to eliminate also the function symbols: first we purify the sets of instances and our ground goal, introducing new constants for extension terms. That is, $\neg(\mathsf{Safe}')$ is purified to

$$(P_1) \quad 0 \leq a < b < \mathsf{n} \;\wedge\; c_1 \leq c_2,$$

$(\mathsf{Update})[\neg(\mathsf{Safe}')]$ is purified to (P_2) and $(\mathsf{Safe})[(\mathsf{Update})[\neg(\mathsf{Safe}')] \cup \neg(\mathsf{Safe}')]$ is purified to (P_3), see Figures 5.3 and 5.4.

Purification produces a set of definitions (D), from which we compute a set N of instances of the congruence axiom for both pos′ and pos. Both (D) and (N) can be found in Figure 5.5.

Now, the original satisfiability problem is equivalent to satisfiability of the resulting set of clauses $(\mathsf{Global}) \cup (P_1) \cup (P_2) \cup (P_3) \cup (N)$ in $\mathcal{T}_0$.

Parametric verification. If we remove some of the global constraints on min, max and d, we find out that the system is unsafe, i.e. the problem becomes

$$
\begin{array}{ll}
(\mathsf{Safe})[a,a] & 0 \le a < a < \mathsf{n} \rightarrow \mathsf{pos}(a) > \mathsf{pos}(a) \\
(\mathsf{Safe})[a,a-1] & 0 \le a < a-1 < \mathsf{n} \rightarrow \mathsf{pos}(a) > \mathsf{pos}(a-1) \\
(\mathsf{Safe})[a,b] & 0 \le a < b < \mathsf{n} \rightarrow \mathsf{pos}(a) > \mathsf{pos}(b) \\
(\mathsf{Safe})[a,b-1] & 0 \le a < b-1 < \mathsf{n} \rightarrow \mathsf{pos}(a) > \mathsf{pos}(b-1) \\
(\mathsf{Safe})[a-1,a] & 0 \le a-1 < a < \mathsf{n} \rightarrow \mathsf{pos}(a-1) > \mathsf{pos}(a) \\
(\mathsf{Safe})[a-1,a-1] & 0 \le a-1 < a-1 < \mathsf{n} \rightarrow \mathsf{pos}(a-1) > \mathsf{pos}(a-1) \\
(\mathsf{Safe})[a-1,b] & 0 \le a-1 < b < \mathsf{n} \rightarrow \mathsf{pos}(a-1) > \mathsf{pos}(b) \\
(\mathsf{Safe})[a-1,b-1] & 0 \le a-1 < b-1 < \mathsf{n} \rightarrow \mathsf{pos}(a-1) > \mathsf{pos}(b-1) \\
(\mathsf{Safe})[b,a] & 0 \le b < a < \mathsf{n} \rightarrow \mathsf{pos}(b) > \mathsf{pos}(a) \\
(\mathsf{Safe})[b,a-1] & 0 \le b < a-1 < \mathsf{n} \rightarrow \mathsf{pos}(b) > \mathsf{pos}(a-1) \\
(\mathsf{Safe})[b,b] & 0 \le b < b < \mathsf{n} \rightarrow \mathsf{pos}(b) > \mathsf{pos}(b) \\
(\mathsf{Safe})[b,b-1] & 0 \le b < b-1 < \mathsf{n} \rightarrow \mathsf{pos}(b) > \mathsf{pos}(b-1) \\
(\mathsf{Safe})[b-1,a] & 0 \le b-1 < a < \mathsf{n} \rightarrow \mathsf{pos}(b-1) > \mathsf{pos}(a) \\
(\mathsf{Safe})[b-1,a-1] & 0 \le b-1 < a-1 < \mathsf{n} \rightarrow \mathsf{pos}(b-1) > \mathsf{pos}(a-1) \\
(\mathsf{Safe})[b-1,b] & 0 \le b-1 < b < \mathsf{n} \rightarrow \mathsf{pos}(b-1) > \mathsf{pos}(b) \\
(\mathsf{Safe})[b-1,b-1] & 0 \le b-1 < b-1 < \mathsf{n} \rightarrow \mathsf{pos}(b-1) > \mathsf{pos}(b-1)
\end{array}
$$

Figure 5.2: Axiom instances in $(\mathsf{Safe})[(\mathsf{Update})[\neg(\mathsf{Safe}')] \cup \neg(\mathsf{Safe}')]$

$$
(P_2) = \left\{ \begin{array}{l}
a = 0 \rightarrow d_1 + \Delta_\mathsf{t} \cdot \mathsf{min} \le c_1 \le d_1 + \Delta_\mathsf{t} \cdot \mathsf{max} \\
0 < a < \mathsf{n} \wedge d_3 > 0 \wedge d_3 - d_1 \ge \mathsf{d} \\
\rightarrow d_1 + \Delta_\mathsf{t} \cdot \mathsf{min} \le c_1 \le d_1 + \Delta_\mathsf{t} \cdot \mathsf{max} \\
0 < a < \mathsf{n} \wedge d_3 > 0 \wedge d_3 - d_1 < \mathsf{d} \\
\rightarrow c_1 = d_1 + \Delta_\mathsf{t} \cdot \mathsf{min} \\
0 < a < \mathsf{n} \wedge d_3 > 0 \wedge d_3 - d_1 \ge \mathsf{d} \\
\rightarrow d_1 + \Delta_\mathsf{t} \cdot \mathsf{min} \le c_1 \le d_1 + \Delta_\mathsf{t} \cdot \mathsf{max} \\
b = 0 \rightarrow d_2 + \Delta_\mathsf{t} \cdot \mathsf{min} \le c_2 \le d_2 + \Delta_\mathsf{t} \cdot \mathsf{max} \\
0 < b < \mathsf{n} \wedge d_4 > 0 \wedge d_4 - d_2 \ge \mathsf{d} \\
\rightarrow d_2 + \Delta_\mathsf{t} \cdot \mathsf{min} \le c_2 \le d_2 + \Delta_\mathsf{t} \cdot \mathsf{max} \\
0 < b < \mathsf{n} \wedge d_4 > 0 \wedge d_4 - d_2 < \mathsf{d} \\
\rightarrow c_2 = d_2 + \Delta_\mathsf{t} \cdot \mathsf{min} \\
0 < b < \mathsf{n} \wedge d_4 > 0 \wedge d_4 - d_2 \ge \mathsf{d} \\
\rightarrow d_2 + \Delta_\mathsf{t} \cdot \mathsf{min} \le c_2 \le d_2 + \Delta_\mathsf{t} \cdot \mathsf{max}
\end{array} \right\}
$$

Figure 5.3: Purification of $(\mathsf{Update})[\neg(\mathsf{Safe}')]$

$$(P_3) = \left\{ \begin{array}{l} 0 \leq a < a < \mathsf{n} \rightarrow d_1 > d_1 \\ 0 \leq a < a-1 < \mathsf{n} \rightarrow d_1 > d_3 \\ 0 \leq a < b < \mathsf{n} \rightarrow d_1 > d_2 \\ 0 \leq a < b-1 < \mathsf{n} \rightarrow d_1 > d_4 \\ 0 \leq a-1 < a < \mathsf{n} \rightarrow d_3 > d_1 \\ 0 \leq a-1 < a-1 < \mathsf{n} \rightarrow d_3 > d_3 \\ 0 \leq a-1 < b < \mathsf{n} \rightarrow d_3 > d_2 \\ 0 \leq a-1 < b-1 < \mathsf{n} \rightarrow d_3 > d_4 \\ 0 \leq b < a < \mathsf{n} \rightarrow d_2 > d_1 \\ 0 \leq b < a-1 < \mathsf{n} \rightarrow d_2 > d_3 \\ 0 \leq b < b < \mathsf{n} \rightarrow d_2 > d_2 \\ 0 \leq b < b-1 < \mathsf{n} \rightarrow d_2 > d_4 \\ 0 \leq b-1 < a < \mathsf{n} \rightarrow d_4 > d_1 \\ 0 \leq b-1 < a-1 < \mathsf{n} \rightarrow d_4 > d_3 \\ 0 \leq b-1 < b < \mathsf{n} \rightarrow d_4 > d_2 \\ 0 \leq b-1 < b-1 < \mathsf{n} \rightarrow d_4 > d_4 \end{array} \right\}$$

Figure 5.4: Purification of $(\mathsf{Safe})[(\mathsf{Update})[\neg(\mathsf{Safe}')] \cup \neg(\mathsf{Safe}')]$

$$(D) = \left\{ \begin{array}{ll} c_1 = \mathsf{pos}'(a) & c_2 = \mathsf{pos}'(b) \\ d_1 = \mathsf{pos}(a) & d_2 = \mathsf{pos}(b) \\ d_3 = \mathsf{pos}(a-1) & d_4 = \mathsf{pos}(b-1) \end{array} \right\}$$

$$(N) = \left\{ \begin{array}{ll} a = b \rightarrow c_1 = c_2 & \\ a = b \rightarrow d_1 = d_2 & a = a-1 \rightarrow d_1 = d_3 \\ a = b-1 \rightarrow d_1 = d_4 & b = a-1 \rightarrow d_2 = d_3 \\ b = b-1 \rightarrow d_2 = d_4 & a-1 = b-1 \rightarrow d_3 = d_4 \end{array} \right\}$$

Figure 5.5: Sets of definitions (D) and congruence axioms (N) from purification

satisfiable.[4] This gives raise to an interesting problem in a different kind of parametric verification.

The method for hierarchical reasoning described above allows us to reduce the problem of checking whether system properties such as collision freeness are inductive invariants to deciding satisfiability of corresponding constraints in $\mathcal{T}_0$.

For the given problem, we can achieve even more: if we remove function symbols completely, we are in the combined theory of $\mathcal{T}_\mathbb{R} \cup \mathcal{T}_\mathbb{Z}$. For this theory, quantifier elimination procedures exist. That is, we can transform the satisfiability problem above by replacing constants with quantified variables, and then eliminate these variables. The result is a problem that is equisatisfiable to the original one, but only contains a subset of the original system variables. We can consider the model without the constraints on $\Delta_\mathsf{t}, \mathsf{min}$ and max, and then eliminate all variables except these three. Thus, we end up with an equisatisfiable formula from which we can find out how to choose (constraints on) $\Delta_\mathsf{t}, \mathsf{min}$ and max such that the formula is not satisfiable.

5.4.3 Bounded Model Checking of the Simple Model

First, consider the TCS T_1 as introduced in Section 5.4.1. Let $(\mathsf{Init}_0), (\mathsf{Update}_i)$ and (Safe_k) be modifications of the original formulas, as defined in Section 5.1.3.

Then, we can show absence of paths of length k that lead to an unsafe state by proving

$$\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Init}_0) \cup \textstyle\bigcup_{i=1}^{k} (\mathsf{Update}_i) \cup \neg(\mathsf{Safe}_k) \models \Box.$$

Since we have proved that (Safe) is an inductive invariant of T_1, we are able to prove absence of error paths for any k.

To obtain a more interesting BMC problem, we now consider a slightly different system than before: suppose the global constraints are

$$(\mathsf{Global}_{BMC}) \quad \mathsf{d} = 100 \ \wedge\ \mathsf{min} = 0 \ \wedge\ \mathsf{max} = 100 \ \wedge\ \Delta_\mathsf{t} = 1,$$

and our initial condition states that between two trains we have at least distance 1000, i.e.

$$(\mathsf{Init}_{BMC}) \quad \forall\, i, j : \mathbb{Z}.\quad 1 \leq i < j \leq \mathsf{n} \ \rightarrow\ \mathsf{pos}(i) - \mathsf{pos}(j) > (j - i) \cdot 1000.$$

The extension $\mathcal{T}_0 \cup (\mathsf{Global}_{BMC}) \subseteq \mathcal{T}_0 \cup (\mathsf{Global}_{BMC}) \cup (\mathsf{Init}_{BMC})$ is local by Corollary 3.34, and universally reducing because $(\mathsf{Init}_{BMC})[G]$ is ground for any set of ground clauses G. Furthermore, the extension $\mathcal{T}_0 \cup (\mathsf{Global}_{BMC}) \cup (\mathsf{Init}_{BMC}) \cup \bigcup_{i=1}^{k-1} (\mathsf{Update}_i) \subseteq \mathcal{T}_0 \cup (\mathsf{Global}_{BMC}) \cup (\mathsf{Init}_{BMC}) \cup \bigcup_{i=1}^{k} (\mathsf{Update}_i)$ is universally reducing for every $k \geq 1$; the proof of Corollary 5.13 can easily be modified to show this.

Now, consider the following safety condition:

$$(\mathsf{Safe}_{BMC}) \quad \forall\, i : \mathbb{Z}.\quad 1 \leq i < \mathsf{n} \ \rightarrow\ \mathsf{pos}(i) > \mathsf{pos}(i + 1).$$

Then, to prove safety of the system for k steps, we have to check whether

$$\mathcal{T}_0 \cup (\mathsf{Global}_{BMC}) \cup (\mathsf{Init}_{BMC}) \cup \textstyle\bigcup_{i=1}^{k} (\mathsf{Update}_i) \cup \neg(\mathsf{Safe}_{BMC}) \models \Box.$$

[4] If we remove the constraints on Δ_t or n, we simply do not have a reasonable model of the case study anymore.

Since all the extensions above are universally reducing, we can use the local reasoning approach to reduce this problem (for any k) to a ground satisfiability problem in $\mathcal{T}_0$.

It turns out that the system is safe for $k \leq 9$, but unsafe states are reachable if we have $k \geq 10$. The reason for this is that with the given values for max and min, the distance between trains can shrink by 100 with every step. The system is unsafe because d is not large enough to prevent collisions: if two trains have distance 100 before an update, the first train will be allowed to stop (i.e. go at speed min), while the following train is allowed to go at maximum speed, having them end up in the same position after the update.

Note that by fixing $\Delta_\mathsf{t} = 1$ and using the modified safety condition, we remain in an effectively linear fragment of arithmetic, even though (Init_{BMC}) contains a multiplication of terms containing variables.[5] This allows us to use SMT solvers for the base theory that do not support non-linear constraints. For invariant checking this would not be an option, since the modified safety condition (Safe_{BMC}) does not satisfy a locality property. However, as we only use the negation of (Safe_{BMC}) in the BMC problem, this is not a problem in this case.

5.4.4 Automatic Verification

We have seen that for a simple model of the ETCS case study, invariant checking and bounded model checking problems are decidable. The corresponding verification conditions can be expressed as first-order formulas modulo the base theories $\mathcal{T}_\mathbb{R} \cup \mathcal{T}_\mathbb{Z}$.

Automation by reasoning in chains of local extensions. The main problem for automatic verification is that the verification conditions contain universally quantified formulas, which have to be checked for satisfiability modulo non-trivial theories. We solve this problem by using locality to efficiently eliminate quantifiers by instantiation, resulting in a ground satisfiability problem in a decidable fragment of the base theory. To this end, we identify corresponding chains of theory extensions$\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \cdots \subseteq \mathcal{T}_k$, such that $\mathcal{T}_i \subseteq \mathcal{T}_{i+1}$ is universally reducing for all i. Every $\mathcal{T}_{i+1}$ extends $\mathcal{T}_i$ with a set of (universally quantified) clauses. Locality of the extensions allows us to reduce the first-order problems to ground problems, which can be efficiently checked by existing methods.

Automatic verification with iLoRe. We can check these first-order satisfiability problems automatically, using our implementation of the hierarchical reasoning approach. If we want a state-of-the-art SMT solver to check the ground problem that results from instantiation of the extension axioms, we need to make sure that we do not have non-linear arithmetic constraints in the instances given to the tool. In the simple model, it is enough to give a fixed valuation to the length of the time interval Δ_t. Otherwise, we may give the non-linear problem to provers that can handle non-linear problems, like REDLOG [13].

Note that the update rules and the invariant to be proved have a very specific structure: in any clause, there is exactly one literal that contains all exten-

[5] If variable i in the negation of (Safe_{BMC}) is replaced by a constant a, then variables j and i in axiom (Init_{BMC}) will always be instantiated with a term of the form $a(+1)^*(-1)^*$, i.e. a after incrementing and decrementing it a number of times. As the two resulting terms are subtracted, a will disappear, leaving a fixed-value multiplication.

sion terms. Because of this, the incremental approach LIG^* produces exactly the same instances as the eager instantiation. In contrast to this, LIG^*_{uc} can give some benefit over the eager approach: manual tests with the SMT solver Yices [14] have shown that safety of the system can be proved with only 10 out of the 28 instances of (Update) and (Safe) that the eager method generates. However, we were not able to check behavior of the actual implementation of LIG^*_{uc} because it currently does not support all features needed to treat the given problem.

Verification without locality arguments. If we try to solve the non-ground invariant checking problem directly with an SMT solver (i.e. without using locality of the axioms), it can only give us a conclusive answer if the problem is unsatisfiable. For satisfiable problems, e.g. if we specify an unsafe system by leaving out the global constraints (Global), SMT solvers run forever or return `unknown`, while a local reasoning-based approach returns a model of the given set of clauses. This is equivalent to a counterexample to safety and can therefore be used to find out why the system is unsafe, and refine it to exclude this behavior.

For the BMC problem, a test with Yices shows that even for $k = 1$ the problem cannot be solved without local reasoning: if the axioms are not instantiated before giving it to Yices, the prover states that this is a non-linear problem and refuses to even start instantiating the axioms.

Next, we show that the local extensions described in Section 2.5 and Chapter 3 allow us to express models and safety properties that are more complex than the one we considered until now. We explore some extensions of this model, and show that the resulting problems still fall into a class that can be decided with local reasoning.

5.5 Extensions of the Simple Model

In this Section, we present a number of extensions of the simple model introduced in 5.4. In Section 5.5.1 we will show that we can model a rail track where trains can leave at one end of the track, and new trains can enter at the other end. That is, not only do we have a parametric number of trains, but the number of trains can change during execution, bounded by a system parameter. In Section 5.5.2, we no longer consider trains as points on a line, but give them a length. This leads to a different axiomatization of collision-freeness. Finally, in Section 5.5.3 we consider a model that not only has the standard mode of operation, but can also handle emergencies at any point of the track. These emergencies appear nondeterministically, and therefore safety of the system depends on some additional properties like a bounded braking distance of trains.

5.5.1 Incoming and Leaving Trains

In order to allow incoming and leaving trains, we introduce a modified model $T_2 = (V_2, \Sigma, (\mathsf{Init}_2) \cup (\mathsf{Global}_2), (\mathsf{Update}_2))$. In this model, we need a measure for the number of trains on the track, and we want to have a bound on this number. Furthermore, this specification has to be compatible with the kind of

update rules that we identified as local extensions. This can be achieved in the following way:

Since the number of trains is not fixed anymore, we do not need n anymore. Instead, we have a new system parameter $\mathsf{maxTrains} : \mathbb{Z}$ and system variables $\mathsf{first} : \mathbb{Z}$ and $\mathsf{last} : \mathbb{Z}$, which can change during execution and at any time give the number of the first and last train on the track.

Since we only want trains to leave at the ends of the track, this is sufficient to model incoming and leaving trains: first will always be the smallest of all numbers that are assigned to trains on the track, and the train with the smallest number should have the greatest position. If a train wants to leave the track, it has to be this one. Thus, we can model a leaving train by simply incrementing first, implicitly stating that the train that was first before is no longer on the track. Similarly, last is always equal to the greatest of all numbers that are assigned to trains, and the train with that number should have the smallest position. A train can only enter the track behind this one, and therefore should have a higher number. Thus, we model incoming trains by incrementing last, implicitly stating that this number now also refers to a train on the track. In this case, we also have to assign a suitable value to $\mathsf{pos}(\mathsf{last})$ for this new train.

The current number of trains on the track is then always equal to $\mathsf{last} - \mathsf{first} + 1$, which should be bounded by $\mathsf{maxTrains}$.

Then, for this extension T_2 of the simple model, our set of system parameters is $P_2 = \{\ \Delta_\mathsf{t}, \mathsf{d}, \mathsf{min}, \mathsf{max}, \mathsf{maxTrains}\ \}$. We use modified global constraints on the system parameters:

$$\begin{aligned}(\mathsf{Global}_2) \qquad & \Delta_\mathsf{t} > 0 \ \wedge\ 0 \le \mathsf{min} \ \wedge\ \mathsf{min} \le \mathsf{max} \ \wedge\ \mathsf{maxTrains} > 0 \\ \wedge\ & \mathsf{d} \ge \Delta_\mathsf{t} \cdot \mathsf{max} - \Delta_\mathsf{t} \cdot \mathsf{min}\end{aligned}$$

The set of system variables is $V_2 = P \cup \{\ \mathsf{first}, \mathsf{last}\ \}$ and the set of system functions is the same as before, but we are now interested in values of $\mathsf{pos}(i)$ for $\mathsf{first} \le i \le \mathsf{last}$, instead of $0 \le i < \mathsf{n}$.

Accordingly, the initial condition (Init_2) is

$$\begin{aligned}(\mathsf{Init}_2) \qquad & \forall\, i, j : \mathbb{Z}.\quad \mathsf{first} \le i < j \le \mathsf{last} \ \rightarrow\ \mathsf{pos}(i) > \mathsf{pos}(j) \\ \wedge\ & \mathsf{first} - \mathsf{last} + 1 \le \mathsf{maxTrains}\end{aligned}$$

For the extended model, we have $(\mathsf{Update}_2) = \bigwedge_{i=1}^{9} \mathsf{V}_i$, where

$$\begin{aligned}
&(\mathsf{V}_1) \quad \forall\, i : \mathbb{Z}.\quad i = \mathsf{first} \ \rightarrow\ \mathsf{pos}(i) + \Delta_\mathsf{t} \cdot \mathsf{min} \le \mathsf{pos}'(i) \le \mathsf{pos}(i) + \Delta_\mathsf{t} \cdot \mathsf{max} \\
&(\mathsf{V}_2) \quad \forall\, i : \mathbb{Z}.\quad \mathsf{first} < i < \mathsf{last} \ \wedge\ \mathsf{pos}(i-1) \le 0 \ \rightarrow\ \mathsf{pos}'(i) = \mathsf{pos}(i) \\
&(\mathsf{V}_3) \quad \forall\, i : \mathbb{Z}.\quad \mathsf{first} < i < \mathsf{last} \ \wedge\ \mathsf{pos}(i-1) > 0 \ \wedge\ \mathsf{pos}(i-1) - \mathsf{pos}(i) < \mathsf{d} \\
&\qquad\qquad\quad \rightarrow\ \mathsf{pos}'(i) = \mathsf{pos}(i) + \Delta_\mathsf{t} \cdot \mathsf{min} \\
&(\mathsf{V}_4) \quad \forall\, i : \mathbb{Z}.\quad \mathsf{first} < i < \mathsf{last} \ \wedge\ \mathsf{pos}(i-1) > 0 \ \wedge\ \mathsf{pos}(i-1) - \mathsf{pos}(i) \ge \mathsf{d} \\
&\qquad\qquad\quad \rightarrow\ \mathsf{pos}(i) + \Delta_\mathsf{t} \cdot \mathsf{min} \le \mathsf{pos}'(i) \le \mathsf{pos}(i) + \Delta_\mathsf{t} \cdot \mathsf{max} \\
&(\mathsf{V}_5) \quad \mathsf{last} - \mathsf{first} + 1 < \mathsf{maxTrains} \ \rightarrow\ \mathsf{last}' = \mathsf{last} \vee \mathsf{last}' = \mathsf{last} + 1 \\
&(\mathsf{V}_6) \quad \mathsf{last} - \mathsf{first} + 1 = \mathsf{maxTrains} \ \rightarrow\ \mathsf{last}' = \mathsf{last} \\
&(\mathsf{V}_7) \quad \mathsf{last} - \mathsf{first} + 1 > 0 \ \rightarrow\ \mathsf{first}' = \mathsf{first} \vee \mathsf{first}' = \mathsf{first} + 1 \\
&(\mathsf{V}_8) \quad \mathsf{last} - \mathsf{first} + 1 = 0 \ \rightarrow\ \mathsf{first}' = \mathsf{first} \\
&(\mathsf{V}_9) \quad \mathsf{last}' = \mathsf{last} + 1 \ \rightarrow\ \mathsf{pos}'(\mathsf{last}') < \mathsf{pos}'(\mathsf{last})
\end{aligned}$$

Clauses $\mathsf{V}_1 - \mathsf{V}_4$ are similar to $\mathsf{F}_1 - \mathsf{F}_4$, except that the fixed bounds are replaced by the constants first and last. V_5 states that if the number of trains is less than $\mathsf{maxTrains}$, then a new train can enter. V_6 says that no train may enter if $\mathsf{maxTrains}$ is already reached. V_7 and V_8 are similar conditions for leaving trains. Finally, V_9 states that if a train enters, its position must be behind the train that was last before the transition.

Verification of the Extended Model

As before, we want to use invariant checking to prove the absence of collisions in our model. Like the initial condition, our safety property will not have to hold between 0 and $n-1$, but between first and last:

$$(\mathsf{Safe}_2) \quad \forall\, i, j : \mathbb{Z}.\quad \mathsf{first} \leq i < j \leq \mathsf{last} \;\rightarrow\; \mathsf{pos}(i) > \mathsf{pos}(j).$$

The verification approach is the same as before. To show that (Safe_2') is an inductive safety invariant of T_2, we need to prove

(1) $\mathcal{T}_0 \cup (\mathsf{Global}_2) \cup (\mathsf{Safe}_2) \models (\mathsf{Safe}_2)$

(2) $\mathcal{T}_0 \cup (\mathsf{Global}_2) \cup (\mathsf{Init}_2) \models (\mathsf{Safe}_2)$, and

(3) $\mathcal{T}_0 \cup (\mathsf{Global}_2) \cup (\mathsf{Safe}_2) \cup (\mathsf{Update}_2) \models (\mathsf{Safe}_2')$.

Again, verification conditions (1) and (2) are trivial (since (Init_2) contains (Safe_2)). We have to prove verification condition (3).

Locality. As we want to use local reasoning, we need to show that for $\mathcal{T}_1 = \mathcal{T}_0 \cup (\mathsf{Global}_2)$, $\mathcal{T}_2 = \mathcal{T}_1 \cup (\mathsf{Safe}_2)$ and $\mathcal{T}_3 = \mathcal{T}_2 \cup (\mathsf{Update}_2)$, every extension in the chain of extensions $\mathcal{T}_0 \subseteq \mathcal{T}_1 \subseteq \mathcal{T}_2 \subseteq \mathcal{T}_3$ is universally reducing. Again, this is trivial for $\mathcal{T}_0 \subseteq \mathcal{T}_1$ since (Global_2) is ground and in the signature of $\mathcal{T}_0$ except for additional constant symbols.

The following is an easy corollary of Corollary 3.14:

Corollary 5.14. *The extension $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup (\mathsf{Safe}_2)$ is universally reducing.*

Proof: Similar to Corollary 5.12, except that (Safe_2) contains different constants than (Safe). □

Locality of the second extension is a corollary of Theorems 3.19 and 3.1, and we can easily show that the extension is also universally reducing:

Corollary 5.15. *The extension $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Update}_2)$ is universally reducing.*

Proof: Similar to Corollary 5.13, except that (Update_2) uses different constants than (Update), and contains an additional set of ground clauses, which do not destroy locality by Theorem 3.1. □

Hierarchical Reasoning. Locality of the two extensions then guarantees that the following is an equisatisfiable reduction:

$$\begin{aligned}
& \mathcal{T}_0 \cup (\mathsf{Global}_2) \cup (\mathsf{Safe}_2) \cup (\mathsf{Update}_2) \cup \neg(\mathsf{Safe}_2') \models \square \\
\Leftrightarrow\quad & \mathcal{T}_0 \cup (\mathsf{Global}_2) \cup (\mathsf{Safe}_2) \cup \underbrace{(\mathsf{Update}_2)[\neg(\mathsf{Safe}_2')] \cup \neg(\mathsf{Safe}_2')}_{G} \models \square \\
\Leftrightarrow\quad & \mathcal{T}_0 \cup (\mathsf{Global}_2) \cup (\mathsf{Safe}_2)[G] \cup G \models \square.
\end{aligned}$$

Since the extensions are universally reducing, the last line is a ground satisfiability problem in $\mathcal{T}_0$ with additional free function symbols, which can be solved by standard SMT solvers. Again, we can also use Ackermann's reduction to remove the extension symbol and reduce the problem to an equisatisfiable problem in $\mathcal{T}_0$.

Regarding the incremental approaches LIG^* and LIG^*_{uc}, we have the same situation as in the simple model: the specific structure of the invariant and update rules lets LIG^* produce the same instances as the eager approach, while LIG^*_{uc} will generate about half of the instances, depending on the unsatisfiable cores obtained from the SMT solver.

5.5.2 A more precise axiomatization of collision-freeness

As mentioned before, we can make the model more precise by not considering trains as points on a line, but instead considering trains of a specific length. An easy way to do this is to add a system parameter $\mathsf{LengthTrain}$, which specifies the standard (or maximal) length of any train that is on the track (or can enter the track). Assume that $\mathsf{pos}(i)$ then specifies the rear-end of train i, i.e. every train i occupies the track in the interval $[\mathsf{pos}(i), \mathsf{pos}(i) + \mathsf{LengthTrain}]$. Obviously, a monotonicity axiom like (Safe_1) or (Safe_2) is no longer sufficient to prove collision freeness of such a system.

In the following, we consider systems that behave like T_1 or T_2 defined before, but have an additional system parameter $\mathsf{LengthTrain}$ and different initial conditions. For the system based on T_1 we define

$$(\mathsf{Init}_3) \quad \forall i, j : \mathbb{Z}. \quad 1 \leq i < j \leq \mathsf{n} \\ \rightarrow \mathsf{pos}(i) - \mathsf{pos}(j) > (j - i) \cdot \mathsf{LengthTrain},$$

and for the system based on T_2 we define

$$(\mathsf{Init}_4) \qquad \forall i, j : \mathbb{Z}. \quad \mathsf{first} \leq i < j \leq \mathsf{last} \\ \rightarrow \mathsf{pos}(i) - \mathsf{pos}(j) > (j - i) \cdot \mathsf{LengthTrain} \\ \wedge \qquad \mathsf{first} - \mathsf{last} + 1 \leq \mathsf{maxTrains}.$$

Let $T_3 = (V \cup \{\mathsf{LengthTrain}\}, \Sigma, (\mathsf{Init}_3) \cup (\mathsf{Global}), (\mathsf{Update}))$ be the modification of T_1 in this way, and $T_4 = (V_2 \cup \{\mathsf{LengthTrain}\}, \Sigma, (\mathsf{Init}_4) \cup (\mathsf{Global}_2), (\mathsf{Update}_2))$ the corresponding modification of T_2. Note that we have a multiplication symbol $\cdot$ of type $\mathbb{Z} \times \mathbb{R} \rightarrow \mathbb{R}$ in these initial properties, which we would usually assume to be a function in the base theory. We will see that this additional requirement on the base theory is not needed if we can show the resulting theory extensions to be universally reducing and we use a suitable representation of the invariant.

Verification of the extended models

We want to prove safety of these systems, where $(\mathsf{Safe}_3) = (\mathsf{Init}_3)$ and

$$(\mathsf{Safe}_4) \quad \forall i, j : \mathbb{Z}. \quad \mathsf{first} \leq i < j \leq \mathsf{last} \\ \rightarrow \mathsf{pos}(i) - \mathsf{pos}(j) > (j - i) \cdot \mathsf{LengthTrain}.$$

Since (Safe_3) is again included in (Init_3), proving that (Safe_3) is an invariant of T_3 amounts to proving

$$\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}_3) \cup (\mathsf{Update}) \cup \neg(\mathsf{Safe}'_3) \models \Box.$$

Similarly, proving that (Safe_4) is an invariant of T_4 amounts to proving

$$\mathcal{T}_0 \cup (\mathsf{Global})(\mathsf{Safe}_4) \cup (\mathsf{Update}_2) \cup \neg(\mathsf{Safe}'_4) \models \Box.$$

Locality. As before, we want to show that the resulting chains of extensions are universally reducing. First, consider the extensions for proving safety of T_3. Let $\mathcal{T}_1 = \mathcal{T}_0 \cup (\mathsf{Global})$ (the trivial extension of $\mathcal{T}_0$). Then the extension of $\mathcal{T}_1$ with (Safe_3) a local theory extension by Corollary 3.34 and universally reducing because all variables appear below extension symbols:

Corollary 5.16. *The extension $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup (\mathsf{Safe}_3)$ is universally reducing.*

Let $\mathcal{T}_2 = \mathcal{T}_1 \cup (\mathsf{Safe}_3)$. Locality of the extension $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Update})$ follows from Theorems 3.19 and 3.1, and we can easily show the extension to be universally reducing:

Corollary 5.17. *The extension $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Update})$ is universally reducing.*

Proof: Similar to Corollary 5.13, except that pos is defined differently in the base theory $\mathcal{T}_1$. This does not affect locality, since the left-hand sides of the implications are still mutually exclusive, and we still have $\mathsf{pos}(i) + \Delta_{\mathsf{t}} \cdot \mathsf{min} \leq \mathsf{pos}(i) + \Delta_{\mathsf{t}} \cdot \mathsf{max}$ for all i. □

Now consider the extensions needed for proving safety of T_4, and let $\mathcal{T}'_1 = \mathcal{T}_0 \cup (\mathsf{Global}_2)$. Then the extension of $\mathcal{T}'_1$ with (Safe_4) is a local theory extension by Corollary 3.34 and universally reducing because all variables appear below extension symbols:

Corollary 5.18. *The extension $\mathcal{T}'_1 \subseteq \mathcal{T}'_1 \cup (\mathsf{Safe}_4)$ is a local theory extension.*

Now let $\mathcal{T}'_2 = \mathcal{T}'_1 \cup (\mathsf{Safe}_4)$. Then, $\mathcal{T}'_2 \subseteq \mathcal{T}'_2 \cup (\mathsf{Update}_2)$ is a universally reducing theory extension:

Corollary 5.19. *The extension $\mathcal{T}'_1 \subseteq \mathcal{T}'_1 \cup (\mathsf{Update}_2)$ is universally reducing.*

Proof: Similar to Corollary 5.15, except that pos is defined differently in the base theory $\mathcal{T}'_1$. As in Corollary 5.15, this does not affect locality. □

Hierarchical Reasoning. Thus, we can use local reasoning to reduce our satisfiability problems to equisatisfiable sets of ground clauses in $\mathcal{T}_0$. For the system T_3 we get

$$\begin{aligned}
& \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}_3) \cup (\mathsf{Update}_1) \cup \neg(\mathsf{Safe}'_3) \models \Box \\
\Leftrightarrow\ & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}_3) \cup \underbrace{(\mathsf{Update}_1)[\neg(\mathsf{Safe}'_3)] \cup \neg(\mathsf{Safe}'_3)}_{G} \models \Box \\
\Leftrightarrow\ & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{Safe}_3)[G] \cup G \models \Box,
\end{aligned}$$

and for T_4

$$\begin{aligned}
& \mathcal{T}_0 \cup (\mathsf{Global}_2) \cup (\mathsf{Safe}_4) \cup (\mathsf{Update}_2) \cup \neg(\mathsf{Safe}'_4) \models \Box \\
\Leftrightarrow\ & \mathcal{T}_0 \cup (\mathsf{Global}_2) \cup (\mathsf{Safe}_4) \cup \underbrace{(\mathsf{Update}_2)[\neg(\mathsf{Safe}'_4)] \cup \neg(\mathsf{Safe}'_4)}_{G} \models \Box \\
\Leftrightarrow\ & \mathcal{T}_0 \cup (\mathsf{Global}_2) \cup (\mathsf{Safe}_4)[G] \cup G \models \Box.
\end{aligned}$$

Note that the resulting ground goals still contain multiplications between integer and real constants, which means we are not in the linear fragment of arithmetic.

We can solve this problem by not using the skolemization of the negated invariants in the reduction, but equisatisfiable formulas with only one constant. E.g., the skolemized negation of (Safe'_3) is

$$1 \leq a < b \leq \mathsf{n} \ \wedge \ \mathsf{pos}'(a) - \mathsf{pos}'(b) \leq (b - a) \cdot \mathsf{LengthTrain}.$$

This is equisatisfiable to

$$1 \leq a < \mathsf{n} \ \wedge \ \mathsf{pos}'(a) - \mathsf{pos}'(a + 1) \leq \mathsf{LengthTrain}.$$

Using the second version as the initial set of ground clauses in the reduction, variables i and j in (Safe_3) will only be instantiated with a and $a + 1$, which in all possible combinations results in constant values $-1, 0$ or 1 as left-hand side of the multiplication. Now, standard SMT solvers can be used to prove unsatisfiability of the resulting set of ground clauses.

Behavior of LIG^* and LIG^*_{uc} on these examples is similar to what we described for T_1 and T_2.

5.5.3 Handling Emergency Messages

In this section, we want to increase complexity of our system in another dimension: in addition to the normal mode of operation, we allow non-deterministic emergencies, which trigger a special behavior of the system. This will be expressed as a more complicated transition relation, in fact consisting of two different transitions: the usual position update and possible emergencies.

Since in the event of an emergency we cannot expect the system to behave in the desired way, we add a measure for the braking distance of a train at a given speed. For emergency situations, we want every train to be able to come to a standstill without colliding with other trains, even if those stop immediately, e.g. because they run into an obstacle on the track.

To illustrate this example, we use an extension of our first model from Section 5.4. After introducing the new extension, we show how it can be combined with the other extensions introduced previously.

TCS of the Model with Emergencies

We introduce a system $T_5 = (V_5, \Sigma_5, (\mathsf{Init}_5), (\mathsf{Update}_5))$ with components as defined in the following. Let $P_5 = \{ \ \Delta_{\mathsf{t}}, \mathsf{d}, \mathsf{min}, \mathsf{max}, \mathsf{n}, \mathsf{maxDec} \ \}$, where maxDec represents the deceleration of a train in case of an emergency. It is used to model the behavior of a train in case of an emergency and to compute the braking distance of a train at a given speed.

Furthermore, let $V_5 = P_5 \cup \{ \ \mathsf{emergTrain}, \mathsf{newEmergTrain} \ \}$, where

- $\mathsf{emergTrain}$ will always store the number of the foremost train which encountered an emergency. All subsequent trains, i.e. those with a number greater than $\mathsf{emergTrain}$, will have to stop. Initially, $\mathsf{emergTrain}$ is set to a value greater than n, which means that we don't have an emergency on the track.

- $\mathsf{newEmergTrain}$ will be used as a nondeterministic input, modeling that any train on the track could encounter an emergency.

In addition to pos, let Σ_5 contain the following function symbols:

- $\mathsf{brakingDist}$ of type $\mathbb{R} \to \mathbb{R}$ will give the braking distance of a train moving with a given speed, and

- speed of type $\mathbb{Z} \to \mathbb{R}$ will give the current speed of a train.

The valuation of $\mathsf{brakingDist}$ will not change during execution, i.e. $\Sigma_{P,5} = \{\mathsf{brakingDist}\}$.
We define the global constraints on the system parameters as

$$(\mathsf{Global}_5) \quad \begin{aligned} &\Delta_\mathsf{t} > 0 \ \wedge\ 0 \leq \mathsf{min} \ \wedge\ \mathsf{min} \leq \mathsf{max} \ \wedge\ \mathsf{n} > 0 \\ &\wedge\ \mathsf{maxDec} > 0 \ \wedge\ \mathsf{d} \geq \Delta_\mathsf{t} \cdot \mathsf{max} - \Delta_\mathsf{t} \cdot \mathsf{min}, \end{aligned}$$

and define $\mathsf{brakingDist}$ by

$$(\mathsf{BDist}) \quad \begin{aligned} &\forall\, s{:}\mathbb{R}. && \mathsf{brakingDist}(s) \geq \tfrac{s^2}{2 \cdot \mathsf{maxDec}} \\ &\forall\, s_1, s_2{:}\mathbb{R}. && s_1 < s_2 \ \to\ \mathsf{brakingDist}(s_1) < \mathsf{brakingDist}(s_2). \end{aligned}$$

Values of the function speed will change during an execution, and we assume that initially we have

$$(\mathsf{Init}_\mathsf{s}) \quad \forall\, i{:}\mathbb{Z}. \quad \mathsf{min} \leq \mathsf{speed}(i) \leq \mathsf{max},$$

which is the first part of our initial condition.

As we mentioned, we not only have to ensure that trains do not collide in their normal mode of operation, but also that all trains are able to come to a standstill in case of an emergency. As a second part of our initial condition, we want to require that

$$\forall\, i{:}\mathbb{Z}. \quad 1 \leq i \leq \mathsf{n} \ \to\ \mathsf{pos}(i) < \mathsf{pos}(i-1) - \mathsf{brakingDist}(\mathsf{speed}(i)).$$

This formula does not satisfy a locality property, but we can find an equivalent formula that fits into the fragment defined in Corollary 3.37 (as an extension of a theory with suitably defined functions $\mathsf{brakingDist}$, speed and $\sum$):

$$\begin{aligned} &\forall\, i, j{:}\mathbb{Z}. \quad 1 \leq i < j \leq \mathsf{n} \\ &\qquad \to\ \mathsf{pos}(j) - \mathsf{pos}(i) < -\textstyle\sum_{k=i}^{j-1} \mathsf{brakingDist}(\mathsf{speed}(k+1)), \end{aligned}$$

which can be rewritten to

$$(\mathsf{Init}_\mathsf{p}) \quad \begin{aligned} &\forall\, i, j{:}\mathbb{Z}. \quad 1 \leq i < j \leq n \\ &\qquad \to\ \mathsf{pos}(i) - \mathsf{pos}(j) > \textstyle\sum_{k=i+1}^{j} \mathsf{brakingDist}(\mathsf{speed}(k)). \end{aligned}$$

Let $(\mathsf{Init}_5) = (\mathsf{Init}_\mathsf{s}) \cup (\mathsf{Init}_\mathsf{p}) \cup (\mathsf{Global}) \cup (\mathsf{BDist})$.

The transition relation is given as $(\mathsf{Update}_5) = (\mathsf{PosRep}) \ \vee\ (\mathsf{Emerg})$, where $(\mathsf{PosRep}) = \bigwedge_{i=1}^{6} \mathsf{P}_i$ with

$$
\begin{array}{lll}
(\mathsf{P}_1) & \forall\, i{:}\mathbb{Z}. & i = 1 \,\wedge\, \mathsf{emergTrain} > i \,\rightarrow\, \mathsf{min} \leq \mathsf{speed}'(i) \leq \mathsf{max} \\
(\mathsf{P}_2) & \forall\, i{:}\mathbb{Z}. & 1 < i < \mathsf{emergTrain} \,\wedge\, \mathsf{pos}(i-1) - \mathsf{pos}(i) \geq \mathsf{d} \\
 & & \rightarrow\, \mathsf{min} \leq \mathsf{speed}'(i) \leq \mathsf{max} \\
(\mathsf{P}_3) & \forall\, i{:}\mathbb{Z}. & 1 < i < \mathsf{emergTrain} \,\wedge\, \mathsf{pos}(i-1) - \mathsf{pos}(i) < \mathsf{d} \\
 & & \rightarrow\, \mathsf{speed}'(i) = \mathsf{min} \\
(\mathsf{P}_4) & \forall\, i{:}\mathbb{Z}. & i \geq \mathsf{emergTrain} \\
 & & \rightarrow\, \mathsf{speed}'(i) = \mathsf{max}\{\mathsf{speed}(i) - \mathsf{maxDec} \cdot \Delta_\mathsf{t}, 0\} \\
(\mathsf{P}_5) & \forall\, i{:}\mathbb{Z}. & 1 \leq i \leq \mathsf{n} \,\rightarrow\, \mathsf{pos}'(i) = \mathsf{pos}(i) + \mathsf{speed}'(i) \cdot \Delta_\mathsf{t} \\
(\mathsf{P}_6) & & \mathsf{newEmergTrain}' > 0,
\end{array}
$$

and $(\mathsf{Emerg}) = \bigwedge_{i=1}^{5} \mathsf{E}_i$ with

$$
\begin{array}{lll}
(\mathsf{E}_1) & & \mathsf{newEmergTrain} \leq \mathsf{n} \\
(\mathsf{E}_2) & & \mathsf{emergTrain}' = \mathsf{min}\{\mathsf{emergTrain}, \mathsf{newEmergTrain}\} \\
(\mathsf{E}_3) & & \mathsf{speed}'(\mathsf{emergTrain}') = 0 \\
(\mathsf{E}_4) & \forall\, i{:}\mathbb{Z}. & i \neq \mathsf{emergTrain} \,\rightarrow\, \mathsf{speed}'(i) = \mathsf{speed}(i) \\
(\mathsf{E}_5) & & \mathsf{newEmergTrain}' > 0.
\end{array}
$$

If $\mathsf{emergTrain} > \mathsf{n}$, then rules (P_1) to (P_4) are similar to (F_1) to (F_4), except that they define speed' explicitly, which is then used to assign pos' in rule (P_5). (P_6) assigns nondeterministically a positive value to $\mathsf{newEmergTrain}'$, modeling the possibility of an emergency at an arbitrary location on the track.

If there is already an emergency on the track, i.e. if $\mathsf{emergTrain} \leq \mathsf{n}$, then all trains i with $i \geq \mathsf{emergTrain}$ apply the emergency brake, while the others move on as usual.

Whenever $\mathsf{newEmergTrain} \leq \mathsf{n}$, we can have an (Emerg) transition. This assigns $\mathsf{emergTrain}'$ to the value of $\mathsf{newEmergTrain}$, unless we already had an emergency at one of the preceding trains ($\mathsf{emergTrain} < \mathsf{newEmergTrain}$). The speed of the emergency train is set to 0 immediately, while all other trains keep their current speed and have to cope with the emergency after the next position report.

Verification of the Model with Emergencies

Again, we want to show absence of collisions in our model, this time in presence of nondeterministic emergencies. We already defined a more restrictive initial condition in order to cope with braking distances. Our safety condition is defined as $(\mathsf{Safe}_5) = (\mathsf{Init}_\mathsf{s}) \cup (\mathsf{Init}_\mathsf{p})$.

Proving safety is similar to what we had before. (Safe_5) is an inductive invariant of T_5 if and only if

$$\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{BDist}) \cup (\mathsf{Safe}_5) \cup (\mathsf{Update}_5) \cup \neg(\mathsf{Safe}_5') \models \Box,$$

in a suitable background theory $\mathcal{T}_0$. Since (Update_5) is a disjunction $(\mathsf{PosRep}) \vee (\mathsf{Emerg})$, we can split the proof task. (Safe_5) is an inductive invariant of T_5 if and only if both

$$\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{BDist}) \cup (\mathsf{Safe}_5) \cup (\mathsf{PosRep}) \cup \neg(\mathsf{Safe}_5') \models \Box,$$

and

$$\mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{BDist}) \cup (\mathsf{Safe}_5) \cup (\mathsf{Emerg}) \cup \neg(\mathsf{Safe}_5') \models \Box.$$

Locality. Again, we to show that we can split the resulting theories into chains of universally reducing extensions. Let $\mathcal{T}_1 = \mathcal{T}_0 \cup (\mathsf{Global})$. Again, the extension $\mathcal{T}_0 \subseteq \mathcal{T}_1$ is trivial.

Next, we consider $\mathcal{T}_2 = \mathcal{T}_1 \cup (\mathsf{BDist})$. The following is a corollary of Corollary 3.27.

Corollary 5.20. *The extension $\mathcal{T}_1 \subseteq \mathcal{T}_1 \cup (\mathsf{BDist})$ is universally reducing.*

Now, consider the theory $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Safe}_5)$. This extension does not fit into any class of local theory extensions that we have defined, and we cannot use Theorem 2.53 for combining extensions since pos is defined in terms of speed, i.e. the function symbols are not independent. Thus, we partition (Safe_5) into $(\mathsf{Init}_\mathsf{s}) \cup (\mathsf{Init}_\mathsf{p})$ and first consider the extension $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Init}_\mathsf{s})$. The following is a corollary of Theorem 3.19.

Corollary 5.21. *The extension $\mathcal{T}_2 \subseteq \mathcal{T}_2 \cup (\mathsf{Init}_\mathsf{s})$ is universally reducing.*

Now, let $\mathcal{T}_3 = \mathcal{T}_2 \cup (\mathsf{Init}_\mathsf{s})$. The following is a consequence of Corollary 3.37.

Corollary 5.22. *The extension $\mathcal{T}_3 \subseteq \mathcal{T}_3 \cup (\mathsf{Init}_\mathsf{p})$ is universally reducing.*

Thus far, we have shown that we can decide the universal fragment of $\mathcal{T}_4 = \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{BDist}) \cup (\mathsf{Safe}_5)$ (assuming that we can decide the universal fragment of $\mathcal{T}_0$). Now, we still have to consider extensions $\mathcal{T}_4 \subseteq \mathcal{T}_4 \cup (\mathsf{PosRep})$ and $\mathcal{T}_4 \subseteq \mathcal{T}_4 \cup (\mathsf{Emerg})$. As with (Safe_5), the extension with (PosRep) does not fit into any class of local theory extensions that we have defined, and in this case pos' is defined in terms of speed'. We partition (PosRep) into $(\mathsf{P_i})_{\mathsf{i}=1}^4 \cup (\mathsf{P_i})_{\mathsf{i}=5}^6$ and show that the resulting chain of extensions is universally reducing:

Corollary 5.23. *The chain of extensions $\mathcal{T}_4 \subseteq \mathcal{T}_4 \cup (\mathsf{P_i})_{\mathsf{i}=1}^4 \subseteq \mathcal{T}_4 \cup (\mathsf{PosRep})$ is universally reducing.*

Proof: An extension of $\mathcal{T}_1$ with axioms $(\mathsf{P_1})$ to $(\mathsf{P_4})$ is local by Theorem 3.19: clearly, the left-hand sides of the implications are mutually exclusive, and $\mathsf{min} < \mathsf{max}$ holds in $\mathcal{T}_1$. Obviously it is also universally reducing.

An extension of $\mathcal{T}_4 \cup (\mathsf{P_i})_{\mathsf{i}=1}^4$ with $(\mathsf{P_5})$ is again local by Theorem 3.19, and adding the ground clause $(\mathsf{P_6})$ does not destroy locality by Theorem 3.1. Thus, the extension $\mathcal{T}_4 \cup (\mathsf{P_i})_{\mathsf{i}=1}^4 \subseteq \mathcal{T}_4 \cup (\mathsf{PosRep})$ is local (and obviously universally reducing).

Since both extensions are universally reducing, the chain of extensions $\mathcal{T}_4 \subseteq \mathcal{T}_4 \cup (\mathsf{P_i})_{\mathsf{i}=1}^4 \subseteq \mathcal{T}_4 \cup (\mathsf{PosRep})$ is universally reducing. □

Corollary 5.24. *The extension $\mathcal{T}_4 \subseteq \mathcal{T}_4 \cup (\mathsf{Emerg})$ is universally reducing.*

Proof: An extension of $\mathcal{T}_1$ with axiom $(\mathsf{P_4})$ is a local extension by Theorem 3.19. Adding the remaining ground clauses does not destroy locality by Theorem 3.1.

The extension is universally reducing since all variables appear below extension functions. □

Hierarchical reasoning. Locality of the extensions allows us to reduce the two satisfiability problems above to equisatisfiable ground problems over $\mathcal{T}_0$:

$$
\begin{aligned}
& \mathcal{T}_4 \cup (\mathsf{PosRep}) \cup \neg(\mathsf{Safe}'_5) \models \square \\
\Leftrightarrow\ & \mathcal{T}_4 \cup (\mathsf{P}_i)_{i=1}^4 \cup \underbrace{(\mathsf{P}_i)_{i=5}^6[\neg(\mathsf{Safe}'_5)] \cup \neg(\mathsf{Safe}'_5)}_{G_0} \models \square \\
\Leftrightarrow\ & \mathcal{T}_2 \cup (\mathsf{Safe}_5) \cup \underbrace{(\mathsf{P}_i)_{i=1}^4[G_0] \cup G_0}_{G_1} \models \square \\
\Leftrightarrow\ & \mathcal{T}_2 \cup (\mathsf{Init_s}) \cup \underbrace{(\mathsf{Init_p})[G_1] \cup G_1}_{G_2} \models \square \\
\Leftrightarrow\ & \mathcal{T}_1 \cup (\mathsf{BDist}) \cup \underbrace{(\mathsf{Init_s}))[G_2] \cup G_2}_{G_3} \models \square \\
\Leftrightarrow\ & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{BDist})[G_3] \cup G_3 \models \square,
\end{aligned}
$$

as well as

$$
\begin{aligned}
& \mathcal{T}_4 \cup (\mathsf{Emerg}) \cup \neg(\mathsf{Safe}'_5) \models \square \\
\Leftrightarrow\ & \mathcal{T}_2 \cup (\mathsf{Safe}_5) \cup \underbrace{(\mathsf{Emerg})[\neg(\mathsf{Safe}'_5)] \cup \neg(\mathsf{Safe}'_5)}_{G'_0} \models \square \\
\Leftrightarrow\ & \mathcal{T}_2 \cup (\mathsf{Init_s}) \cup \underbrace{(\mathsf{Init_p})[G'_0] \cup G'_0}_{G'_1} \models \square \\
\Leftrightarrow\ & \mathcal{T}_1 \cup (\mathsf{BDist}) \cup \underbrace{(\mathsf{Init_s}))[G'_1] \cup G'_1}_{G'_2} \models \square \\
\Leftrightarrow\ & \mathcal{T}_0 \cup (\mathsf{Global}) \cup (\mathsf{BDist})[G'_2] \cup G'_2 \models \square.
\end{aligned}
$$

The resulting sets of ground clauses are non-linear (if we do not fix a value for Δ_t), and contain summation by $\sum$. To solve the resulting satisfiability problem, we need a reasoner for the base theory that can handle constraints containing these terms (which is not the case for state of the art SMT tools).

However, we can use a similar approach as for the model with length of trains, i.e. using an equisatisfiable set of ground clauses instead of the skolemization of $\neg(\mathsf{Safe}'_5)$. In particular, instead of the skolemized negation

$$1 \leq a < b \leq n \ \wedge\ \mathsf{pos}'(a) - \mathsf{pos}'(b) \leq \textstyle\sum_{k=a+1}^{b} \mathsf{brakingDist}(\mathsf{speed}'(k))$$

of $(\mathsf{Init}'_\mathsf{p})$, we can use the equisatisfiable formula

$$1 \leq a < n \ \wedge\ \mathsf{pos}'(a) - \mathsf{pos}'(a+1) \leq \mathsf{brakingDist}(\mathsf{speed}'(a+1)).$$

As a result, all sums in the instantiation of $(\mathsf{Init_p})$ have either $0, 1$ or 2 components and can thus easily be expressed in a base theory without summation symbol $\sum$.

For this model, we do not have evidence of how LIG^* and LIG^*_uc would compare to the eager approach. Its size makes manual investigation cumbersome and error-prone, and our prototype implementation currently does not support all features needed to handle this problem.

Combination of the Extensions

The extension with emergency handling can be combined with the extensions from Sections 5.5.1 and 5.5.2.

When combined with the extension from Section 5.5.1, we use rules $(\mathsf{V}_i)_{i=1}^9$ as a basis for our rules, instead of $(\mathsf{F}_i)_{i=1}^4$. Like in the step from Section 5.4

to 5.5.1, we introduce new system variables $\mathsf{first}, \mathsf{last}$, a new system parameter $\mathsf{maxTrains}$ instead of n, and modify (PosRep) and (Emerg) accordingly. The locality argument is the same as before, since the essential difference between the two version is in the ground part of the update rules.

The combination with the extensions from Section 5.5.2 is even easier: we can simply add the constant distance $\mathsf{LengthTrain}$ to our safety distance, i.e. replace $\mathsf{brakingDist}(\mathsf{speed}(i))$ with $\mathsf{brakingDist}(\mathsf{speed}(i)) + \mathsf{LengthTrain}$. As the latter is still a term in the base theory, our locality argument remains the same.

Chapter 6

Conclusions

In this dissertation, we have advanced the state of the art for the framework of local theory extensions, and thus the field of automated reasoning in complex theories, in different directions. This includes new locality results, a more efficient method of local reasoning, as well as applications of local reasoning in the verification of parameterized systems.

We have identified several local theory extensions that have not been known to be local before. Extensions of previous results for monotone functions include different versions of strict monotonicity, as well as quasi-monotonicity, which does not refer to a (partial) ordering in the domain of a function, but to any transitive relation. Furthermore, we identified a class of guarded boundedness axioms with a locality property. These allow us to define update rules for parameterized systems. We proved locality of different data structure specifications via constructor and selector functions, and finally showed that axioms modelling cardinality functions for boolean algebras also satisfy a locality property.

By applying ideas from first-order instance generation methods, we improved efficiency of reasoning in local theory extensions. We developed incremental versions of reasoning modulo a single extension and a chain of extensions, and have implemented both approaches. Experimental results give evidence that the incremental method is in average more efficient than the standard method, and dramatically so for a certain class of examples.

As one possible application of our decidability results we have shown that we can decide safety properties of a certain class of parameterized systems. We have developed successively more complex models of the ETCS case study, and have shown that interesting safety properties for all of these systems can be decided with local reasoning.

This dissertation demonstrates that local theory extensions can provide a framework with unified notation and common proof methodology for a broad range of interesting decidability results. An efficient implementation and example applications show that from this theoretical framework we can obtain a practical tool for verification and other applications of automated reasoning.

6.1 Related Work

Reasoning in complex theories. Most of the existing work on automated reasoning procedures for solving quantified satisfiability problems modulo a background theory can be separated into two categories: On the one hand, there are decision procedures that consider rather uniform background theories (as opposed to complex combinations of several background theories) allowing quantifier elimination, like linear arithmetic [68], fragments of the theory of arrays [28, 9] or sets with cardinality constraints [43]. On the other hand, there are methods that use heuristics for quantifier instantiation [12, 26]. These work in complex theories, but sacrifice completeness. There has also been some work on integrating theory reasoning with first-order theorem proving methods [62, 23, 53], but in general these methods also sacrifice completeness.

In contrast to these methods, we focus on classes of satisfiability problems that we can show to be decidable, even in the presence of complex background theories. If we can show that a set of axioms falls into one of the classes that we have identified as a *local theory extension* of a background theory for which we can decide the universal fragment (or in some cases the $\forall\exists$-fragment), then we can decide the universal fragment of the extended theory. Very recently, a method for complete instantiation strategies for some first-order fragments has been developed [27]. With respect to the distinction made above, this is probably the closest to our own work.

Verification of parameterized systems. We have shown that our decision procedures can be used to decide safety properties of systems with a parametric number of components, where the state of each component is defined by several values over the real numbers.

In related work, the components of such systems are often subject to restrictions that we do not consider here. Typical restrictions are that each component of the system needs to be restricted to a finite number of states [4], or each component is a timed (but otherwise finite-state) automaton [2].

On the other hand, there are approaches that allow infinite-state components, but in turn use an approximation on the state of the overall system. E.g., a regular model checking procedure has been developed [3], which can handle a parametric number of homogeneous linear processes and systems operating on queues or stacks. There is also work on the analysis of safety properties for parameterized systems with an arbitrary number of processes operating on unbounded integer variables [1, 10, 45]. By using abstractions of the state space or the transition relation of the system, all of these approaches sacrifice completeness.

In contrast to these approaches, our reasoning over the system is complete and does not restrict the state representation of the components. The only restriction we impose is that both the transition relation of the system and the property we want to prove can be expressed in terms of a a chain of local theory extensions of a suitable base theory. Also, we currently assume the invariant to be user-specified and have no automatic method to generate invariants, like some of the approaches above do. An approach which is similar to ours is taken by Ghilardi et al. [29] for so-called *array-based systems*. The main differences to our approach are on the one hand that they have a strict separation between index and element theories, and require element theories to be decidable. On

the other hand, they consider the more general problem of model checking for these systems, and can also handle a class of liveness properties.

6.2 Future Work

The results of this dissertation can be separated into three directions: new locality results, combination of instance generation methods and local reasoning, and applications of local reasoning in verification problems. In all three directions, we can identify several possibilities for future work.

New locality results. We believe that several existing decidability results can be expressed as local theory extensions of decidable theories, e.g. bridging functions between theories of sets and arithmetic as a generalization of cardinality functions. Also, we aim at expressing more complex theories of data structures as local theory extensions.

Instance generation and locality. The incremental instantiation procedure we have introduced works only for standard locality, but can be extended to stable and Ψ-locality. There are also other interesting applications of instance generation in problems with a large but finite search space of instantiations, like SMT-based synthesis [19] or finite model finding [41].

Applications in verification. We are currently developing another extension of our model for the ETCS case study, which uses recent locality results that e.g. allow us to model behavior of trains on a complex track network instead of only a single track.

Also, with several local axiom sets that describe data structures at our disposal, it seems worthwhile to investigate which software verification problems that are problematic for other approaches can be solved by local reasoning.

Zusammenfassung

Informationsverarbeitende Systeme sind in unserer Gesellschaft allgegenwärtig und werden von Tag zu Tag komplizierter. Dabei kann es sich um reine Hardware- oder Softwaresysteme handeln, oder um komplexe Systeme von Hardware und Software, die mit ihrer physikalischen Umgebung interagieren.

Mittels Verifikation kann sichergestellt werden, dass ein System sich in der erwarteten Weise verhält. Es gibt viele sicherheitskritische Systeme, bei denen eine Abweichung vom erwarteten Verhalten erhebliche Risiken mit sich bringt, wie etwa computergesteuerte medizinische Geräte, automatische Steuerungssysteme für Autos, Züge und Flugzeuge, oder die computergesteuerte Abschaltung von Kernkraftwerken im Notfall.

Manuelle Verifikation von Systemeigenschaften ist mühsam und fehleranfällig. Die Größe der genannten Systeme macht es unmöglich, ihr Verhalten vollständig von Hand zu verifizieren. Deshalb benötigen wir automatische oder computergestützte Verifikationsmethoden.

Bei der Analyse und Verifikation von Systemen ist automatisches Beweisen bereits weit verbreitet. Für reine Hardwaresysteme und eine eingeschränkte Klasse von Software sind die auftretenden Verifikationsprobleme von Natur aus endlich. Entscheidungsverfahren für propositionale Logik (SAT-Solver) können solche Probleme sehr effizient lösen. Für komplexe Systeme und andere Arten von Software kann eine Endlichkeit der Verifikationsprobleme nicht angenommen werden. Um Eigenschaften solcher Systeme ausdrücken und beweisen zu können, brauchen wir eine formale Sprache und Beweismethoden, die mit universeller Quantifizierung, arithmetischen Ausdrücken und unbeschränkten Datentypen gleichzeitig umgehen können. Es gibt Beweissysteme für Prädikatenlogik, die gut mit quantifizierten Formeln umgehen können, und effiziente Entscheidungsverfahren für (nicht quantifizierte) Probleme in Theorien von Arithmetik und Datentypen. Um die anfallenden Verifikationsprobleme zu lösen, benötigen wir Verfahren, die die Stärken dieser beiden Ansätze zusammenbringen.

Aus diesem Grund gab es in den letzten Jahren ein großes Interesse an Methoden, die universell quantifizierte Probleme in solchen Hintergrundtheorien lösen können. Es ist bekannt, dass solche Probleme im Allgemeinen unentscheidbar sind, und die Forschung konzentriert sich auf Methoden, die unter Verzicht auf Vollständigkeit möglichst viele Probleme schnell lösen können. Die bekannten Ansätze basieren entweder auf der Integration von Entscheidungsverfahren in Beweissysteme für Prädikatenlogik, oder auf der heuristischen Instanziierung von Quantoren. In beiden Fällen können viele Probleme aus einer bestimmten Klasse schnell gelöst werden, aber es gibt im allgemeinen keine Garantie, dass das Verfahren für ein gegebenes Problem terminiert.

Wir verfolgen einen anderen Ansatz und konzentrieren uns auf Problem-

klassen, die wir als lokale Theorieerweiterungen ausdrücken und somit ihre Entscheidbarkeit zeigen können. Liegt ein gegebenes Problem in einer solchen Klasse, so können wir es durch endliche Instanziierung der Quantoren effizient lösen und gleichzeitig das Terminieren der Prozedur garantieren.

Diese Dissertation basiert auf der Arbeit von Sofronie-Stokkermans an lokalen Theorieerweiterungen, sowie der Arbeit von Ganzinger und Korovin an instanziierungs-basierten Methoden zum Theorembeweisen in Prädikatenlogik erster Ordnung. Wir geben einen kurzen Überblick über die bisherigen Resultate in diesen beiden Bereichen, erweitern sie im Bereich lokaler Theorieerweiterungen und zeigen eine spezialisierte Anwendung instanziierungs-basierter Methoden in diesem Bereich.

Wir führen die Arbeit an lokalen Theorieerweiterungen fort, indem wir neue Beispiele von Axiomen geben, die eine Lokalitätseigenschaft erfüllen. Darunter fallen Axiome für die Modellierung streng monotoner Funktionen mit verschiedenen zusätzlichen Eigenschaften wie Beschränktheit oder einer minimalen Steigung sowie Funktionen mit monotonie-ähnlichen Eigenschaften. Wir zeigen auch, dass wir mit unserem Ansatz unbeschränkte rekursive Datentypen und Mengen mit Kardinalitätsfunktionen modellieren können. Schließlich stellen wir eine Klasse von Axiomen vor, mit deren Hilfe das Verhalten von Systemen modelliert werden kann.

Wir benutzen Ideen aus instanziierungs-basierten Methoden zum Theorembeweisen in Prädikatenlogik, um lokales Beweisen effizienter zu machen. Im bisherigen Ansatz zum lokalen Beweisen wird eine endliche Menge von Instanzen der Axiome berechnet, so dass das resultierende (nicht quantifizierte) Problem in der gegebenen Hintergrundtheorie genau dann erfüllbar ist, wenn das ursprüngliche Problem in der Theorieerweiterung erfüllbar ist. Diese Menge von Instanzen wird ohne Zwischenschritte berechnet und zusammen mit dem ursprünglichen Problem als ganzes an ein Entscheidungsverfahren für die Basistheorie gegeben. In unserem neuen Ansatz berechnen wir diese Menge von Instanzen inkrementell, indem wir wiederholt zwischen Instanziierung eines Teils der Axiome und Überprüfung der Erfüllbarkeit in der Basistheorie wechseln. Die resultierende Prozedur kann in vielen Fällen die Erfüllbarkeit oder Unerfüllbarkeit eines Problems zeigen, ohne die vollständige Menge von Instanzen aus dem bisherigen Ansatz zu berechnen. Unsere Experimente mit Implementierungen beider Ansätze zeigen, dass der neue Ansatz für die Mehrheit der getesteten Probleme effizienter ist.

Zuletzt zeigen wir Anwendungen der existierenden und der neuen Resultate in der Verifikation komplexer Systeme. Wir zeigen, dass wir für eine bestimmte Klasse von parametrisierten Systemen die Einhaltung einer bestimmten Klasse von Sicherheitseigenschaften entscheiden können. Als Beispiel entwickeln wir eine Reihe komplexer werdender Modelle eines eingebetteten Zugsteuerungssystems, deren Eigenschaften wir mit Axiomen beschreiben, die eine Lokalitätseigenschaft erfüllen. Unsere Modellierung beschreibt Kontrollsysteme für eine unbeschränkte Anzahl von Komponenten mit jeweils unendlich vielen möglichen Zuständen. Wir zeigen, wie man mittels lokalen Beweisens Sicherheitseigenschaften solcher Systeme verifizieren kann.

Bibliography

[1] Parosh Aziz Abdulla, Giorgio Delzanno, and Ahmed Rezine. Approximated context-sensitive analysis for parameterized verification. In David Lee, Antónia Lopes, and Arnd Poetzsch-Heffter, editors, *Formal Techniques for Distributed Systems, FMOODS/FORTE'09*, volume 5522 of *LNCS*, pages 41–56. Springer, 2009.

[2] Parosh Aziz Abdulla and Bengt Jonsson. Verifying networks of timed processes. In Bernhard Steffen, editor, *Tools for Analysis and Construction of Systems, TACAS'98*, volume 1384 of *LNCS*, pages 298–312. Springer, 1998.

[3] Parosh Aziz Abdulla, Bengt Jonsson, Marcus Nilsson, and Mayank Saksena. A survey of regular model checking. In Philippa Gardner and Nobuko Yoshida, editors, *Conference on Concurrency Theory, CONCUR'04*, volume 3170 of *LNCS*, pages 35–48. Springer, 2004.

[4] Tamarah Arons, Amir Pnueli, Sitvanit Ruah, Jiazhao Xu, and Lenore D. Zuck. Parameterized verification with automatically computed inductive assertions. In *Computer-Aided Verfication, CAV'01*, volume 2101 of *LNCS*, pages 221–234. Springer, 2001.

[5] Clark Barrett and Cesare Tinelli. CVC3. In Werner Damm and Holger Hermanns, editors, *Computer Aided Verification, CAV'07*, volume 4590 of *LNCS*, pages 298–302. Springer, 2007.

[6] Jon Barwise, editor. *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, Amsterdam, Holland, 1978.

[7] David Basin and Harald Ganzinger. Automated complexity analysis based on ordered resolution. *Journal of the ACM*, 48(1):70–109, 2001.

[8] Aaron R. Bradley and Zohar Manna. *The Calculus of Computation: Decision Procedures with Applications to Verification*. Springer-Verlag New York, Inc., 2007.

[9] Aaron R. Bradley, Zohar Manna, and Henny B. Sipma. What's decidable about arrays? In E. Allen Emerson and Kedar S. Namjoshi, editors, *Verification, Model-Checking, and Abstract-Interpretation, VMCAI'06*, volume 3855 of *LNCS*, pages 427–442. Springer, 2006.

[10] Edmund M. Clarke, Muralidhar Talupur, and Helmut Veith. Environment abstraction for parameterized verification. In *Verification, Model Checking, and Abstract Interpretation, VMCAI'06*, volume 3855 of *LNCS*, pages 126–141. Springer, 2006.

[11] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In C.R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'08*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[12] David Detlefs, Greg Nelson, and James B. Saxe. Simplify: a theorem prover for program checking. *Journal of the ACM*, 52(3):365–473, 2005.

[13] Andreas Dolzmann and Thomas Sturm. Redlog: computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, 1997.

[14] Bruno Dutertre and Leonardo de Moura. The Yices SMT solver. Available at http://yices.csl.sri.com/tool-paper.pdf, 2006.

[15] Niklas En and Niklas Sörensson. An extensible SAT-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing, SAT'03*, volume 2919 of *LNCS*, pages 333–336. Springer, 2004.

[16] Johannes Faber. Verifying real-time aspects of the European Train Control System. In *Nordic Workshop on Programming Theory, NWPT'05*, pages 67–70. University of Copenhagen, Denmark, 2005.

[17] Johannes Faber, Swen Jacobs, and Viorica Sofronie-Stokkermans. Verifying CSP-OZ-DC specifications with complex data types and timing parameters. In Jim Davies Jeremy Gibbons, editor, *Integrated Formal Methods, IFM'07*, volume 4591 of *LNCS*, pages 233–252. Springer, 2007.

[18] Johannes Faber and Roland Meyer. Model checking data-dependent real-time properties of the european train control system. In *Formal Methods in Computer Aided Design, FMCAD'06*, pages 76–77. IEEE, 2006.

[19] Bernd Finkbeiner and Sven Schewe. SMT-based synthesis of distributed systems. In *Automated Formal Methods, AFM'07*, pages 69–76, 2007.

[20] Harald Ganzinger. Relating semantic and proof-theoretic concepts for polynomial time decidability of uniform word problems. In *IEEE Symposium on Logic in Computer Science, LICS'01*, pages 81–92. IEEE, 2001.

[21] Harald Ganzinger and Konstantin Korovin. New directions in instantiation-based theorem proving. In *IEEE Symposium on Logic in Computer Science, LICS'03*, pages 55–64, Ottawa, Canada, 2003. IEEE.

[22] Harald Ganzinger and Konstantin Korovin. Integration of equational reasoning into instantiation-based theorem proving. In Jerzy Marcinkowski and Andrzej Tarlecki, editors, *Computer Science Logic, CSL'04*, volume 3210 of *LNCS*, pages 71–84, Karpacz, Poland, 2004. Springer.

[23] Harald Ganzinger and Konstantin Korovin. Theory instantiation. In Miki Hermann and Andrei Voronkov, editors, *Logic for Programming Artificial Intelligence Reasoning, LPAR'06*, volume 4246 of *LNCS*, pages 497–511, Phnom Penh, Cambodia, 2006. Springer.

[24] Harald Ganzinger, Viorica Sofronie-Stokkermans, and Uwe Waldmann. Modular proof systems for partial functions with weak equality. In David Basin and Michael Rusinowitch, editors, *Automated reasoning : Second International Joint Conference, IJCAR'04*, volume 3097 of *Lecture Notes in Artificial Intelligence*, pages 168–182. Springer, 2004.

[25] Harald Ganzinger, Viorica Sofronie-Stokkermans, and Uwe Waldmann. Modular proof systems for partial functions with Evans equality. *Information and Computation*, 204(10):1453–1492, 2006.

[26] Yeting Ge, Clark Barrett, and Cesare Tinelli. Solving quantified verification conditions using satisfiability modulo theories. In Frank Pfenning, editor, *Conference on Automated Deduction, CADE-21*, volume 4603 of *LNCS*, pages 167–182. Springer, 2007.

[27] Yeting Ge and Leonardo de Moura. Complete instantiation for quantified SMT formulas. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification, CAV'09*, volume 5643 of *LNCS*, pages 306–320. Springer, 2009.

[28] Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Deciding extensions of the theory of arrays by integrating decision procedures and instantiation strategies. In Michael Fisher, Wiebe van der Hoek, Boris Konev, and Alexei Lisitsa, editors, *European Conference on Logics in Artificial Intelligence, JELIA'06*, volume 4160 of *LNCS*, pages 177–189. Springer, 2006.

[29] Silvio Ghilardi, Enrica Nicolini, Silvio Ranise, and Daniele Zucchelli. Towards smt model checking of array-based systems. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR'08*, pages 67–82, 2008.

[30] Robert Givan and David A. McAllester. New results on local inference relations. In Bernhard Nebel, Charles Rich, and William R. Swartout, editors, *Principles of Knowledge Representation and Reasoning, KR'92*, pages 403–412. Morgan Kaufmann Press, 1992.

[31] Robert Givan and David A. McAllester. Polynomial-time computation via local inference relations. *ACM Transactions on Computational Logic*, 3(4):521–541, 2002.

[32] Jochen Hoenicke. *Combination of Processes, Data, and Time*. PhD thesis, University of Oldenburg, Germany, 2006.

[33] Jochen Hoenicke and Patrick Maier. Model-checking of specifications integrating processes, data and time. In John Fitzgerald, Ian J. Hayes, and Andrzej Tarlecki, editors, *Formal Methods, FM'05*, volume 3582 of *LNCS*, pages 465–480. Springer, 2005.

[34] Jochen Hoenicke and Ernst-Rüdiger Olderog. CSP-OZ-DC: A combination of specification techniques for processes, data and time. *Nordic Journal of Computing*, 9(4):301–334, 2002.

[35] Carsten Ihlemann. PhD thesis, Saarland University, Germany. To appear.

[36] Carsten Ihlemann, Swen Jacobs, and Viorica Sofronie-Stokkermans. On local reasoning in verification. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, TACAS'08*, volume 4963 of *LNCS*, pages 265–281, Budapest, Hungary, 2008. Springer.

[37] Swen Jacobs. Incremental instance generation in local reasoning. In Franz Baader, Silvio Ghilardi, Miki Hermann, Ulrike Sattler, and Viorica Sofronie-Stokkermans, editors, *Complexity, Expressibility, and Decidability in Automated Reasoning, CEDAR'08*, pages 47–62, 2008.

[38] Swen Jacobs. Incremental instance generation in local reasoning. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification, CAV'09*, volume 5643 of *LNCS*, pages 368–382. Springer, 2009.

[39] Swen Jacobs and Viorica Sofronie-Stokkermans. Applications of hierarchical reasoning in the verification of complex systems. In Byron Cook and Roberto Sebastiani, editors, *Pragmatical Aspects of Decision Procedures in Automated Reasoning, PDPAR'07*, pages 15–26, 2006.

[40] Swen Jacobs and Viorica Sofronie-Stokkermans. Applications of hierarchical reasoning in the verification of complex systems. *Electronic Notes in Theoretical Computer Science*, 174(8):39–54, 2007.

[41] Viktor Kuncak and Daniel Jackson. Relational analysis of algebraic datatypes. In Michel Wermelinger and Harald Gall, editors, *European Software Engineering Conference, ESEC'05, and Symposium on Foundations of Software Engineering, FSE'05*, pages 207–216. ACM, 2005.

[42] Viktor Kuncak, Huu Hai Nguyen, and Martin Rinard. Deciding Boolean Algebra with Presburger Arithmetic. *Journal of Automated Reasoning*, 36(3):213–239, 2006.

[43] Viktor Kuncak and Martin Rinard. An overview of the Jahob analysis system: Project goals and current status. In *Next Generation Software Workshop, NSF'06*, 2006.

[44] Viktor Kuncak and Martin Rinard. Towards efficient satisfiability checking for Boolean Algebra with Presburger Arithmetic. In Frank Pfenning, editor, *Conference on Automated Deduction, CADE-21*, volume 4603 of *LNCS*. Springer, 2007.

[45] Shuvendu K. Lahiri and Randal E. Bryant. Indexed predicate discovery for unbounded system verification. In *Computer-Aided Verfication, CAV'04*, volume 3114 of *LNCS*, pages 135–147. Springer, 2004.

[46] Scott McPeak and George C. Necula. Data structure specifications via local equality axioms. In Kousha Etessami and Sriram K. Rajamani, editors, *Computer Aided Verification, CAV'05*, volume 3576 of *LNCS*, pages 476–490. Springer, 2005.

[47] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Design Automation Conference, DAC'01*, pages 530–535. IEEE, 2001.

[48] Greg Nelson and Derek C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980.

[49] Hans Jürgen Ohlbach. Set description languages and reasoning about numerical features of sets. Technical Report PMS-FB-2001-1, TU München, 2001.

[50] Hans Jürgen Ohlbach and Jana Köhler. How to extend a formal system with a boolean algebra component. In W. Bibel P.H. Schmidt, editor, *Automated Deduction. A Basis for Applications*, volume III, pages 57–75. Kluwer Academic Publishers, 1998.

[51] Derek C. Oppen. Reasoning about recursively defined data structures. *Journal of the ACM*, 27(3):403–411, 1980.

[52] M. Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Sprawozdanie z I Kongresu metematyków slowiańskich*, pages 92–101, 1929.

[53] Virgile Prevosto and Uwe Waldmann. SPASS+T. In Geoff Sutcliffe, Renate Schmidt, and Stephan Schulz, editors, *Empirically Successful Computerized Reasoning, ESCoR'06*, volume 192 of *CEUR Workshop Proceedings*, pages 18–33, 2006.

[54] Alexandre Riazanov and Andrei Voronkov. The design and implementation of vampire. *AI Communications*, 15(2–3):91–110, 2002.

[55] Stephan Schulz. E — a brainiac theorem prover. *AI Communications*, 15(2–3):111–126, 2002.

[56] Viorica Sofronie-Stokkermans. Hierarchic reasoning in local theory extensions. In Robert Nieuwenhuis, editor, *Conference on Automated Deduction, CADE-20*, volume 3632 of *LNAI*, pages 219–234. Springer, 2005.

[57] Viorica Sofronie-Stokkermans. Interpolation in local theory extensions. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning: 3rd International Joint Conference, IJCAR'06*, volume 4130 of *LNCS*, pages 235–250. Springer, 2006.

[58] Viorica Sofronie-Stokkermans. Local reasoning in verification. In Serge Autexier and Heiko Mantel, editors, *VERIFY'06: Verification Workshop*, pages 128–145, 2006.

[59] Viorica Sofronie-Stokkermans. Hierarchical and modular reasoning in complex theories: The case of local theory extensions. In Boris Konev and Frank Wolter, editors, *Frontiers of Combining Systems, FroCos'07)*, volume 4720 of *LNCS*, pages 47–71. Springer, 2007.

[60] Viorica Sofronie-Stokkermans. Locality results for certain extensions of theories with bridging functions. In *Conference on Automated Deduction, CADE-22*, pages 67–83. Springer, 2009.

[61] Viorica Sofronie-Stokkermans and Carsten Ihlemann. Automated reasoning in some local extensions of ordered structures. *Journal of Multiple-Valued Logic and Soft Computing*, 13(4-6):397–414, 2007.

[62] Mark E. Stickel. Automated deduction by theory resolution. *Journal of Automated Reasoning*, 1(4):333–355, 1985.

[63] Geoff Sutcliffe. The 4th IJCAR automated theorem proving system competition - CASC-J4. *AI Communications*, 22(1):59–72, 2009.

[64] Edward Szpilrajn. Sur l'extension de l'ordre partiel. *Fundamenta Mathematicae*, XVI:386–389, 1930.

[65] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, Berkeley, 1951.

[66] Dirk van Dalen. *Logic and Structure*. Springer, Berlin, Heidelberg, fourth edition, 2004.

[67] Christoph Weidenbach, Renate A. Schmidt, Thomas Hillenbrand, Rostislav Rusev, and Dalibor Topic. System description: SPASS version 3.0. In Frank Pfenning, editor, *Automated Deduction — CADE-21*, volume 4603 of *LNCS*, pages 514–520. Springer, 2007.

[68] Volker Weispfenning. Mixed real-integer linear quantifier elimination. In *International Symposium on Symbolic and Algebraic Computation, ISSAC'99*, pages 129–136. ACM, 1999.

[69] Ting Zhang, Henny B. Sipma, and Zohar Manna. Decision procedures for term algebras with integer constraints. *Information and Computation*, 204(10):1526–1574, 2006.

Index