On Complete Reasoning about Axiomatic Specifications

EPFL-REPORT-151486

Swen Jacobs and Viktor Kuncak

École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
firstname.lastname@epfl.ch

Abstract. Automated software verification tools typically accept specifications of functions in terms of pre- and postconditions. However, many properties of functional programs can be more naturally specified using a more general form of universally quantified properties. Such general specifications may relate multiple user-defined functions, and compare multiple invocations of a function on different arguments.
We present new decision procedures for complete and terminating reasoning about such universally quantified properties of functional programs. Our results use local theory extension methodology. We establish new classes of universally quantified formulas whose satisfiability can be checked in a complete way by finite quantifier instantiation. These new classes include single-invocation axioms that generalize standard function contracts, but also certain many-invocation axioms, specifying that functions satisfy congruence, injectivity, or monotonicity with respect to abstraction functions, as well as conjunctions of some of these properties. These many-invocation axioms can specify correctness of abstract data type implementations as well as certain information-flow properties. We also present a construction that enables the same function to be specified using different classes of decidable specifications on different partitions of its domain. This results in complete and terminating decision procedure for proving an interesting class of universally quantified specifications of functional programs.

1 Introduction

Modular software verification. Modular software verification is a promising approach to prove software properties. One way to scalability and precision is to combine specification of procedures written by developers with the automation provided by modern satisfiability-modulo theory solvers [2, 6]. Recent work in modular verification includes tools VCC [4], Spec# [1], and Jahob [18, 21, 31]. As in the Canvas project [22] and the high-level analysis of Hob [19], we focus particularly on checking that specified functions are correctly used to implement the desired higher-level functionality. We simplify the problem by considering purely functional instead of imperative code, but consider more complex program properties than those checked by previous automated verification approaches.

Most existing approaches for specifying a function f use contracts that establish a relationship between a given input, x, and the output of the function, $f(x)$. Such contracts can be expressed as universally quantified statements $\forall x.\Phi(x, f(x))$, where Φ is a formula expressing the desired property of the function. Moreover, depending on the particular verification system, the property Φ may or may not refer to other functions in the system, with non-trivial issues arising in the use of imperative functions within contracts [5].

Algebraic specifications. In this paper, we consider a broad class of specifications that follow algebraic specification style [12]. Our specifications may relate different functions, using, for example, abstraction functions to specify the behavior of an abstract data type implementation. Moreover, our properties may include multiple universal quantifiers, which allows us to express congruence, monotonicity, injectivity, and non-interference properties of functions. Such properties cannot be directly expressed using standard pre/post condition specifications, which refer an arbitrary, but *only one* function invocation.

Sound, complete and terminating approach. We study the verification problem for such general properties from the theorem proving point of view. In analogy with modular verification of contracts, the verification conditions assume certain universally quantified formulas (specifying the behavior of basic functions) and the goal is to prove further universally quantified formulas that express the behavior of functions composed from basic ones. The key challenge is the presence of quantifiers in the assumptions. Current SMT provers typically have incomplete support for quantifiers. Therefore, they typically cannot establish that a quantified formulas is satisfiable, limiting the ability to provide meaningful counterexamples to the developer.

In this paper, we overcome the incompleteness of quantifier reasoning for a number of properties of interest. We use the methodology of local theory extensions [23] to obtain complete quantifier instantiation strategies. We therefore arrive at decision procedures for quantified formulas about functions. The procedure can prove the validity of universally quantified properties of functions, where functions themselves are specified with universally quantified axioms.

Our approach is terminating for all specifications in a given class. Moreover, it always either proves validity of the formula, or generates a concrete counterexample. This is a significant aspect of usability: the ability to do proofs ensures a higher degree of confidence than purely bug-finding approaches, whereas the ability to generate counterexamples helps users identify concrete errors.

Single-invocation axioms. As our first class of properties that specify a function f we consider single-invocation axioms of the form $\forall \bar{x}.\Phi(\bar{x}, f(\bar{x}))$ for a formula Φ in a decidable theory. We show that it suffices to instantiate the universal quantifier over $\bar{x}$ only with $\bar{a}$ such that $f(\bar{a})$ occurs in the property being proved. Single-invocation axioms are a fairly general result about local theory extensions and it has not been, to the best of our knowledge, documented as such before. This result also gives a systematic explanation of field constraint analysis [29], which was used to prove invariants of imperative data structures.

Such completeness of instantiation is not to be taken for granted. As an illustration, we show that this instantiation is *incomplete* for a slightly more general form of axioms, two-invocation axioms of the form $\forall \bar{x}, \bar{y}.\Phi(\bar{x}, \bar{y}, f(\bar{x}), f(\bar{y}))$.

Many-invocation axioms. Despite the limitations of locality for general two-invocation axioms, we show a number of useful two-invocation axioms that are local, and can thus be instantiated in a complete way. These tractable cases include properties of functions such as monotonicity, injectivity, congruence, and non-interference, all under a given abstraction function.

Locality of sufficiently surjective abstractions. The results on both single-invocation and many-invocation axioms are particularly useful in the presence of recursively defined functions. Recent result shows the decidability of theories with sufficiently surjective abstraction functions [28], which includes e.g. functions to compute set or multiset content, size, or height of algebraic data type values. We show that the result [28], like other cases of homomorphisms [25], can be explained using locality, which suggest more efficient implementations than the high-level description in [28].

Combination results. Given several classes of axioms for which reasoning is complete, a question arises whether they can be combined. We consider combinations of the axioms that are separately local, and show several positive and negative results for the locality of their combinations. Finally, we give a positive result on *piecewise* combinations of local axioms. This is of significant practical value, because it enables functions to be specified using different local axioms on different regions of their input domain, and generates a local axiom as a result.

In summary, we show a number of new decidability results for theories that support quantifiers, by showing that these quantifiers can be finitely instantiated in a complete way. In this way, we extend the predictability and efficiency of quantifier-free reasoning to interesting class of axiomatically specified functions.

2 Example

Figure 1 shows an illustrative functional program in the notation of the Scala programming language. We specify functions using global assertions written in Scala extended with a $\forall$ operator for universal quantification.

Many interesting properties of functional programs can be expressed by *contracts*, assigning a pre- and a postcondition to every call of a function, where the postcondition may depend on the value with which the function is called. The specification of function merge in Figure 1 illustrates how contracts can be encoded using axioms. The result of the function is a list containing the elements of both input lists. We express this requirement using an abstraction function contents, mapping a list to the set of elements it contains.

Some interesting properties of functions however are not expressible by contracts, because values of different invocations of the same function need to be compared. Consider a user-defined equality on data structures, defined by the abstraction function contents: $x \sim y \iff \mathsf{contents}(x) = \mathsf{contents}(y)$. Then, we

```
def merge(l1: List, l2: List): List = { ... }
assert(∀l1 : List⇒ ∀l2 : List⇒
        contents(merge(l1,l2)) = contents(l1) ∪ contents(l2))

def even(l : list): List = { ... }
assert(∀l: List ⇒ contents(even(l)) ⊆ contents(l))
assert(∀l1 : List⇒ ∀ l2 : List⇒
        contents(l1) = contents(l2) → contents(even(l1)) = contents(even(l2)))

def insert(c : Int, l: List): List = { ... }
assert(∀c: Int⇒ ∀l1 : List⇒ ∀l2 : List⇒
        contents(l1) ⊆ contents(l2) → contents(insert(c,l1)) ⊆ contents(insert(c,l2)))

def fill(l: List): List = { ... }
assert(∀l: List⇒ if (contents(l).size < 3) contents(fill(l)).size=3
                 else contents(fill(l)) = contents(l)

def main(c: Int, l1: List, l2: List): List = merge(even(l1), fill(insert(c,l2)))

case class Employee(name : String, age : Int, bankAccountNo : Int)
def samePublic1(e1 : Employee, e2 : Employee) : Boolean =
  { e1.name = e2.name && e1.age = e2.age }
def samePublic(el1 : List[Employee], el2 : List[Employee]) : Boolean =
   zip(el1,el2).forAll(samePublic1) && length(el1)=length(el2)
assert(equivalence(samePublic))

def averageAge(emp : List[Employee]) : Int = { ... }
// information flow property: averageAge does not depend on private data
assert(∀l1: List[Employee]⇒ ∀l2: List[Employee]⇒
  samePublic(l1,l2) → averageAge(l1) = averageAge(l2)
```

Fig. 1. Example of Functional Program Specified Using Axioms

know that $\sim$ is an equivalence relation, but we do not know whether the congruence axiom $x \sim y \rightarrow f(x) \sim f(y)$ holds, i.e. whether equivalence is preserved under function application. For many functions manipulating data structures, we want congruence wrt. contents, which means that in general we have to require it specifically, as for the function even that takes a list and returns the list of (integer) elements which are even numbers.

For some functions (like insert in Figure 1), we want to require stronger properties, like monotonicity wrt. to the abstraction and a suitable ordering on its codomain. In case of the contents abstraction, this would be the subset ordering, and we define $x \preceq y \iff \mathsf{contents}(x) \subseteq \mathsf{contents}(y)$. Then, we expect a function that inserts a given element into the data structure to satisfy $x \preceq y \rightarrow \mathsf{insert}(c, x) \preceq \mathsf{insert}(c, y)$.

It can also be useful to specify the data that a function is allowed to access. For example, averageAge in Figure 1) is a function that should not access

the private information about employees. We specify this by first defining an equivalence relation samePublic1 on employees, abstracting from the private data bankAccountNo, then an equivalence relation on lists of employees (which are equivalent if they have the same length and elements at the same position in a list are in relation samePublic1), and finally assert that whenever two lists are equivalent wrt. samePublic, then averageAge will give the same result for both lists.

Finally, we may also want to use conjunctive or disjunctive combinations of specifications, as shown in Figure 1 for functions even (which in addition to congruence wrt. contents should not insert new elements into a list) and fill (which in case of lists with less than 3 elements should insert additional elements, such that the resulting list has 3 distinct elements, and otherwise should not change the contents of the list).

As with usual contracts, our overall goal is to verify partial correctness of programs in a modular way. Assuming that assertions of the functions it calls do hold, we want to check properties of main like congruence or monotonicity wrt. one or both of its list arguments, whether the result will contain only elements from the input lists, or whether the result contains more than 3 elements (regardless of the size of input lists).

The approach that we will introduce allows us to decide such questions, assuming that given specifications for the called functions hold. Moreover, if a property cannot be proved, we can obtain a counterexample which points us to the weakness in our specification, and can be used to strengthen it.

3 Background

We give a short introduction to local theory extensions, which are key to our decidability results. For more details, we refer to [13, 14, 16, 23].

Theories and models. Consider a signature $\Pi = (\Sigma, \mathsf{Pred})$, where Σ is a set of function symbols and Pred a set of predicate symbols (both with given arities). A *Π-structure* $\mathcal{M}$ consists of a non-empty set of elements M, a total function $f^{\mathcal{M}} : M^n \to M$ for every n-ary function symbol $f \in \Sigma$, as well as a set $P^{\mathcal{M}} \subseteq M^n$ for every n-ary predicate symbol $P \in \mathsf{Pred}$. We regard *theories* as sets of formulas closed under consequences, defined by a set of *axioms*. A given (Π-)structure $\mathcal{M}$ is a *model* of a theory $\mathcal{T}$ iff every axiom of $\mathcal{T}$ is satisfied by $\mathcal{M}$. If a formula F is satisfied by a structure $\mathcal{M}$, we write $\mathcal{M} \models F$. If F is true in all models of $\mathcal{T}$, we write $\models_{\mathcal{T}} F$. A formula F_2 is a *consequence* of F_1 (modulo $\mathcal{T}$), written $F_1 \models_{\mathcal{T}} F_2$, if F_2 is true in every model of $\mathcal{T}$ that also satisfies F_1. If no model of $\mathcal{T}$ satisfies F, we write $F \models_{\mathcal{T}} \square$, where $\square$ represents the empty clause.

Local Theory Extensions. *Theory extensions* extend a given theory with new function symbols, defined by a set of axioms. *Locality* of the extension ensures that reasoning about these symbols can be reduced to reasoning in the base theory by finite instantiation of the axioms. The new symbols are called *extension symbols*, terms starting with extension symbols are *extension terms*.

Consider a background theory $\mathcal{T}$ with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred}_0)$, and an extension $\Pi = (\Sigma_0 \cup \Sigma_1, \mathsf{Pred})$ of this signature with additional function symbols

in Σ_1. An *augmented Π-clause* is a Π-formula $\forall \overline{x}.\ \Phi(\overline{x}) \vee C(\overline{x})$, where $\Phi(\overline{x})$ is an arbitrary Π_0-formula and $C(\overline{x})$ is a disjunction of Π-literals. We say that it is Σ_1-*ground* if $C(\overline{x})$ is ground. A *theory extension* of a theory $\mathcal{T}$ with signature Π_0 is given by a set $\mathcal{K}$ of augmented Π-clauses, representing axioms for the extension symbols.

A substitution σ is a function from variables to terms. By $F\sigma$ we denote the result of simultaneously replacing each free variable x in F with $\sigma(x)$. For a set of formulas $\mathcal{K}$, define $\mathsf{st}(\mathcal{K})$ as the set of ground subterms appearing in $\mathcal{K}$. For a set of Π-formulas $\mathcal{K}$ and a Π-formula G, let

$$\mathcal{K}[G] = \{\ F\sigma \mid \begin{array}{l} F \in \mathcal{K} \text{ and } \sigma \text{ is such that} \\ f(\overline{t})\sigma \in \mathsf{st}(\mathcal{K} \cup G) \text{ for each extension subterm } f(\overline{t}) \text{ of } F, \\ \text{and } \sigma(x) = x \text{ if } x \text{ does not appear in an extension term } \}. \end{array}$$

We consider theory extensions $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}$ with the following locality property:

(ELoc) For every set G of Σ_1-ground augmented Π-clauses, we have $\mathcal{K} \cup G \models_{\mathcal{T}} \Box \iff \mathcal{K}[G] \cup G \models_{\mathcal{T}} \Box$

Decidability. Assuming (ELoc), satisfiability of G modulo $\mathcal{T} \cup \mathcal{K}$ is decidable whenever $\mathcal{K}[G] \cup G$ is finite and belongs to a decidable fragment of $\mathcal{T}$ (plus free function symbols). In particular, if G is ground, and all universally quantified variables in $\mathcal{K}$ appear in extension terms, then $\mathcal{K}[G] \cup G$ is ground, and decidability of the ground fragment of $\mathcal{T}$ is sufficient.

Identifying Local Theory Extensions. To formulate a sufficient condition for theory extensions satisfying (ELoc), we need some additional definitions.

A *partial Π-structure* is the same as a Π-structure, except that function symbols may be assigned partial functions. In a partial structure $\mathcal{M}$, terms are evaluated wrt. a variable assignment β like in total structures, except that the evaluation of $\beta(f(t_1, \ldots, t_n))$ is undefined if either $(\beta(t_1), \ldots, \beta(t_n))$ is not in the domain of $f^{\mathcal{M}}$, or at least one of the $\beta(t_i)$ is undefined. A partial Π-structure $\mathcal{M}$ and a variable assignment β *weakly satisfy* a literal L if either all terms in L are defined and the usual notion of satisfaction applies, or if at least one of the terms in L is undefined. Based on weak satisfaction of literals, weak satisfaction of formulas is defined recursively in the usual way. If $\mathcal{M}$ satisfies F for all variable assignments β, $\mathcal{M}$ is a *weak partial model* of F.

For $\Pi = (\Sigma, \mathsf{Pred})$, a total Π-structure $\mathcal{M}$ is a *completion* of a partial Π-structure $\mathcal{M}'$ if $M = M'$ and

1. for every $f \in \Sigma$: $f^{\mathcal{M}}(\overline{x}) = f^{\mathcal{M}'}(\overline{x})$ whenever $f^{\mathcal{M}'}(\overline{x})$ is defined, and
2. for every $P \in \mathsf{Pred}$: $P^{\mathcal{M}} = P^{\mathcal{M}'}$.

For theory extensions of a theory $\mathcal{T}$ with base signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$ with a set of axioms $\mathcal{K}$ with extended signature $\Pi = (\Sigma_0 \cup \Sigma_1, \mathsf{Pred})$, we consider the completability property

($\mathsf{Comp_w}$) For every weak partial Π-model $\mathcal{M}$ of $\mathcal{T} \cup \mathcal{K}$ where Σ_0-functions are total, there exists a completion which is a model of $\mathcal{T} \cup \mathcal{K}$

A formula F is Σ_1-*flat* if it does not contain occurrences of function symbols below a Σ_1-symbol. A Σ_1-flat formula F is Σ_1-*linear* if all extension terms in F which contain the same variable are syntactically equal, and no extension term in F contains two or more occurrences of the same variable.

Theorem 1 (Completability implies extended locality [23]). *If $\mathcal{K}$ consists of Σ_1-linear augmented clauses and the extension of $\mathcal{T}$ with $\mathcal{K}$ satisfies* ($\mathsf{Comp_w}$)*, then it also satisfies* (ELoc)*.*

Combinations and chains of extensions. There are two ways of modularly combining local theory extensions. If we have two extensions of $\mathcal{T}$ with $\mathcal{K}_1$ and $\mathcal{K}_2$, respectively, that introduce disjoint sets of function symbols and individually satisfy ($\mathsf{Comp_w}$), then the extension of $\mathcal{T}$ with $\mathcal{K}_1 \cup \mathcal{K}_2$ also satisfies ($\mathsf{Comp_w}$) [24]. On the other hand, an extended theory can be extended again, so we can extend $\mathcal{T}$ with $\mathcal{K}_1$, then $\mathcal{T} \cup \mathcal{K}_1$ with $\mathcal{K}_2$, and so on, if every extension satisfies ($\mathsf{Comp_w}$).

These combination results allow us to efficiently reason about programs containing multiple user-specified functions: if two functions are defined separately (i.e. if they don't appear in the definition of each other), then the combination of both axioms is also a local extension, and if one is defined in terms of the other, we can use a chain of extensions. There are currently no combination results for mutually recursive functions.

In sections 5 to 8, we introduce several classes of axioms and prove that they satisfy ($\mathsf{Comp_w}$), and thus the locality property (ELoc).

4 Reasoning about Algebraic Data Types with Abstractions

In this section, we introduce a logic that allows us to reason about axiomatic specifications of functional programs. We consider the theory of algebraic data types with parametric element theories and recursive abstraction functions introduced in [28] as our base theory, and use the framework of local theory extensions [23] to reason about axiomatic specifications of additional functions that manipulate these data structures. The syntax of our logic, parametrized by element and collection theory, can be found in Figure 2. In the rest of the paper, we will prove decidability of this logic.

In [28], a method for reasoning about algebraic data types with recursive abstraction functions was introduced. Independently, [25] proved decidability results for recursive functions over absolutely free data structures, based on the notion of local theory extensions. In the following, we give a decision procedure for recursive abstraction functions which is also based on local theory extensions. It decides satisfiability of ground formulas in our logic (see Fig. 2; we ignore additional function symbols $f \notin \Sigma_T$ for now, but they could easily be added as free function symbols). After introducing the new approach, we compare it to the other approaches.

Element terms:	E	$::= e \mid T_E$
Tree terms:	T	$::= t \mid \mathsf{Leaf} \mid \mathsf{Node}(T, E, T) \mid \mathsf{Left}(T) \mid \mathsf{Right}(T)$ $\mid f(T)$ for $f \notin \Sigma_T$
Collection terms:	C	$::= c \mid m(T) \mid T_C$
Element literals:	L_E	$::= \mathcal{L}_E$ (given by $\mathcal{T}_E$)
Tree literals:	L_T	$::= T = T$
Collection literals:	L_C	$::= C = C \mid C \leq C \mid \mathcal{L}_C$ (given by $\mathcal{T}_C$)
Ground Formulas:	φ	$::= L_E \mid L_T \mid L_C \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi$
Axioms:	ϕ	$::= \forall x.\Phi(x, f(x))^\star \mid (\mathsf{Con}) \mid (\mathsf{Mon}) \mid (\mathsf{Inj})$
Conjunctive combinations:	$\phi^\wedge$	$::= \phi \mid \forall x.\Phi(x, f(x)) \wedge (\mathsf{Con})^\star \mid \forall x.\Phi(x, f(x)) \wedge (\mathsf{Inj})^\star$ $\mid \forall x.\Phi(x, f(x)) \wedge (\mathsf{Con}) \wedge (\mathsf{Inj})^\star \mid (\mathsf{Con}) \wedge (\mathsf{Mon})$ $\mid (\mathsf{Con}) \wedge (\mathsf{Inj})$
Piecewise combinations:	$\phi^\vee$	$::= \bigwedge_j \left((\bigwedge_i \phi_j(x_i)) \rightarrow \phi_j^\wedge \right)^\star$
Formulas:	ϕ^+	$::= \phi^\vee \wedge \varphi$

Cases marked with $\star$ require additional side conditions, see Sections 5 to 8.

Fig. 2. Syntax of our logic

We present our decision procedure for the specific algebraic data type of binary trees, but it generalizes to data types with other constructors. Our decision procedure is parametrized by an *element theory* $\mathcal{T}_E$, a *collection theory* $\mathcal{T}_C$ and an *abstraction function* α, which is recursively defined on the tree structure by

$$\mathcal{K}_\alpha = \left\{ \begin{array}{c} \alpha(\mathsf{Leaf}) = \mathsf{empty}, \\ \forall t_1, e, t_2.\ \alpha(\mathsf{Node}(t_1, e, t_2) = \mathsf{combine}(\alpha(t_1), e, \alpha(t_2)) \end{array} \right\},$$

where empty is a ground term in $\mathcal{T}_C$ and $\mathsf{combine}$ is a function which maps two arguments of collection sort and one argument of element sort to a value of collection sort.[1]. We assume that α has the property of *sufficient surjectivity* as defined in [28].

Key steps of the Decision Procedure. We give a decision procedure for conjunctions of literals. It can be lifted to arbitrary ground formulas by using the well-known $DPLL(\mathcal{T})$ approach.

Purification and flattening. We purify the formula s.t. every literal only contains symbols from one of the theories $\mathcal{T}_E, \mathcal{T}_C$ or $\mathcal{T}_T$ (the theory of trees). By our syntax of ground formulas, the only mixed literals are those where α is applied to a tree term. For every such $\alpha(T)$, we introduce a new collection variable c and a tree variable t, add definition literals $t = T$ and $c = \alpha(t)$ to the formula and replace other occurrences $\alpha(T)$ (if any) by c in the rest of the formula.

Then, we flatten tree terms by introduction of fresh tree variables s.t. constructors and selectors are only applied to tree variables (and arbitrary element terms), and tree disequalities do not contain non-variable terms.

[1] Typically, we consider theories of sets or multisets as collection theory, but integers or even booleans can also be considered as "collections".

Elimination of selectors. We rewrite equations $t = \mathsf{Left}(t_1)$ to $t_1 = \mathsf{Node}(t_L, e, t_R) \wedge t = t_L$, and $t = \mathsf{Right}(t_1)$ to $t_1 = \mathsf{Node}(t_L, e, t_R) \wedge t = t_R$, where t_L, t_R and e are always fresh variables.
Unification of tree literals. We apply unification on the positive tree literals (see [28] for details). If unification fails, we have shown unsatisfiability. Otherwise, we obtain a solution σ that can be seen as a substitution for tree and element variables. We apply this substitution to the formula and call the result G. Tree variables t with $\sigma(t) = t$ will in the following be called *parameter variables*. If for any disequality $x \neq y$ between tree or element variables we have $\sigma(x) = \sigma(y)$, then we also have shown unsatisfiability. Otherwise, we continue.
Adding additional constraint P**.** To ensure equisatisfiability with the original problem, we now have to add an additional constraint P that depends on the abstraction function α, as well as on the variables and terms of tree sort in the given formula. For details we refer again to [28].
Partial evaluation of α**.** We partially evaluate α in $G \wedge P$ by adding, for every term $\alpha(\mathsf{Node}(T_1, e, T_2))$, the recursive definition

$$\alpha(\mathsf{Node}(T_1, e, T_2)) = \mathsf{combine}(\alpha(T_1), e, \alpha(T_2)),$$

and recursively for the subterms T_1 and T_2. The set of all these recursive definitions, including $\alpha(\mathsf{Leaf}) = \mathsf{empty}$, will be called $\mathcal{K}_\alpha[G]$ (note that all terms in P which appear below α are also in G, so the set of needed definitions does not depend on P). G may still contain equalities of the form $\alpha(T_j) = c_j$, where T_j may be either a parameter variable or a term containing parameter variables. In the latter case, $\mathcal{K}_\alpha[G]$ contains all recursive definitions to uniquely determine $\alpha(T_j)$, given the values $\alpha(t_i)$ for all parameter variables t_i and values of element variables e_i appearing in T_j.
Solving the resulting problem. The resulting formula, when considering α as a free function symbol, is equisatisfiable to the original one. We can check its satisfiability with a combined decision procedure for $\mathcal{T}_T \cup \mathcal{T}_c \cup \mathcal{T}_E$ (plus free function symbols). Since P has a significant propositional structure, we will in general need to employ the base decision procedure in a DPLL($\mathcal{T}$) framework.

Correctness of the decision procedure. It is clear that the preprocessing steps are satisfiability-preserving. Therefore, we only state that the constraint G (in the theory $\mathcal{T}_T \cup \mathcal{T}_c \cup \mathcal{T}_E \cup \mathcal{K}_\alpha$, where $\mathcal{K}_\alpha$ are the universal recursive definitions of α) is equisatisfiable to the resulting formula $G \wedge P \wedge \mathcal{K}_\alpha[G]$ in $\mathcal{T}_T \cup \mathcal{T}_c \cup \mathcal{T}_E$.

Theorem 2 (Correctness of Decision Procedure for Abstraction Functions). *Let G be a conjunction of literals which has been preprocessed as mentioned above, let P be as defined in [28] (based on G and α), and $\mathcal{K}_\alpha$ and $\mathcal{K}_\alpha[G]$ as defined above. Then*

$$G \models_{\mathcal{T}_T \cup \mathcal{T}_c \cup \mathcal{T}_E \cup \mathcal{K}_\alpha} \square \iff G \cup P \cup \mathcal{K}_\alpha[G] \models_{\mathcal{T}_T \cup \mathcal{T}_c \cup \mathcal{T}_E} \square.$$

Comparison to previous approaches. Compared to the approach from [25], the preprocessing steps that ensure completeness of the following quantifier instantiation are simpler and described in detail, and we can handle a class of

functions over algebraic data types, using the sufficient surjectivity condition and the according additional constraints from [28] (the problem of adding "counting constraints" for such functions was left open in [25]).

Compared to [28], we improved efficiency of the decision procedure by removing the case split on all possible equalities/disequalities of tree and element variables, which are now partially determined by unification, and then negotiated by a Nelson-Oppen combination of decision procedures for the base theory. Additionally, we do not need the DNF conversion of formula P, since our correctness argument works for arbitrary boolean structure of P.

5 Complete Reasoning about Single-Invocation Axioms

We next present a decision procedure for the following fundamental problem. Suppose we are given a function f, and we wish to prove that it satisfies some quantifier-free condition $P(f(\overline{t}_1, \ldots, \overline{t}_n))$. In the condition P function f is applied to arbitrary terms $\overline{t}_1, \ldots, \overline{t}_n$ from some decidable theory, such as theory of linear arithmetic, lists, sets, or trees. We wish to prove P using only a contract-like specification of f as an assumption. Consider a contract for f with a quantifier-free precondition $\mathsf{Pre}(\overline{x})$ and a quantifier-free postcondition $\mathsf{Post}(\overline{x}, r)$, with r denoting the resulting value. We model such contract as the formula $\forall \overline{x}.\Phi(\overline{x}, f(\overline{x}))$, where $\Phi(\overline{x}, r)$ is $\mathsf{Pre}(\overline{x}) \rightarrow \mathsf{Post}(\overline{x}, r)$. Our goal is to show that the contract implies the desired property, that is, that the following formula is valid: $(\forall \overline{x}.\Phi(\overline{x}, f(\overline{x}))) \rightarrow P(f(\overline{t}_1, \ldots, \overline{t}_n))$, Equivalently, we aim to show that $(\forall \overline{x}.\Phi(\overline{x}, f(\overline{x}))) \wedge \neg P(f(\overline{t}_1, \ldots, \overline{t}_n))$ is an unsatisfiable formula. In this section we show that we can reduce such satisfiability problems to the simpler *quantifier-free* satisfiability problem $\left(\bigwedge_{i=1}^{n} \Phi(\overline{t}_i, f(\overline{t}_i))\right) \wedge \neg P(f(\overline{t}_1, \ldots, \overline{t}_n))$, in which the universal quantifier is instantiated only with the terms t_i. Note that the above instantiation confirms two important special cases. First, when $\Phi(\overline{x}, r)$ specifies the behavior of f completely, that is, r is unique for a given $\boldsymbol{x}$, then the axiom is simply a form of one-point rule applied to f as a variable. Second, if we view P as a program that invokes f, then the instantiation principle above reflects modular reasoning by inlining contracts. While it is known that such reasoning is sound, this is usually shown using operational semantics. In our formulation, the approach is sound thanks to quantifier instantiation. Moreover, the next theorem shows that it is also complete, as a consequence of a locality result.

Theorem 3 (Locality for Single-Invocation Axioms). *Let $\mathcal{T}$ be a theory with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$, f a fresh function symbol and $\Phi(\overline{x}, f(\overline{x}))$ a $(\Sigma_0 \cup \{f\}, \mathsf{Pred})$-formula with $\overline{x}$ as the vector of all free variables and $f(\overline{x})$ the only term in which free variables appear below f.*

The extension of $\mathcal{T}$ with $\forall \overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$ satisfies ($\mathsf{Comp_w}$) if and only if $\models_{\mathcal{T}} \forall \overline{x} \exists y.\ \Phi(\overline{x}, y)$.

By Theorem 1, condition ($\mathsf{Comp_w}$) implies (ELoc), which ensures that we can decide satisfiability of formulas with respect to such axioms by simply adding

one instance of the axiom for every occurrence of the function symbol in a given formula.

Note that the side condition $\models_{\mathcal{T}} \forall\overline{x}\exists y.\ \Phi(\overline{x}, y)$ simply guarantees that the specification is consistent. The condition holds whenever there exists a function f with $\forall\overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$. Consequently, if we have proved that some function satisfies its contract, we can use finite instantiation to reason about the contract in a complete way.

Let us also remark that Theorem 3 subsumes the decidability result from field constraint analysis [30] in a more systematic form, by expressing it as a locality result. We have shown that this result applies also to contracts of functional programs. Moreover, in addition to better understanding, locality gives us efficient implementations in the form of incremental instance generation [15].

Loss of locality for general many-invocation axioms. In general, the straightforward modification of Theorem 3 for axioms with multiple function invocations does not hold. Consider the background theory of integers $\mathcal{T}_{\mathbb{Z}}$ and its extension with strict monotonicity

$$\forall x_1, x_2.\ x_1 < x_2 \ \rightarrow\ f(x_1) < f(x_2). \qquad (\mathsf{SMon})$$

Even though the axiom is consistent, the extension of $\mathcal{T}_{\mathbb{Z}}$ with (SMon) is not local (and thus cannot satisfy ($\mathsf{Comp_w}$)). Indeed, take $G = \{f(0) = 0, f(2) = 1\}$. Then the instantiation (SMon)$[G] \cup G$ is $\mathcal{T}_{\mathbb{Z}}$-satisfiable, but (SMon)$\cup G$ is unsatisfiable. because a strictly monotonic function on integers cannot have $f(0) = 0$ and $f(2) = 1$. Despite this negative result, in the next sections we show that in a number of cases of interest we can also obtain locality results for many-invocation axioms.

6 Complete Reasoning about Many-Invocation Axioms

In this section, we push decidability of reasoning over axiomatic specifications beyond function contracts to more expressive axiom classes.

Congruence. We want to consider axiomatic specifications which go beyond contracts, in that they allow multiple function invocations. We start with congruence properties, like e.g. defined for function even in Figure 1.

We consider equivalence relations on data types given by abstraction functions like $\mathsf{contents}$:

$$x \sim y \iff \mathsf{contents}(x) = \mathsf{contents}(y).$$

The following result allows us to reason about specifications ensuring congruence of a function wrt. an equivalence relation $\sim$. In the following Theorem, the equivalence relation $\dot{\sim}$ may be defined by $\overline{x}\dot{\sim}\overline{y} \iff \bigwedge_{i=1}^{n} x_i \sim y_i$, but it can also be any other equivalence relation on the domain of f.

Theorem 4 (Locality of Congruence). *If $\mathcal{T}$ is a theory with equivalence relations $\sim$ and $\dot{\sim}$, then the extension of $\mathcal{T}$ with a function f satisfying*

$$\overline{x}\dot{\sim}\overline{y} \ \rightarrow\ f(\overline{x}) \sim f(\overline{y}) \qquad (\mathsf{Con})$$

satisfies ($\mathsf{Comp_w}$).

This Theorem allows us to decide satisfiability problems with user-defined functions that are specified to be congruent wrt. a given abstraction.

Monotonicity. Another important requirement for many functions is monotonicity wrt. a given abstraction, like e.g. specified for function insert in Figure 1.

By ordering data structures wrt. the given abstraction, we define a new order

$$x \preceq y \iff \mathsf{contents}(x) \subseteq \mathsf{contents}(y).$$

Because it is defined by a function, such a user-defined order has all defining properties of the original order, except antisymmetry (unless the abstraction function is injective). That is, if the original order is a total order then the user-defined order will be a total preorder, if it is a lattice the user-defined order will be a prelattice, etc.

The following theorem extends known results on local extensions with monotone functions [16, 27] to the case of preorders and bounded (semi-)prelattices:

Theorem 5 (Locality of Monotonicity). *Let $\mathcal{T}$ be a theory with a binary relation $\preceq$ such that either (i) $\preceq$ is a total preorder, or (ii) $\preceq$ defines a bounded semi-prelattice. Let furthermore $R(x, y)$ be a binary predicate which is transitive in $\mathcal{T}$. Then the extension of $\mathcal{T}$ with a function f satisfying*

$$R(\overline{x}, \overline{y}) \rightarrow f(\overline{x}) \preceq f(\overline{y}) \qquad (\mathsf{Mon})$$

satisfies ($\mathsf{Comp_w}$).

For the abstraction of data structures to their set of elements, $\preceq$ is a bounded prelattice, and for abstractions to the length or size of a data structure, it is a total preorder.

Injectivity. Another important property of functions manipulating data structures is injectivity. The following result allows us to decide specifications that require injectivity of a function wrt. a given abstraction, and is a generalization of known locality results wrt. equality [13]:

Theorem 6 (Locality of Injectivity). *If $\mathcal{T}$ is a theory with equivalence relations $\sim$ and $\dot{\sim}$ such that (in every model of $\mathcal{T}$)[2] there are infinitely many equivalence classes wrt. $\sim$, then the extension of $\mathcal{T}$ with a function f satisfying*

$$\neg(\overline{x} \dot{\sim} \overline{y}) \rightarrow f(\overline{x}) \not\sim f(\overline{y}) \qquad (\mathsf{Inj})$$

satisfies ($\mathsf{Comp_{w,f}}$).[3]

[2] In fact, it would be sufficient to have a notion similar to *stable infiniteness* wrt. the equivalence classes, i.e. whenever a formula is satisfiable, it is also satisfiable in a model with infinitely many equivalence classes.

[3] ($\mathsf{Comp_{w,f}}$) is slightly weaker than ($\mathsf{Comp_w}$) in that it only considers embeddability of models where partial functions have a finite domain. It implies the slightly weakened locality condition ($\mathsf{ELoc_f}$), which requires G to be finite.

7 Complete Reasoning about Conjunctive Combinations

In this Section, we consider the problem of reasoning about specifications which impose several axioms from the classes introduced in Sections 5 and 6 on the same function, like e.g. the function even from Figure 1.

We start with combinations of the axioms in Section 6, and then look at possibilities to combine these with single-invocation axioms.

Theorem 7 (Locality of $(\mathsf{Con}) \cup (\mathsf{Mon})$**).** *Let $\mathcal{T}$ be a theory with equivalence relations $\sim$ and $\dot{\sim}$, and a binary relation $\preceq$ s.t. $\models_{\mathcal{T}} x \preceq y \wedge y \preceq x \iff x \sim y$, and either (i) $\preceq$ is a total preorder, or (ii) $\preceq$ defines a bounded semi-prelattice. Let furthermore $R(\overline{x}, \overline{y})$ be a binary predicate which is transitive in $\mathcal{T}$. Then the extension of $\mathcal{T}$ with a function f satisfying* $(\mathsf{Con}) \cup (\mathsf{Mon})$ *satisfies* $(\mathsf{Comp_w})$*.*

Theorem 8 (Locality of $(\mathsf{Con}) \cup (\mathsf{Inj})$**).** *Let $\mathcal{T}$ be a theory with equivalence relations $\sim$ and $\dot{\sim}$. Then the extension of $\mathcal{T}$ with a function f satisfying* $(\mathsf{Con}) \cup (\mathsf{Inj})$ *satisfies* $(\mathsf{Comp_w})$ *if and only if $|dom(f)/\dot{\sim}| \leq |codom(f)/\sim|$.*

In contrast to these two positive results, an extension with $(\mathsf{Mon}) \cup (\mathsf{Inj})$ or $(\mathsf{Con}) \cup (\mathsf{Mon}) \cup (\mathsf{Inj})$ will in general not be local. This essentially corresponds to strict monotonicity, mentioned as a counterexample to locality at the end of Section 5. We can however obtain some positive results for combinations of single-invocation axioms with axioms from Section 6:

Theorem 9 (Locality of $(\mathsf{Con}) \cup \Phi(\overline{x}, f(\overline{x}))$**).** *Let $\mathcal{T}$ be a theory with equivalence relations $\sim$ and $\dot{\sim}$, and $\Phi(x, f(x))$ as in Theorem 3. Then the extension of $\mathcal{T}$ with a function f satisfying* $(\mathsf{Con}) \cup \forall \overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$ *satisfies* $(\mathsf{Comp_w})$ *if and only if $\models_{\mathcal{T}} \forall \overline{x_1}, \overline{x_2}, y_1.\ (\overline{x_1} \dot{\sim} \overline{x_2} \wedge \Phi(\overline{x_1}, y_1) \rightarrow \exists y_2.\ y_1 \sim y_2 \wedge \Phi(\overline{x_2}, y_2))$.*

Theorem 10 (Locality of $(\mathsf{Inj}) \cup \Phi(\overline{x}, f(\overline{x}))$**).** *Let $\mathcal{T}$ be a theory with equivalence relations $\sim$ and $\dot{\sim}$, and $\Phi(\overline{x}, f(\overline{x}))$ as in Theorem 3. Then the extension of $\mathcal{T}$ with a function f satisfying* $(\mathsf{Inj}) \cup \forall \overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$ *satisfies* $(\mathsf{Comp_w})$ *if $\models_{\mathcal{T}} \forall \overline{x_1}, \overline{x_2}, y_1.\ (\neg(\overline{x_1} \dot{\sim} \overline{x_2}) \wedge \Phi(\overline{x_1}, y_1) \rightarrow \exists^{\infty} y_2.\ y_1 \not\sim y_2 \wedge \Phi(\overline{x_2}, y_2))$.*

Above, $\exists^{\infty} y_2$ means that there must exist infinitely many values y_2 in different equivalence classes wrt. $\sim$. In contrast to Theorem 9, the condition is sufficient, but not necessary.

Theorem 11 (Locality of $(\mathsf{Con}) \cup (\mathsf{Inj}) \cup \Phi(\overline{x}, f(\overline{x}))$**).** *Let $\mathcal{T}$ be a theory with equivalence relations $\sim$ and $\dot{\sim}$, and $\Phi(\overline{x}, f(\overline{x}))$ as in Theorem 3. The extension of $\mathcal{T}$ with a function f satisfying* $(\mathsf{Con}) \cup (\mathsf{Inj}) \cup \forall x.\ \Phi(x, f(x))$ *satisfies* $(\mathsf{Comp_w})$ *if and only if $\models_{\mathcal{T}} \forall \overline{x_1}, \overline{x_2}, y_1.\ (\overline{x_1} \dot{\sim} \overline{x_2} \wedge \Phi(\overline{x_1}, y_1) \rightarrow \exists y_2.\ y_1 \sim y_2 \wedge \Phi(\overline{x_2}, y_2))$ and $|dom(f)/\dot{\sim}| \leq |codom(f)/\sim|$.*

For the combination of contracts with monotonicity we do not have conclusive results. The combination is in general not local, not even with the restriction to contracts which allow monotone functions. It seems to be hard to find general conditions under which such a combination is local.

8 Complete Reasoning about Piecewise Combinations

In some cases, it is desirable to use specifications with a case distinction over different partitions of the function domain, like e.g. for function fill in Figure 1.

While single-invocation axioms allow case distinctions intrinsically, this is not the case for the other axiom classes we introduced, or local theory extensions in general. In the following, we consider sets of axioms $\mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$, where in each axiom there may be up to n invocations of f. We first show that we can have several different local axiomatizations in disjoint subsets of the domain, and the resulting "piecewise" axiomatization will again be local. Furthermore, we show that a piecewise local axiomatization can also be obtained if local axiomatizations are given for non-disjoint subsets of the domain, as long as the overlaps are finite.

In the following, we use restrictions of formulas $\mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ to a subset of the domain of f, specified by a formula $\Phi(\overline{x})$. We denote by $\bigwedge_{i=1}^{n} \Phi(\overline{x}_i) \rightarrow \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ the set of augmented Π-clauses $\{\forall \overline{x}_1, \ldots, \overline{x}_n.\ \bigwedge_{i=1}^{n} \Phi(\overline{x}_i) \rightarrow F \vee C \mid F \vee C \in \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))\}$. We state that such restrictions do not destroy locality properties:

Lemma 1. *Let $\mathcal{T}$ be a Π_0-theory and $\mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ a set of augmented Π-clauses such that $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ satisfies* $(\mathsf{Comp_w})$*. For any Π_0-formula $\Phi(\overline{x})$, the extension $\mathcal{T} \subseteq \mathcal{T} \cup (\Phi(\overline{x}) \rightarrow \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ also satisfies* $(\mathsf{Comp_w})$*.*

Together with this lemma, the following theorem allows piecewise combinations of axioms satisfying $(\mathsf{Comp_w})$:

Theorem 12 (Locality of Disjoint Piecewise Combinations). *Let $\mathcal{T}$ be a Π_0-theory and consider Π_0-formulas $\Phi_1(\overline{x}), \Phi_2(\overline{x})$ such that $\models_{\mathcal{T}} \neg(\Phi_1(\overline{x}) \wedge \Phi_2(\overline{x}))$. If $\mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $\mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ are Π-formulas such that both $\mathcal{T} \subseteq \mathcal{T} \cup (\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $\mathcal{T} \subseteq \mathcal{T} \cup (\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ satisfy condition* $(\mathsf{Comp_w})$*, then for*

$$\begin{aligned} \mathcal{K} = \quad & (\textstyle\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i)) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n)) \\ \cup\ & (\textstyle\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i)) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m)), \end{aligned}$$

the extension $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}$ satisfies $(\mathsf{Comp_w})$ *and is a local extension.*

Repeated application of Theorem 12 directly gives a locality result for arbitrarily many case distinctions. If the subsets defined by the Φ_i are not disjoint, we can still obtain a decision procedure in some cases:

Non-disjoint subsets with a finite intersection. If the overlap between cases is finite, we can preserve completeness by instantiating the axioms additionally for all elements in the overlap. To this end, consider a closure operator Ψ on ground terms and the more general notion of Ψ-completability

$(\mathsf{Comp_w^{\Psi}})$ For every weak partial Π-model $\mathcal{M}$ of $\mathcal{T} \cup \mathcal{K}$ where Σ_0-functions are total and the definition domain of Σ_1-functions is closed under Ψ, there exists a completion which is a model of $\mathcal{T} \cup \mathcal{K}$,

which implies the Ψ-locality condition

(ELoc^{Ψ}) For every set G of Σ_1-ground augmented Π-clauses, we have $\mathcal{K} \cup G \models_{\mathcal{T}} \square \iff \mathcal{K}^{\Psi}[G] \cup G \models_{\mathcal{T}} \square$

with $\mathcal{K}^{\Psi}[G]$ defined like $\mathcal{K}[G]$, except extension terms may be in $\Psi(\mathsf{st}(\mathcal{K} \cup G))$.

Then, with a suitable Ψ we can prove Ψ-locality for piecewise combinations with finite overlaps:

Theorem 13 (Locality of Piecewise Combinations with Finite Intersection). *Let $\mathcal{T}$ be a theory with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$ and consider Π_0-formulas $\Phi_1(\overline{x}), \Phi_2(\overline{x})$ such that in every $\mathcal{T}$-model $\mathcal{M}$, the set $O = \{\overline{x} \in M | \Phi_1(\overline{x}) \wedge \Phi_2(\overline{x})\}$ is finite. Let furthermore T_0 be a set of Σ_0-terms such that in every such model $\mathcal{M}$, $O \subseteq \{t^{\mathcal{M}} \mid t \in T_0\}$.*

If $\mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $\mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ are sets of augmented Π-clauses such that both the extensions of $\mathcal{T}$ with $(\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $(\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ satisfy ($\mathsf{Comp_w}$), then for

$$\begin{array}{rl} \mathcal{K} = & (\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i)) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n)) \\ & \cup\, (\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i)) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m)) \end{array}$$

and $\Psi(T) = T \cup \{f(t) | t \in T_0\}$, the extension of $\mathcal{T}$ with $\mathcal{K}$ satisfies ($\mathsf{Comp_w}^{\Psi}$).

Theorem 13 can easily be extended to a combination of arbitrarily many pieces, where $O = \{\overline{x} \in M | \Phi_i(\overline{x}) \wedge \Phi_j(\overline{x})$, for some $i \neq j\}$ (and T_0 and Ψ change accordingly).

9 Related Work

The notion of *local theory extensions* was introduced by Sofronie-Stokkermans [23]. The approach is based on earlier results by Givan and McAllester [11] and Ganzinger [8]. Local theory extensions are one of the few approaches that allow us to obtain decision procedures for quantified satisfiability problems modulo a background theory. They have been shown to be useful in the verification of parametrized systems [7, 17, 26] and properties of data structures [13, 25], in reasoning about certain properties of numerical functions [24] and about certain properties of functions in ordered domains [27]. An overview of a large part of the results on local theory extensions can be found in [16]. In addition, there has been other research on handling quantified formulas in a complete way, by identifying decidable fragments of the theory of arrays [3, 10] or of pointer data structures [20], or certain forms of formulas which ensure decidability with the right instantiation strategy [9].

Note that each locality result needs to be proved separately for each class of axioms of interest. A contribution of our paper is to identify new classes of local theories, and to show that they are useful in verification of new classes of properties of functional programs.

References

1. M. Barnett, K. R. M. Leino, and W. Schulte. The Spec# programming system: An overview. In *CASSIS: Int. Workshop on Construction and Analysis of Safe, Secure and Interoperable Smart devices*, 2004.
2. C. Barrett and C. Tinelli. CVC3. In *CAV*, volume 4590 of *LNCS*, 2007.
3. A. R. Bradley, Z. Manna, and H. B. Sipma. What's decidable about arrays? In E. A. Emerson and K. S. Namjoshi, editors, *Verification, Model-Checking, and Abstract-Interpretation, VMCAI'06*, volume 3855 of *LNCS*, pages 427–442. Springer, 2006.
4. E. Cohen, M. Dahlweid, M. Hillebrand, D. Leinenbach, M. Moskal, T. Santen, W. Schulte, and S. Tobies. Vcc: A practical system for verifying concurrent c. In *Conf. Theorem Proving in Higher Order Logics (TPHOLs)*, volume 5674 of *LNCS*, 2009.
5. D. R. Cok. Reasoning with specifications containing method calls and model fields. *Journal of Object Technology*, 4(8):77–103, September–October 2005.
6. L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *TACAS*, pages 337–340, 2008.
7. J. Faber, S. Jacobs, and V. Sofronie-Stokkermans. Verifying CSP-OZ-DC specifications with complex data types and timing parameters. In J. D. J. Gibbons, editor, *Integrated Formal Methods, IFM'07*, volume 4591 of *LNCS*, pages 233–252. Springer, 2007.
8. H. 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.
9. Y. Ge and L. de Moura. Complete instantiation for quantified SMT formulas. In A. Bouajjani and O. Maler, editors, *Computer Aided Verification, CAV'09*, volume 5643 of *LNCS*, pages 306–320. Springer, 2009.
10. S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Decision procedures for extensions of the theory of arrays. *Annals of Mathematics and Artificial Intelligence*, 50(3-4):231–254, 2007.
11. R. Givan and D. A. McAllester. Polynomial-time computation via local inference relations. *ACM Transactions on Computational Logic*, 3(4):521–541, 2002.
12. J. Guttag and J. Horning. *Larch: Languages and Tools for Formal Specification*. Springer-Verlag, 1993.
13. C. Ihlemann, S. Jacobs, and V. Sofronie-Stokkermans. On local reasoning in verification. In C. R. Ramakrishnan and J. 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.
14. C. Ihlemann and V. Sofronie-Stokkermans. On hierarchical reasoning in combinations of theories. In *International Joint Conference on Automated Reasoning, IJCAR'10*, 2010. To appear.
15. S. Jacobs. Incremental instance generation in local reasoning. In A. Bouajjani and O. Maler, editors, *Computer Aided Verification, CAV'09*, volume 5643 of *LNCS*, pages 368–382. Springer, 2009.
16. S. Jacobs. *Hierarchic Decision Procedures for Verification*. PhD thesis, Saarland University, Germany, 2010.
17. S. Jacobs and V. Sofronie-Stokkermans. Applications of hierarchical reasoning in the verification of complex systems. *Electronic Notes in Theoretical Computer Science*, 174(8):39–54, 2007.

18. V. Kuncak. *Modular Data Structure Verification*. PhD thesis, EECS Department, Massachusetts Institute of Technology, February 2007.
19. P. Lam, V. Kuncak, and M. Rinard. Generalized typestate checking for data structure consistency. In *6th Int. Conf. Verification, Model Checking and Abstract Interpretation*, 2005.
20. S. McPeak and G. C. Necula. Data structure specifications via local equality axioms. In K. Etessami and S. K. Rajamani, editors, *Computer Aided Verification, CAV'05*, volume 3576 of *LNCS*, pages 476–490. Springer, 2005.
21. A. Podelski and T. Wies. Counterexample-guided focus. In *POPL*, 2010.
22. G. Ramalingam, A. Warshavsky, J. Field, D. Goyal, and M. Sagiv. Deriving specialized program analyses for certifying component-client conformance. In *PLDI*, 2002.
23. V. Sofronie-Stokkermans. Hierarchic reasoning in local theory extensions. In R. Nieuwenhuis, editor, *Conference on Automated Deduction, CADE-20*, volume 3632 of *LNAI*, pages 219–234. Springer, 2005.
24. V. Sofronie-Stokkermans. Efficient hierarchical reasoning about functions over numerical domains. In K. Berns and T. Breuel, editors, *KI 2008: Advances in Artificial Intelligence*, volume 5243 of *Lecture Notes in Artificial Intelligence*, pages 135–143, Kaiserslautern, Germany, 2008. Springer.
25. V. Sofronie-Stokkermans. Locality results for certain extensions of theories with bridging functions. In *Conference on Automated Deduction, CADE-22*, pages 67–83. Springer, 2009.
26. V. Sofronie-Stokkermans. Hierarchical reasoning for the verification of parametric systems. In *International Joint Conference on Automated Reasoning, IJCAR'10*, 2010. To appear.
27. V. Sofronie-Stokkermans and C. Ihlemann. Automated reasoning in some local extensions of ordered structures. *Journal of Multiple-Valued Logic and Soft Computing*, 13(4-6):397–414, 2007.
28. P. Suter, M. Dotta, and V. Kuncak. Decision procedures for algebraic data types with abstractions. In *37th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL)*, 2010.
29. T. Wies, V. Kuncak, P. Lam, A. Podelski, and M. Rinard. Field constraint analysis. In *Proc. Int. Conf. Verification, Model Checking, and Abstract Interpratation*, 2006.
30. T. Wies, V. Kuncak, P. Lam, A. Podelski, and M. Rinard. Field constraint analysis. In *Verification, Model Checking, and Abstract Interpratation, VMCAI'06*, volume 3855 of *LNCS*, 2006.
31. K. Zee, V. Kuncak, and M. Rinard. Full functional verification of linked data structures. In *ACM Conf. Programming Language Design and Implementation (PLDI)*, 2008.

A Theorems with Proofs

Theorem 2 (Correctness of Decision Procedure for Abstraction Functions). *Let G be a conjunction of literals which has been preprocessed as mentioned above, let P be as defined in [28] (based on G and α), and $\mathcal{K}_\alpha$ and $\mathcal{K}_\alpha[G]$ as defined above. Then*

$$G \models_{\mathcal{T}_T \cup \mathcal{T}_c \cup \mathcal{T}_E \cup \mathcal{K}_\alpha} \bot \iff G \cup P \cup \mathcal{K}_\alpha[G] \models_{\mathcal{T}_T \cup \mathcal{T}_C \cup \mathcal{T}_E} \bot$$

Proof. ($\Rightarrow$) Suppose $\mathcal{M}$ is a model of $G \cup \mathcal{T}_T \cup \mathcal{T}_c \cup \mathcal{T}_E \cup \mathcal{K}_\alpha$. Then certainly $\mathcal{M} \models G \wedge \mathcal{K}_\alpha[G]$. By sufficient surjectivity of α and construction of P, P is valid in any model of $G \cup \mathcal{T}_T \cup \mathcal{T}_c \cup \mathcal{T}_E \cup \mathcal{K}_\alpha$, so in particular $\mathcal{M} \models P$.

($\Leftarrow$) Now suppose $\mathcal{M}$ is a model of $G \cup P \cup \mathcal{K}_\alpha[G] \cup \mathcal{T}_T \cup \mathcal{T}_c \cup \mathcal{T}_E$. We will show that based on $\mathcal{M}$ we can find a model for $G \cup \mathcal{T}_T \cup \mathcal{T}_c \cup \mathcal{T}_E \cup \mathcal{K}_\alpha$. Since $\mathcal{M}$ is based only on a partial evaluation of α, the valutions of α and of the tree, element and collection variables in $\mathcal{M}$ (depicted as $\alpha_\mathcal{M}, t_{i_\mathcal{M}}, e_{k_\mathcal{M}}, c_{j_\mathcal{M}}$ in the following) may not be consistent with $\mathcal{K}_\alpha$. Thus, consider the model $\mathcal{M}'$ where

- $e_{k_{\mathcal{M}'}} = e_{k_\mathcal{M}}$
- $c_{j_{\mathcal{M}'}} = c_{j_\mathcal{M}}$
- $\alpha_{\mathcal{M}'}$ is defined recursively on all trees according to $\mathcal{K}_\alpha$

It remains to find valuations of the t_i in $\mathcal{M}'$ such that all disequalities between tree terms in G are preserved (there are no equalities containing t_i because G is saturated under unification, and terms in P are either taken from G or ground terms up to element variables), and all constraints $\alpha(T_j) = c_j$ in G hold, where T_j is a parameter variable t_i or a term defined using the t_i.

By construction of P, for every parameter variable t_i, $\mathcal{M}$ must satisfy either $M_p(\alpha(t_i))$ or $inst(s, i) = t_i$ (see [28]) for some $s \in S_p$. In the second case, its valuation in $\mathcal{M}'$ is already fixed (because of the fixed shape and the valuations of element variables taken from $\mathcal{M}$). Otherwise we have $M_p(\alpha(t_i))$, which by sufficient surjectivity implies $|\alpha^{-1}(\alpha(t_i))| > p$, i.e. for every such t_i we can pick from at least $p + 1$ values v_l which all satisfy $\alpha(v_l) = c_i$. Since G contains at most p disequalities, by Lemma 2 from [28], we can find a satisfying valuation which preserves both the disequalities and the constraints $\alpha(t_i) = c_i$ in G.

Finally, since for every term T_j containing parameter variables, $\mathcal{M}$ satisfies the recursive definition of α in $\mathcal{K}_\alpha[G]$, the valuations (in $\mathcal{M}$) of collection terms c_j in equalities $\alpha(T_j) = c_j$ must be consistent with the recursive definition (for arbitrary valuations of the t_i and e_i). Thus, all constraints $\alpha(T_j) = c_j$ for non-variable terms will also be satisfied in this model. □

Theorem 3 (Locality for Single-Invocation Axioms). *Let $\mathcal{T}$ be a theory with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$, f a fresh function symbol and $\Phi(\overline{x}, f(\overline{x}))$ a $(\Sigma_0 \cup \{f\}, \mathsf{Pred})$-formula with $\overline{x}$ as the vector of all free variables and $f(\overline{x})$ the only term in which free variables appear below f.*

The extension of $\mathcal{T}$ with $\forall \overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$ satisfies ($\mathsf{Comp_w}$) if and only if $\models_\mathcal{T} \forall \overline{x} \exists y.\ \Phi(\overline{x}, y)$.

Proof. First, assume $\models_{\mathcal{T}} \forall\overline{x}\exists y.\ \Phi(\overline{x}, y)$. We prove that then $\mathcal{T} \subseteq \mathcal{T} \cup \forall\overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$ satisfies ($\mathsf{Comp_w}$): Let $\mathcal{M}$ be a partial model of $\mathcal{T} \cup \forall\overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$, with $f^{\mathcal{M}}(a_i)$ defined for finitely many values $a_1, \ldots, a_n$. Consider the structure $\mathcal{M}'$ obtained from $\mathcal{M}$ by letting

$$f^{\mathcal{M}'}(\overline{x}) = \begin{cases} f^{\mathcal{M}}(\overline{x}), & \text{if } \overline{x} = a_i \\ y, \text{ for some } y \text{ with } \mathcal{M} \models \Phi(\overline{x}, y), & \text{else} \end{cases}$$

Since $\models_{\mathcal{T}} \forall\overline{x}\exists y.\ \Phi(\overline{x}, y)$, such a value y always exists (in every model of $\mathcal{T}$). Clearly, the resulting structure is a model of $\mathcal{T} \cup \forall\overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$, i.e. ($\mathsf{Comp_w}$) is satisfied.

Now, assume that $\mathcal{T} \subseteq \mathcal{T} \cup \forall\overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$ satisfies ($\mathsf{Comp_w}$). Since every weak partial model of $\mathcal{T} \cup \forall\overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$ can be completed to a total model of $\mathcal{T} \cup \forall\overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$, we know that for every $\overline{x}$ there is a value $f(\overline{x})$ which satisfies $\Phi(\overline{x}, f(\overline{x}))$. In particular, we have $\models_{\mathcal{T}} \forall\overline{x}\exists y.\ \Phi(\overline{x}, y)$. □

Theorem 4 (Locality of Congruence). *If $\mathcal{T}$ is a theory with equivalence relations $\sim$ and $\dot{\sim}$, then the extension of $\mathcal{T}$ with a function f satisfying*

$$\overline{x}\dot{\sim}\overline{y} \ \rightarrow\ f(\overline{x}) \sim f(\overline{y}) \qquad (\mathsf{Con})$$

satisfies ($\mathsf{Comp_w}$).

Proof idea: In any partial model $\mathcal{M}$, defined values of $f_{\mathcal{M}}$ must respect congruence. We can complete $f_{\mathcal{M}}$ to a total function by mapping values from equivalence classes (of $\dot{\sim}$) which contain a value a with $f_{\mathcal{M}}(a)$ defined to $f_{\mathcal{M}}(a)$, and mapping all other values to an arbitrary value b. □

Theorem 5 (Locality of Monotonicity). *Let $\mathcal{T}$ be a theory with a binary relation $\preceq$ such that either (i) $\preceq$ is a total preorder, or (ii) $\preceq$ defines a bounded semi-prelattice. Let furthermore $R(x, y)$ be a binary predicate which is transitive in $\mathcal{T}$. Then the extension of $\mathcal{T}$ with a function f satsfying*

$$R(\overline{x}, \overline{y}) \ \rightarrow\ f(\overline{x}) \preceq f(\overline{y}) \qquad (\mathsf{Mon})$$

satisfies ($\mathsf{Comp_w}$).

Proof idea: The proof for quasi-monotone functions from [16] can be modified to work without antisymmetry of the relation $\preceq$ in the following way: in a partial model $\mathcal{M}$ of $\mathcal{T} \cup (\mathsf{Mon})$, consider equivalence classes of elements with $(x \preceq_{\mathcal{M}} y \wedge y \preceq_{\mathcal{M}} x)$, and replace every such class with one representative a. In the resulting structure, $\preceq$ has the antisymmetry property, and we can use the completion from the original proof. After that, replace representatives again with their equivalence classes and define the completion of $f_{\mathcal{M}}$ for elements of a class as for their representative (unless it was already defined before). In the resulting structure $\mathcal{M}'$, $f_{\mathcal{M}'}$ is total and satisfies $\mathcal{T} \cup (\mathsf{Mon})$. □

Theorem 6 (Locality of Injectivity). *If $\mathcal{T}$ is a theory with equivalence relations $\sim$ and $\dot{\sim}$ such that (in every model of $\mathcal{T}$)*[4] *there are infinitely many*

[4] In fact, it would be sufficient to have a notion similar to *stable infiniteness* wrt. the equivalence classes, i.e. whenever a formula is satisfiable, it is also satisfiable in a model with infinitely many equivalence classes.

equivalence classes wrt. $\sim$, then the extension of $\mathcal{T}$ with a function f satisfying

$$\neg(\overline{x} \dot{\sim} \overline{y}) \rightarrow f(\overline{x}) \not\sim f(\overline{y}) \qquad (\mathsf{Inj})$$

satisfies $(\mathsf{Comp}_{\mathsf{w,f}})$.[5]

Proof idea: Any finite partial model of $\mathcal{T} \cup (\mathsf{Inj})$ must map elements in different equivalence classes wrt. $\dot{\sim}$ to different equivalence classes wrt. $\sim$. As $f_{\mathcal{M}}$ is only defined for finitely many values, we have infinitely many classes wrt. $\sim$ which are not mapped to by $f_{\mathcal{M}}$. Since we are interested in validity of first-order formulas, we can wlog. assume that the number of equivalence classes for both $\sim$ and $\dot{\sim}$ are at most countably infinite. Thus, we can find an injective function between the unused equivalence classes. Based on this, we can extend $f_{\mathcal{M}}$ to a total function satisfying (Inj). □

Theorem 7 (Locality of $(\mathsf{Con}) \cup (\mathsf{Mon})$**).** *Let $\mathcal{T}$ be a theory with equivalence relations $\sim$ and $\dot{\sim}$, and a binary relation $\preceq$ s.t. $\models_{\mathcal{T}} x \preceq y \wedge y \preceq x \iff x \sim y$, and either (i) $\preceq$ is a total preorder, or (ii) $\preceq$ defines a bounded semi-prelattice. Let furthermore $R(\overline{x}, \overline{y})$ be a binary predicate which is transitive in $\mathcal{T}$. Then the extension of $\mathcal{T}$ with a function f satsfying* $(\mathsf{Con}) \cup (\mathsf{Mon})$ *satisfies* $(\mathsf{Comp}_{\mathsf{w}})$.

Proof idea: The completion from the proof of Theorem 5 respects equivalence classes, i.e. values from the same equivalence class are mapped into the same equivalence class, unless the partial model already contains values that do not satisfy this condition. Thus, if the partial model satisfies (Con), then the completion will satisfy $(\mathsf{Con}) \cup (\mathsf{Mon})$. □

Theorem 8 (Locality of $(\mathsf{Con}) \cup (\mathsf{Inj})$**).** *Let $\mathcal{T}$ be a theory with equivalence relations $\sim$ and $\dot{\sim}$. Then the extension of $\mathcal{T}$ with a function f satisfying* $(\mathsf{Con}) \cup (\mathsf{Inj})$ *satisfies* $(\mathsf{Comp}_{\mathsf{w}})$ *if and only if $|dom(f)/\dot{\sim}| \leq |codom(f)/\sim|$.*

Proof idea: Since (Con) is required on the partial models, completion is easier as when only (Inj) is required. As elements from the same equivalence class need to be mapped to the same class, we can find an injective function between unused classes whenever $|dom(f)/\dot{\sim}| \leq |codom(f)/\sim|$. □

Theorem 9 (Locality of $(\mathsf{Con}) \cup \Phi(\overline{x}, f(\overline{x}))$**).** *Let $\mathcal{T}$ be a theory with equivalence relations $\sim$ and $\dot{\sim}$, and $\Phi(x, f(x))$ as in Theorem 3. Then the extension of $\mathcal{T}$ with a function f satisfying* $(\mathsf{Con}) \cup \forall \overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$ *satisfies* $(\mathsf{Comp}_{\mathsf{w}})$ *if and only if $\models_{\mathcal{T}} \forall \overline{x_1}, \overline{x_2}, y_1.\ (\overline{x_1} \dot{\sim} \overline{x_2} \wedge \Phi(\overline{x_1}, y_1) \rightarrow \exists y_2.\ y_1 \sim y_2 \wedge \Phi(\overline{x_2}, y_2))$.*

Proof idea: When completing a partial model $\mathcal{M}$ of $\mathcal{T} \cup (\mathsf{Con}) \cup \forall x.\ \Phi(\overline{x}, f(\overline{x}))$, for every value $\overline{a}$ where $f_{\mathcal{M}}(\overline{a})$ is undefined we need to find a value c such that $\Phi(\overline{a}, c)$ holds and whenever $\overline{a} \dot{\sim} \overline{b}$, then we also need to have $c \sim f_{\mathcal{M}}(\overline{b})$ (if defined). The condition guarantees that such a value can always be found.

[5] $(\mathsf{Comp}_{\mathsf{w,f}})$ is slightly weaker than $(\mathsf{Comp}_{\mathsf{w}})$ in that it only considers embeddability of models where partial functions have a finite domain. It implies the slightly weakened locality condition $(\mathsf{ELoc}_{\mathsf{f}})$, which requires G to be finite.

On the other hand, if the condition does not hold, then there exist $\overline{a_1}, \overline{a_2}, b_1$ with $\overline{a_1} \dot{\sim} \overline{a_2}$ and $\Phi(\overline{a_1}, b_1)$, but for all b_2 with $b_1 \sim b_2$ we have $\neg\Phi(\overline{a_2}, b_2)$. This means that the partial model $\mathcal{M}$ where $f_{\mathcal{M}}(\overline{a_1}) = b_1$ can not be extended to a total model, since for $f_{\mathcal{M}}(\overline{a_2})$ there is no value b_2 which satisfies both $b_1 \sim b_2$ and $\Phi(\overline{a_2}, b_2)$. □

Theorem 10 (Locality of $(\mathsf{Inj}) \cup \Phi(\overline{x}, f(\overline{x}))$**).** *Let $\mathcal{T}$ be a theory with equivalence relations $\sim$ and $\dot{\sim}$, and $\Phi(\overline{x}, f(\overline{x}))$ as in Theorem 3. Then the extension of $\mathcal{T}$ with a function f satisfying* $(\mathsf{Inj}) \cup \forall\overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$ *satisfies* $(\mathsf{Comp_w})$ *if* $\models_{\mathcal{T}} \forall\overline{x_1}, \overline{x_2}, y_1.\ (\neg(\overline{x_1} \dot{\sim} \overline{x_2}) \wedge \Phi(\overline{x_1}, y_1) \rightarrow \exists^{\infty} y_2.\ y_1 \not\sim y_2 \wedge \Phi(\overline{x_2}, y_2))$.

Above, $\exists^{\infty} y_2$ means that there must exist infinitely many values y_2 in different equivalence classes wrt. $\sim$. In contrast to Theorem 9, the condition is sufficient, but not necessary.

Proof idea: The completion is similar to that of Theorem 9, except that for $\overline{a}$ with $f_{\mathcal{M}}(\overline{a})$ undefined we need to find values b that satisfy $\Phi(\overline{a}, b)$ and need to be in a different equivalence class than the values of any other $\overline{a'}$ with $f_{\mathcal{M}}(\overline{a'})$ defined and $\neg(\overline{a} \dot{\sim} \overline{a'})$. If there are infinitely many values in different equivalence classes satisfying this, then we can complete $f_{\mathcal{M}}$ to a total function satisfying $(\mathsf{Inj}) \cup \forall\overline{x}.\ \Phi(\overline{x}, f(\overline{x}))$. □

Theorem 11 (Locality of $(\mathsf{Con}) \cup (\mathsf{Inj}) \cup \Phi(\overline{x}, f(\overline{x}))$**).** *Let $\mathcal{T}$ be a theory with equivalence relations $\sim$ and $\dot{\sim}$, and $\Phi(\overline{x}, f(\overline{x}))$ as in Theorem 3. The extension of $\mathcal{T}$ with a function f satisfying* $(\mathsf{Con}) \cup (\mathsf{Inj}) \cup \forall x.\ \Phi(x, f(x))$ *satisfies* $(\mathsf{Comp_w})$ *if and only if* $\models_{\mathcal{T}} \forall\overline{x_1}, \overline{x_2}, y_1.\ (\overline{x_1} \dot{\sim} \overline{x_2} \wedge \Phi(\overline{x_1}, y_1) \rightarrow \exists y_2.\ y_1 \sim y_2 \wedge \Phi(\overline{x_2}, y_2))$ *and* $|dom(f)/\dot{\sim}| \leq |codom(f)/\sim|$.

Proof idea: We can combine the completion from previous proofs, first defining $f_{\mathcal{M}}(\overline{a})$ for values where there is an $\overline{a'}$ with $\overline{a} \dot{\sim} \overline{a'}$ and $f_{\mathcal{M}}(\overline{a'})$ is defined (like in the proof of Theorem 9), and then for the other values (like in the proof of Theorem 10). Since now the partial model must satisfy (Con), we are guaranteed to find a completion whenever $|dom(f)/\dot{\sim}| \leq |codom(f)/\sim|$.

On the other hand, the completion cannot work if one of the conditions does not hold, either by the same argument as in the proof of Theorem 9, or because no injective functions wrt. $\sim$ can exists if $|dom(f)/\dot{\sim}| > |codom(f)/\sim|$ □

Lemma 1. *Let $\mathcal{T}$ be a Π_0-theory and $\mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ a set of augmented Π-clauses such that $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ satisfies* $(\mathsf{Comp_w})$*. For any Π_0-formula $\Phi(\overline{x})$, the extension $\mathcal{T} \subseteq \mathcal{T} \cup (\Phi(\overline{x}) \rightarrow \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ also satisfies* $(\mathsf{Comp_w})$.

Proof. Since $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ satisfies $(\mathsf{Comp_w})$, every partial model of $\mathcal{T} \cup \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ can be completed. For every partial model $\mathcal{M}$ of $\mathcal{T} \cup (\Phi(\overline{x}) \rightarrow \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$, there is a partial model $\mathcal{M}'$ of $\mathcal{T} \cup (\Phi(\overline{x}) \rightarrow \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ which is also a partial model of $\mathcal{T} \cup \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ (it can be obtained from $\mathcal{M}$ by letting $f^{\mathcal{M}'}(x)$ be defined only if $\mathcal{M} \models \Phi(x)$). By

assumption, we can complete $\mathcal{M}'$ to a total model $\mathcal{M}''$ of $\mathcal{T} \cup \mathcal{K}(f(\overline{x}_1), \ldots, f(\overline{x}_n))$. Now, let

$$\overline{f}(x) = \begin{cases} f^{\mathcal{M}}(x), & \text{if defined} \\ f^{\mathcal{M}''}(x), & \text{if } f^{\mathcal{M}}(x) \text{ undefined and } \mathcal{M} \models \Phi(x) \\ \textit{arbitrary}, & \text{else} \end{cases}$$

□

Based on Lemma 1, we can prove that case distinctions with disjoint cases preserves decidability:

Theorem 12 (Locality of Disjoint Piecewise Combinations). *Let $\mathcal{T}$ be a Π_0-theory and consider Π_0-formulas $\Phi_1(\overline{x}), \Phi_2(\overline{x})$ such that $\models_{\mathcal{T}} \neg(\Phi_1(\overline{x}) \wedge \Phi_2(\overline{x}))$. If $\mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $\mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ are Π-formulas such that both $\mathcal{T} \subseteq \mathcal{T} \cup (\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $\mathcal{T} \subseteq \mathcal{T} \cup (\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ satisfy condition $(\mathsf{Comp_w})$, then for*

$$\begin{aligned} \mathcal{K} = \quad & (\textstyle\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i)) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n)) \\ \cup\ & (\textstyle\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i)) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m)), \end{aligned}$$

the extension $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}$ satisfies $(\mathsf{Comp_w})$ and is a local extension.

Proof. Consider a partial model $\mathcal{M}$ of $\mathcal{T} \cup \mathcal{K}$. Since $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $\mathcal{T} \subseteq \mathcal{T} \cup \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ satisfy $(\mathsf{Comp_w})$, by Lemma 1 also $\mathcal{T} \subseteq \mathcal{T} \cup (\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i)) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $\mathcal{T} \subseteq \mathcal{T} \cup (\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i)) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ satisfy $(\mathsf{Comp_w})$.

Thus, we can complete $\mathcal{M}$ to a total model $\mathcal{M}_1$ of $\mathcal{T} \cup (\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i)) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and to a total model $\mathcal{M}_2$ of $\mathcal{T} \cup (\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i)) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$. Then, define

$$\overline{f}(\overline{x}) = \begin{cases} f^{\mathcal{M}_1}(\overline{x}), & \text{if } \mathcal{M} \models \Phi_1(\overline{x}) \\ f^{\mathcal{M}_2}(\overline{x}), & \text{if } \mathcal{M} \models \Phi_2(\overline{x}) \\ \textit{arbitrary}, & \text{else} \end{cases}$$

It is easy to see that the resulting structure satisfies $\mathcal{T} \cup \mathcal{K}$. □

Theorem 13 (Locality of Piecewise Combinations with Finite Intersection). *Let $\mathcal{T}$ be a theory with signature $\Pi_0 = (\Sigma_0, \mathsf{Pred})$ and consider Π_0-formulas $\Phi_1(\overline{x}), \Phi_2(\overline{x})$ such that in every $\mathcal{T}$-model $\mathcal{M}$, the set $O = \{\overline{x} \in M | \Phi_1(\overline{x}) \wedge \Phi_2(\overline{x})\}$ is finite. Let furthermore T_0 be a set of Σ_0-terms such that in every such model $\mathcal{M}$, $O \subseteq \{t^{\mathcal{M}} \mid t \in T_0\}$.*

If $\mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $\mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ are sets of augmented Π-clauses such that both the extensions of $\mathcal{T}$ with $(\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $(\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ satisfy $(\mathsf{Comp_w})$, then for

$$\begin{aligned} \mathcal{K} = \quad & (\textstyle\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i)) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n)) \\ \cup\ & (\textstyle\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i)) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m)) \end{aligned}$$

and $\Psi(T) = T \cup \{f(t) | t \in T_0\}$, the extension of $\mathcal{T}$ with $\mathcal{K}$ satisfies $(\mathsf{Comp_w}^{\Psi})$.

Proof. Let $\mathcal{M}$ be a partial model of $\mathcal{T} \cup \mathcal{K}$ in which $f^{\mathcal{M}}(a_i)$ is defined for a finite set of values $a_1, \ldots, a_k$, where for every $t \in T_0$ we have $t^{\mathcal{M}} = a_i$ for some i. Since both $(\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i)) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $(\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i)) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$ satisfy ($\mathsf{Comp_w}$) individually, there are completions f_1, f_2 of $f^{\mathcal{M}}$, satisfying these sets of axioms. We define another completion $\overline{f}$ of $f^{\mathcal{M}}$ by

$$\overline{f}(\overline{x}) = \begin{cases} f^{\mathcal{M}}(\overline{x}), & \text{if defined} \\ f_1(\overline{x}), & \text{if } f^{\mathcal{M}}(\overline{x}) \text{ undefined and } \mathcal{M} \models \Phi_1(\overline{x}) \\ f_2(\overline{x}), & \text{if } f^{\mathcal{M}}(\overline{x}) \text{ undefined and } \mathcal{M} \models \Phi_2(\overline{x}) \\ \mathit{arbitrary}, & \text{else} \end{cases}$$

We need to show that this completion satisfies both $(\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i)) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$ and $(\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i)) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$. If $a_1, \ldots, a_n$ are values with $\Phi_1^{\mathcal{M}}(a_i)$ for each i, then $\overline{f}(a_i) = f_1(a_i)$, and since $\mathcal{M}_1 \models (\bigwedge_{i=1}^{n} \Phi_1(\overline{x}_i)) \rightarrow \mathcal{K}_1(f(\overline{x}_1), \ldots, f(\overline{x}_n))$, we know that this axiom will also be satisfied by $\overline{f}$. A similar argument holds for values $a_1, \ldots, a_m$ with $\Phi_2^{\mathcal{M}}(a_i)$ and axiom $(\bigwedge_{i=1}^{m} \Phi_2(\overline{x}_i)) \rightarrow \mathcal{K}_2(f(\overline{x}_1), \ldots, f(\overline{x}_m))$. Finally, we know that for all values a_i with $\Phi_1^{\mathcal{M}}(a_i) \wedge \Phi_2^{\mathcal{M}}(a_i)$ we know that $f^{\mathcal{M}}(a_i)$ is defined, so $f_1(a_i) = f_2(a_1)$. Thus, we know that for points in the intersection, both of the axioms hold. Clearly, for values a_i where both $\Phi_i^{\mathcal{M}}(a_i)$ are false, the axioms are satisfied trivially. □