Journal of Automated Reasoning (2022) 66:331–355
https://doi.org/10.1007/s10817-022-09625-3

Polite Combination of Algebraic Datatypes

Ying Sheng[1] · Yoni Zohar[2] · Christophe Ringeissen[3] · Jane Lange[4] · Pascal Fontaine[5] · Clark Barrett[1]

Received: 29 April 2021 / Accepted: 22 February 2022 / Published online: 5 May 2022

Abstract
Algebraic datatypes, and among them lists and trees, have attracted a lot of interest in automated reasoning and Satisfiability Modulo Theories (SMT). Since its latest stable version, the SMT-LIB standard defines a theory of algebraic datatypes, which is currently supported by several mainstream SMT solvers. In this paper, we study this particular theory of datatypes and prove that it is strongly polite, showing how it can be combined with other arbitrary disjoint theories using polite combination. The combination method uses a new, simple, and natural notion of additivity that enables deducing strong politeness from (weak) politeness.

Keywords Satisfiability Modulo Theories · Automated reasoning · Theory combination · Algebraic datatypes · Polite combination

1 Introduction

Algebraic datatypes such as records, lists, and trees are extremely common in many programming languages. Reasoning about them is therefore crucial for modeling and verifying programs. For this reason, various decision procedures for algebraic datatypes have been, and continue to be developed and employed by formal reasoning tools such as theorem provers and Satisfiability Modulo Theories (SMT) solvers. For example, the general algorithm of [4] describes a decision procedure for datatypes suitable for SMT solvers. Consistently with the SMT paradigm, [4] leaves the combination of datatypes with other theories to general

This project was partially supported by a Grant from the Defense Advanced Research Projects Agency (N66001-18-C-4012), the Stanford CURIS program, and Jasmin Blanchette's European Research Council (ERC) starting Grant Matryoshka (713999).

✉ Ying Sheng
ying1123@stanford.edu

1 Stanford University, Stanford, USA

2 Bar Ilan University, Ramat Gan, Israel

3 Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

4 MIT, Cambridge, USA

5 Université de Liège, Liège, Belgium

combination methods, and focuses on parametric datatypes (or *generic* datatypes as they are called in the programming languages community), where the interpretation of the "values" is left uninterpreted.

The traditional combination method of Nelson and Oppen [22] is applicable for the combination of this theory with many other theories, as long as the other theory is *stably infinite* (a technical condition that intuitively amounts to the ability to extend every model to an infinite one). Some theories of interest, however, are not stably infinite, the most notable one being the theory of fixed-width bit-vectors, which is commonly used for modeling and verifying both hardware and software. Further, many theories that are compounded from bit-vectors and other theories are also not stably infinite, e.g., arrays/sets of bit-vectors. Combining these theories with algebraic datatypes cannot be done using the Nelson–Oppen approach. To be able to perform combinations with such theories, a more general combination method was designed [23], which relies on *polite theories*. Roughly speaking, a theory is polite if: (i) every model can be arbitrarily enlarged; and (ii) there is a *witness*, a function that transforms any quantifier-free formula to an equivalent quantifier-free formula such that if the original formula is satisfiable, the new formula is satisfiable in a "minimal" interpretation. This notion was later strengthened to *strongly polite theories* [16], which also account for possible arrangements of the variables in the formula, as well as arbitrary auxiliary variables. Strongly polite theories can be combined with any other disjoint decidable theory, even if that other theory is not stably infinite. While strong politeness was already proven for several useful theories (such as equality, arrays, sets, multisets [23]), strong politeness of algebraic datatypes remained an unanswered question.

The main contribution of this paper is an affirmative answer to this question. While enlarging models of algebraic datatypes is trivial, the challenge lies with finding a witness function, as well as minimal models for formulas that are produced by this function. We introduce a witness function that essentially "guesses" the right constructors of variables that do not have an explicit constructor in the formula. We show how to "shrink" any model of a formula that is the output of this function into a minimal model. The witness function, as well as the model construction, can be used by any SMT solver for the theory of datatypes that implements polite theory combination. We introduce and use the notion of additive witnesses, that offer a sufficient condition for a polite theory to be also strongly polite. This allows us to only prove politeness using our witness function, and conclude strong politeness, since our function is indeed additive. We further study the theory of datatypes beyond politeness and extend a decision procedure for a subset of this theory presented in [11] to support the full theory.

Related Work

The theory investigated in this paper is that of algebraic datatypes, as defined by the SMT-LIB 2 standard [6]. Detailed information on this theory, including a decision procedure and related work, can be found in [4]. Later work extends this procedure to handle shared selectors [25] and co-datatypes [24]. More recent approaches for solving formulas about datatypes use, e.g., theorem provers [17], variant satisfiability [14, 21], and reduction-based decision procedures [1, 8, 15].

In this paper, we focus on polite theory combination. Other combination methods for non stably infinite theories include shiny theories [32], gentle theories [13], and parametric theories [19]. The politeness property was introduced in [23], and extends the stable infiniteness assumption initially used by Nelson and Oppen. Polite theories can be combined à la Nelson–Oppen with any arbitrary decidable theory. Later, a flaw in the original definition of politeness was found [16], and a corrected definition (here called *strong politeness*) was introduced. Polite combination is implemented in the SMT-solver cvc5 (the successor of CVC4 [5]) that also includes a solver for the theory of algebraic datatypes. Strongly polite theories were further studied in [10], where the authors proved their equivalence with shiny theories. The relation between politeness and strong politeness was investigated in [27], where it was shown that the latter is a strictly stronger notion than the former.

More recently, it was proved [11] that a general family of datatype theories extended with bridging functions is strongly polite. This includes the theories of lists/trees with length/size functions. The authors also proved that a class of axiomatizations of datatypes is strongly polite. In contrast, in this paper, we focus on standard interpretations, as defined by the SMT-LIB 2 standard, without any size function, but including selectors and testers. One can notice that the theory of standard lists without the length function and more generally the theory of finite trees without the size function were not mentioned as polite in a recent survey [9]. Actually, it was unclear to the authors of [9] whether these theories are strongly polite. This is now clarified in the current paper.

Outline

The paper is organized as follows. Section 2 provides the necessary notions from first-order logic, algebraic datatypes, and polite theories. Section 3 discusses the difference between politeness and strong politeness and introduces a useful condition for their equivalence. Section 4 contains the main result of this paper, namely that the theory of algebraic datatypes is strongly polite. Section 5 studies various axiomatizations of the theory of datatypes, and relates them to politeness. Section 6 concludes with directions for further research.[1]

2 Preliminaries

2.1 Signatures and Structures

We briefly review usual definitions of many-sorted first-order logic with equality (see [12, 31] for more details). Let S denote a set of *sorts*. An *S-sorted set* A associates non-empty pairwise disjoint sets to the sorts of S. That is, A is a function from S to $\mathcal{P}(X) \setminus \{\emptyset\}$ for some set X, such that $A(\sigma) \cap A(\sigma') = \emptyset$ whenever $\sigma \neq \sigma'$, $\sigma, \sigma' \in S$. We use A_σ to denote $A(\sigma)$ for every $\sigma \in S$. When there is no ambiguity, we sometimes treat sorted sets as sets (e.g., when writing expressions like $x \in A$ for $x \in \bigcup_{\sigma \in S} A_\sigma$). Given a set S (of sorts), the *canonical S-sorted set*, denoted $[[S]]$, satisfies $[[S]]_\sigma = \{\sigma\}$ for every $\sigma \in S$. A *many-sorted signature* Σ consists of a set $\mathcal{S}_\Sigma$ (of *sorts*), a set $\mathcal{F}_\Sigma$ of function symbols, and a set $\mathcal{P}_\Sigma$ of predicate symbols. Function symbols have arities of the form $\sigma_1 \times \cdots \times \sigma_n \to \sigma$, and predicate symbols have arities of the form $\sigma_1 \times \cdots \times \sigma_n$, with $\sigma_1, \ldots, \sigma_n, \sigma \in \mathcal{S}_\Sigma$. For each sort $\sigma \in \mathcal{S}_\Sigma$, the logic includes an *equality symbol* $=_\sigma$ of arity $\sigma \times \sigma$, whose interpretation is fixed to be the identity. We denote it by $=$ when σ is clear from context. Σ is called *finite* if $\mathcal{S}_\Sigma$, $\mathcal{F}_\Sigma$, and $\mathcal{P}_\Sigma$ are finite.

We assume an underlying $\mathcal{S}_\Sigma$-sorted set of *variables*. Terms, formulas, and literals are defined in the usual way. For a Σ-formula ϕ and a sort σ, we denote the set of free variables in

[1] A preliminary version of this work was published in the proceedings of IJCAR 2020 [26]. The current article extends the original versions with complete proofs, as well as a discussion and results regarding existential theories (see Proposition 2). Additionally, Sect. 5 is extended to provide a more comprehensive treatment of axiomatizations for trees.

ϕ of sort σ by $vars_\sigma(\phi)$. This notation naturally extends to $vars_S(\phi)$ when S is a set of sorts. A sentence is a formula without free variables. We denote by $QF(\Sigma)$ the set of quantifier-free formulas of Σ. A Σ-literal is called *flat* if it has one of the following forms: $x = y$, $x \neq y$, $x = f(x_1, \ldots, x_n)$, $P(x_1, \ldots, x_n)$, or $\neg P(x_1, \ldots, x_n)$ for some variables $x, y, x_1, \ldots, x_n$, function symbol f, and predicate symbol P from Σ.

A Σ-*structure* is a many-sorted structure for Σ, without interpretation of variables. It consists of a $\mathcal{S}_\Sigma$-sorted set A that interprets the sort symbols of Σ as sets, and interpretations of the function and predicate symbols of Σ. For any Σ-term α, $\alpha^{\mathcal{A}}$ denotes the interpretation of α in $\mathcal{A}$. In particular, every function symbol f of arity $\sigma_1 \times \cdots \times \sigma_n \to \sigma$ is interpreted as a function in $\sigma_1^{\mathcal{A}} \times \cdots \times \sigma_n^{\mathcal{A}} \to \sigma^{\mathcal{A}}$, and every predicate symbol P of arity $\sigma_1 \times \cdots \times \sigma_n$ is interpreted as a subset of $\sigma_1^{\mathcal{A}} \times \cdots \times \sigma_n^{\mathcal{A}}$.

A Σ-*interpretation* $\mathcal{A}$ is an extension of a Σ-structure with interpretations to some set of variables. When α is a set of Σ-terms, $\alpha^{\mathcal{A}} = \{t^{\mathcal{A}} \mid t \in \alpha\}$. Similarly, $\sigma^{\mathcal{A}}$, $f^{\mathcal{A}}$ and $P^{\mathcal{A}}$ denote the interpretation of σ, f and P in $\mathcal{A}$. Satisfaction is defined as usual. $\mathcal{A} \models \varphi$ denotes that $\mathcal{A}$ satisfies φ.

A Σ-theory T is a class of Σ-structures. A Σ-interpretation whose variable-free part is in T is called a T-interpretation. A Σ-formula ϕ is T-satisfiable if $\mathcal{A} \models \phi$ for some T-interpretation $\mathcal{A}$. Two formulas ϕ and ψ are T-*equivalent* if they are satisfied by the same class of T-interpretations. Let Σ_1 and Σ_2 be signatures, T_1 a Σ_1-theory, and T_2 a Σ_2-theory. The combination of T_1 and T_2, denoted $T_1 \oplus T_2$, is the class of $\Sigma_1 \cup \Sigma_2$-structures $\mathcal{A}$ such that $\mathcal{A}^{\Sigma_1}$ is in T_1 and $\mathcal{A}^{\Sigma_2}$ is in T_2, where $\mathcal{A}^{\Sigma_i}$ is the restriction of $\mathcal{A}$ to Σ_i for $i \in \{1, 2\}$.

2.2 The SMT-LIB 2 Theory of Datatypes

In this section, we formally define the SMT-LIB 2 theory of algebraic datatypes. The formalization is based on [6], but is adjusted to suit our investigation of politeness.

Definition 1 (Σ-*Trees*) Given a signature Σ, a set $S \subseteq \mathcal{S}_\Sigma$ and an S-sorted set A, the set of Σ-*trees* over A of sort $\sigma \in \mathcal{S}_\Sigma$ is denoted by $T_\sigma(\Sigma, A)$ and is inductively defined as follows:

- $T_{\sigma,0}(\Sigma, A) = A_\sigma$ if $\sigma \in S$ and $\emptyset$ otherwise.
- $T_{\sigma,i+1}(\Sigma, A) = T_{\sigma,i}(\Sigma, A) \cup \{c(t_1, \ldots, t_n) \mid c : \sigma_1 \times \cdots \times \sigma_n \to \sigma \in \mathcal{F}_\Sigma, t_j \in T_{\sigma_j,i}(\Sigma, A)$ for $j = 1, \ldots, n\}$ for each $i \geq 0$.

Then $T_\sigma(\Sigma, A) = \bigcup_{i \geq 0} T_{\sigma,i}(\Sigma, A)$. The *depth* of a Σ-tree over A is inductively defined by $depth(a) = 0$ for every $a \in A$, $depth(c) = 1$ for every 0-argument function symbol $c \in \mathcal{F}_\Sigma$, and $depth(c(t_1, \ldots, t_n)) = 1 + max(depth(t_1), \ldots, depth(t_n))$ for every n-argument function symbol c of Σ.

The idea behind Definition 1 is that $T_\sigma(\Sigma, A)$ contains all ground σ-sorted terms constructed from the elements of A (considered as constant symbols) and the function symbols of Σ.

Example 1 Let Σ be a signature with two sorts, **elem** and **struct**, and whose function symbols are b of arity **struct**, and c of arity (**elem** $\times$ **struct** $\times$ **struct**) $\to$ **struct**. Consider the $\{$**elem**$\}$-sorted set $A = \{a\}$. $T_{\mathbf{elem}}(\Sigma, A)$ is the singleton $A = \{a\}$ and the Σ-tree a is of depth 0. $T_{\mathbf{struct}}(\Sigma, A)$ includes infinitely many Σ-trees, such as b of depth 1, $c(a, b, b)$ of depth 2, and $c(a, c(a, b, b), b)$ of depth 3.

Definition 2 (*Datatypes Signature*) A finite signature Σ is called a *datatypes signature* if $\mathcal{S}_\Sigma$ is the disjoint union of two sets of sorts $\mathcal{S}_\Sigma = \mathbf{Elem}_\Sigma \uplus \mathbf{Struct}_\Sigma$, $\mathcal{F}_\Sigma$ is the disjoint union of two sets of function symbols $\mathcal{F}_\Sigma = \mathcal{CO}_\Sigma \uplus \mathcal{SE}_\Sigma$, such that every $c \in \mathcal{CO}_\Sigma$ has arity $\sigma_1 \times \cdots \times \sigma_n \to \sigma$ with $\sigma \in \mathbf{Struct}_\Sigma$, $\mathcal{SE}_\Sigma = \{s_{c,i} \mid c \in \mathcal{CO}_\Sigma, c : \sigma_1 \times \cdots \times \sigma_n \to \sigma, 1 \leq i \leq n\}$ where for each $c \in \mathcal{CO}_\Sigma$ with $c : \sigma_1 \times \cdots \times \sigma_n \to \sigma$ and for each $i = 1, \ldots, n$, $s_{c,i}$ is a function symbol of arity $\sigma \to \sigma_i$, and $\mathcal{P}_\Sigma = \{is_c \mid c \in \mathcal{CO}_\Sigma, c : \sigma_1 \times \cdots \times \sigma_n \to \sigma\}$ where for each $c \in \mathcal{CO}_\Sigma$ with $c : \sigma_1 \times \cdots \times \sigma_n \to \sigma$, is_c is a predicate symbol of arity σ. We denote by $\Sigma_{|\mathcal{CO}}$ the signature with the same sorts as Σ, $\mathcal{CO}_\Sigma$ as set of function symbols, and an empty set of predicate symbols. We further require the following *well-foundedness* requirement for Σ to be called a datatypes signature: $T_\sigma(\Sigma_{|\mathcal{CO}}, [[\mathbf{Elem}_\Sigma]]) \neq \emptyset$ for any $\sigma \in \mathbf{Struct}_\Sigma$.

From now on, we omit the subscript Σ from the above notations (e.g., when writing $[[\mathbf{Elem}]]$ rather than $[[\mathbf{Elem}_\Sigma]]$ and $\mathcal{CO}$ rather than $\mathcal{CO}_\Sigma$) whenever Σ is clear from the context. Notice that in Definition 2, the well-foundedness requirement ensures that the set of $\Sigma_{|\mathcal{CO}}$-trees of sort σ over $[[\mathbf{Elem}]]$ is not empty for every $\sigma \in \mathbf{Struct}_\Sigma$. The choice of $[[\mathbf{Elem}]]$ is not essential here, and the definition remains equivalent if $[[\mathbf{Elem}]]$ is replaced by any **Elem**-sorted set that is non-empty for each sort. The set $[[\mathbf{Elem}]]$ has been chosen since it is a minimal such **Elem**-sorted set.

In accordance with SMT-LIB 2, we call the elements of $\mathcal{CO}$ *constructors*, the elements of $\mathcal{SE}$ *selectors*, and the elements of $\mathcal{P}$ *testers*. Constructors that take no arguments are called *nullary*. In what follows, Σ denotes an arbitrary datatypes signature.

In the next example, we review some common datatypes signatures.

Example 2 The signature Σ_{list} has two sorts, **elem** and **list**. Its function symbols are *cons* of arity $(\mathbf{elem} \times \mathbf{list}) \to \mathbf{list}$, *nil* of arity **list**, *car* of arity $\mathbf{list} \to \mathbf{elem}$ and *cdr* of arity $\mathbf{list} \to \mathbf{list}$. Its predicate symbols are is_{nil} and is_{cons}, both of arity **list**. It is a datatypes signature, with $\mathbf{Elem} = \{\mathbf{elem}\}$, $\mathbf{Struct} = \{\mathbf{list}\}$, $\mathcal{CO} = \{nil, cons\}$ and $\mathcal{SE} = \{car, cdr\}$. It is often used to model lisp-style linked lists. *car* represents the head of the list and *cdr* represents its tail. *nil* represents the empty list. Σ_{list} is well founded as $T_{\mathbf{list}}(\Sigma_{list\,|\mathcal{CO}}, [[\mathbf{Elem}]])$ includes *nil*.

The signature Σ_{pair} also has two sorts, **elem** and **pair**. Its function symbols are *pair* of arity $(\mathbf{elem} \times \mathbf{elem}) \to \mathbf{pair}$ and *first* and *second* of arity $\mathbf{pair} \to \mathbf{elem}$. Its predicate symbol is is_{pair} of arity **pair**. It is a datatypes signature, with $\mathbf{Elem} = \{\mathbf{elem}\}$, $\mathbf{Struct} = \{\mathbf{pair}\}$, $\mathcal{CO} = \{pair\}$, and $\mathcal{SE} = \{first, second\}$. It can be used to model ordered pairs, together with projection functions. It is well founded as $T_{\mathbf{pair}}(\Sigma_{pair\,|\mathcal{CO}}, [[\mathbf{Elem}]])$ is not empty (as $[[\mathbf{Elem}]]$ is not empty).

The signature Σ_{lp} has three sorts, **elem**, **pair** and **list**, with $\mathbf{Elem} = \{\mathbf{elem}\}$ and $\mathbf{Struct} = \{\mathbf{pair}, \mathbf{list}\}$. Its function symbols are *cons* of arity $(\mathbf{pair} \times \mathbf{list}) \to \mathbf{list}$, *car* of arity $\mathbf{list} \to \mathbf{pair}$, as well as *nil*, *cdr*, *first*, *second* with arities as above. Its predicate symbols are is_{pair}, is_{cons} and is_{nil}, with arities as above. It can be used to model lists of ordered pairs. Similarly to the above signatures, it is a datatypes signature.

Next, we distinguish between finite datatypes (e.g., records) and inductive datatypes (e.g., lists).

Definition 3 (*Inductive and Finite Sorts*) A sort $\sigma \in \mathbf{Struct}$ is called *finite* if $T_\sigma(\Sigma_{|\mathcal{CO}}, [[\mathbf{Elem}]])$ is finite and is called *inductive* otherwise.

We denote the set of inductive sorts in Σ by $Ind(\Sigma)$ and the set of its finite sorts by $Fin(\Sigma)$. Note that if σ is inductive, then according to Definitions 1 and 3 we have that for any natural number i, there exists a natural number $i' > i$ such that $T_{\sigma,i'}(\Sigma_{|\mathcal{CO}}, [[\mathbf{Elem}]]) \neq T_{\sigma,i}(\Sigma_{|\mathcal{CO}}, [[\mathbf{Elem}]])$. Further, for any natural number d and every **Elem**-sorted set D there exists a natural number i' such that $T_{\sigma,i'}(\Sigma_{|\mathcal{CO}}, D)$ contains an element whose depth is greater than d.

Example 3 **list** is inductive in Σ_{list} and Σ_{lp}. **pair** is finite in Σ_{pair} and Σ_{lp}.

Finally, we define datatypes structures and the theory of algebraic datatypes.

Definition 4 (*Datatypes Structure*) Let Σ be a datatypes signature and D an **Elem**-sorted set. A Σ-structure $\mathcal{A}$ is said to be a *datatypes Σ-structure generated by D* if:

- $\sigma^{\mathcal{A}} = T_\sigma(\Sigma_{|\mathcal{CO}}, D)$ for every sort $\sigma \in \mathcal{S}_\Sigma$,
- $c^{\mathcal{A}}(t_1, \ldots, t_n) = c(t_1, \ldots, t_n)$ for every $c \in \mathcal{CO}$ of arity $(\sigma_1 \times \cdots \times \sigma_n) \to \sigma$ and $t_1 \in \sigma_1^{\mathcal{A}}, \ldots, t_n \in \sigma_n^{\mathcal{A}}$,
- $s_{c,i}^{\mathcal{A}}(c(t_1, \ldots, t_n)) = t_i$ for every $c \in \mathcal{CO}$ of arity $(\sigma_1 \times \cdots \times \sigma_n) \to \sigma, t_1 \in \sigma_1^{\mathcal{A}}, \ldots, t_n \in \sigma_n^{\mathcal{A}}$ and $1 \leq i \leq n$,
- $is_c^{\mathcal{A}} = \left\{c(t_1, \ldots, t_n) \mid t_1 \in \sigma_1^{\mathcal{A}}, \ldots, t_n \in \sigma_n^{\mathcal{A}}\right\}$ for every $c \in \mathcal{CO}$ of arity $(\sigma_1 \times \cdots \times \sigma_n) \to \sigma$.

$\mathcal{A}$ is said to be a *datatypes Σ-structure* if it is a datatypes Σ-structure generated by D for some **Elem**-sorted set D. The *Σ-theory of datatypes*, denoted $\mathcal{T}_\Sigma$ is the class of datatypes Σ-structures.

Notice that the interpretation of selector functions $s_{c,i}$ when applied to terms that are constructed using a constructor different than c is not fixed and can be set arbitrarily in datatypes structures, consistently with SMT-LIB 2.

Example 4 If $\mathcal{A}$ is a datatypes Σ_{list}-structure then $\textbf{list}^{\mathcal{A}}$ is the set of terms constructed from $\textbf{elem}^{\mathcal{A}}$ and *cons*, plus *nil*. If $\textbf{elem}^{\mathcal{A}}$ is the set of natural numbers, then $\textbf{list}^{\mathcal{A}}$ contains, e.g., *nil*, $cons(1, nil)$, and $cons(1, cons(1, cons(2, nil)))$. These correspond to the lists [] (the empty list), [1] and [1, 1, 2], respectively.

If $\mathcal{A}$ is a datatypes Σ_{pair}-structure then $\textbf{pair}^{\mathcal{A}}$ is the set of terms of the form $pair(a, b)$ with $a, b \in \textbf{elem}^{\mathcal{A}}$. If $\textbf{elem}^{\mathcal{A}}$ is again interpreted as the set of natural numbers, $\textbf{pair}^{\mathcal{A}}$ includes, for example, the terms $pair(1, 1)$ and $pair(1, 2)$, that correspond to (1, 1) and (1, 2), respectively. Notice that in this case, $\textbf{pair}^{\mathcal{A}}$ is an infinite set even though **pair** is a finite sort (in terms of Definition 3).

Datatypes Σ_{lp}-structures with the same interpretation for **elem** include the terms *nil*, $cons(pair(1, 1), nil)$, and $cons(pair(1, 1), cons(pair(1, 2), nil))$ in the interpretation for **list** that correspond to [], [(1, 1)] and [(1, 1), (1, 2)], respectively. If we rename **elem** in the definition of Σ_{list} to **pair**, we get that $\mathcal{T}_{\Sigma_{lp}} = \mathcal{T}_{\Sigma_{list}} \oplus \mathcal{T}_{\Sigma_{pair}}$.

2.3 Polite Theories

Given two theories T_1 and T_2, a combination method à la Nelson–Oppen provides a modular way to decide $T_1 \oplus T_2$-satisfiability problems using the satisfiability procedures known for T_1 and T_2. Assuming that T_1 and T_2 have disjoint signatures (except that they share sorts) is not sufficient to get a complete combination method for deciding any $T_1 \oplus T_2$-satisfiability problem $\phi_1 \wedge \phi_2$ where ϕ_i is a T_i-satisfiability problem for $i = 1, 2$. The reason is that T_1 and T_2 may share sorts, and this implies the existence of shared formulas built over the corresponding equality symbols and the finite set of variables SV shared by ϕ_1 and ϕ_2. To be complete, T_1 and T_2 must agree on the cardinality of their respective models, and there must be an agreement between T_1 and T_2 on the interpretation of shared formulas. These two requirements can be fulfilled, based on the following definitions:

Definition 5 (*Stable Infiniteness*) Given a signature Σ and a set $S \subseteq \mathcal{S}_\Sigma$, we say that a Σ-theory T is *stably infinite with respect to* S if every quantifier-free Σ-formula that is T-satisfiable is also T-satisfiable by a T-interpretation $\mathcal{A}$ in which $\sigma^{\mathcal{A}}$ is infinite for every $\sigma \in S$.

Definition 6 (*Arrangement*) Let Σ be a signature, $S \subseteq \mathcal{S}_\Sigma$, V be a finite set of variables whose sorts are in S and $\{V_\sigma \mid \sigma \in S\}$ the partition of V such that V_σ is the set of variables of sort σ in V. We say that a formula δ is an *arrangement of* V if $\delta = \bigwedge_{\sigma \in S}(\bigwedge_{(x,y)\in E_\sigma}(x = y) \wedge \bigwedge_{(x,y)\notin E_\sigma}(x \neq y))$, where E_σ is some equivalence relation over V_σ for each $\sigma \in S$.

Assume that T_1 and T_2 are two signature-disjoint theories with the property of being stably infinite w.r.t. their shared sorts. Under this assumption, T_1 and T_2 can agree on an infinite cardinality, and guessing an arrangement of the finite set of shared variables SV suffices to get an agreement on the interpretation of shared formulas.

In this paper, we are interested in an asymmetric disjoint combination where T_1 and T_2 are not both stably infinite. In this scenario, one theory can be arbitrary. As a counterpart, the other theory must be more than stably infinite: it must be strongly polite, meaning that it is always possible to increase the cardinality of a model and to have a model whose cardinality is finite.

In the following, we decompose the politeness definition from [16, 23] in order to distinguish between politeness and strong politeness (in terms of [10]) in various levels of the definition. In what follows, Σ is an arbitrary (many-sorted) signature, $S \subseteq \mathcal{S}_\Sigma$, and T is a Σ-theory.

Definition 7 (*Smooth*) The theory T is *smooth* w.r.t. S if for every quantifier-free formula ϕ, T-interpretation $\mathcal{A}$ that satisfies ϕ, and function κ from S to the class of cardinals such that $\kappa(\sigma) \geq |\sigma^{\mathcal{A}}|$ for every $\sigma \in S$, there exists a T-interpretation $\mathcal{A}'$ that satisfies ϕ with $|\sigma^{\mathcal{A}'}| = \kappa(\sigma)$ for every $\sigma \in S$.

In definitions introduced above, as well as below, we often identify singletons with their single elements when there is no ambiguity (e.g., when saying that a theory is smooth w.r.t. a sort σ).

We now introduce some concepts in order to define finite witnessability.

Definition 8 (*Finitely Witnessable*) Let ϕ be a quantifier-free Σ-formula and $\mathcal{A}$ a Σ-interpretation. We say that $\mathcal{A}$ *finitely witnesses* ϕ *for* T *w.r.t.* S (or, is a *finite witness of* ϕ *for* T *w.r.t.* S), if $\mathcal{A}$ is a T-interpretation, $\mathcal{A} \models \phi$, and $\sigma^{\mathcal{A}} = vars_\sigma(\phi)^{\mathcal{A}}$ for every $\sigma \in S$.

We say that ϕ is *finitely witnessed for* T *w.r.t.* S if it is either T-unsatisfiable or it has a finite witness for T w.r.t. S. We say that ϕ is *strongly finitely witnessed for* T *w.r.t.* S if for any set of variables V whose sorts are in S, and any arrangement δ_V of V, $\phi \wedge \delta_V$ is finitely witnessed for T w.r.t. S.

We say that a function $wtn : QF(\Sigma) \rightarrow QF(\Sigma)$ is a *(strong) witness for* T *w.r.t.* S if for every $\phi \in QF(\Sigma)$ we have that: 1. ϕ and $\exists \overrightarrow{w}.\, wtn(\phi)$ are T-equivalent for $\overrightarrow{w} = vars(wtn(\phi)) \setminus vars(\phi)$; and 2. $wtn(\phi)$ is (strongly) finitely witnessed for T w.r.t. S.[2]

The theory T is *(strongly) finitely witnessable* w.r.t. S if there exists a (strong) witness for T w.r.t. S which is computable.

Definition 9 (*Polite*) T is called *(strongly) polite w.r.t.* S if it is smooth and (strongly) finitely witnessable w.r.t. S.

2 We note that in practice, the new variables in $wtn(\phi)$ are assumed to be fresh not only with respect to ϕ, but also with respect to the formula from the second theory being combined.

Finally, we recall the following theorem from [16].

Theorem 1 ([16]) *Let Σ_1 and Σ_2 be signatures and let $S = \mathcal{S}_{\Sigma_1} \cap \mathcal{S}_{\Sigma_2}$. If T_1 is a Σ_1-theory strongly polite w.r.t. $S_1 \subseteq \mathcal{S}_{\Sigma_1}$, T_2 is a Σ_2-theory strongly polite w.r.t. $S_2 \subseteq \mathcal{S}_{\Sigma_2}$, and $S \subseteq S_2$, then $T_1 \oplus T_2$ is strongly polite w.r.t. $S_1 \cup (S_2 \setminus S)$.*

3 Additive Witnesses

It was shown in [16] that politeness is not sufficient for the proof of the polite combination method from [23]. Strong politeness was introduced to fix the problem. In this section, we offer a simple (yet useful) criterion for the equivalence of the two notions. Throughout this section, unless stated otherwise, Σ and S denote an arbitrary signature and a subset of its set of sorts, and T, T_1, T_2 denote arbitrary Σ-theories.

The following example, which is based on [16] using notions of the current paper, shows that strong and non-strong witnesses are different. Let Σ_0 be a signature with a single sort σ and no function or predicate symbols (except $=_\sigma$), and T_0 the Σ_0-theory consisting of all Σ_0-structures $\mathcal{A}$ with $\left|\sigma^{\mathcal{A}}\right| \geq 2$. It was shown in [16] that the function *wtn* defined by $wtn(\phi) = (\phi \ \wedge \ w_1 = w_1 \ \wedge \ w_2 = w_2)$ for fresh w_1, w_2 is a witness for T_0 w.r.t. σ, but not a strong one. In fact, T_0 is also strongly polite since the function $wtn'(\phi) = \phi \wedge w_1 \neq w_2$ for fresh w_1, w_2 is a strong witness for T_0 w.r.t. σ. This was shown in [16].

We introduce the notion of additivity, which ensures that the witness is able to “absorb" arrangements and thus lift politeness to strong politeness.

Definition 10 (*Additivity*) Let $f : QF(\Sigma) \rightarrow QF(\Sigma)$. We say that f is S-additive for T if $f(f(\phi) \wedge \varphi)$ and $f(\phi) \wedge \varphi$ are T-equivalent and have the same set of S-sorted variables for every $\phi, \varphi \in QF(\Sigma)$, provided that φ is a conjunction of flat literals such that every term in φ is a variable whose sort is in S. When T is clear from the context, we say that f is S-additive. We say that T is *additively* finitely witnessable w.r.t. S if there exists a witness for T w.r.t. S which is both computable and S-additive. T is said to be *additively polite w.r.t. S* if it is smooth and additively finitely witnessable w.r.t. S.

We show that additive witnesses are strong:

Proposition 1 *Let wtn be a witness for T w.r.t. S. If wtn is S-additive then it is a strong witness for T w.r.t. S.*

Proof Let $\phi \in QF(\Sigma)$. We prove that $wtn(\phi)$ is strongly finitely witnessed for T w.r.t. S. Let V be a set of variables of sorts in S and δ_V an arrangement of V. We prove that $wtn(\phi) \wedge \delta_V$ is finitely witnessed for T w.r.t. S. Suppose it is T-satisfiable. Then since *wtn* is S-additive and δ_V is a conjunction of flat literals that contains only variables of sorts in S as terms, $wtn(wtn(\phi) \wedge \delta_V)$ is also T-satisfiable. *wtn* is a witness for T w.r.t. S, and hence $wtn(wtn(\phi) \wedge \delta_V)$ has a finite witness $\mathcal{A}$ for T w.r.t. S. By T-equivalence, $\mathcal{A} \models wtn(\phi) \wedge \delta_V$. Since both formulas have the same set of S-variables, $\mathcal{A}$ is also a finite witness of $wtn(\phi) \wedge \delta_V$. □

Corollary 1 *An additively polite theory w.r.t. S is strongly polite w.r.t. S.*

The theory T_0 from above is additively finitely witnessable w.r.t. σ, even though wtn' is not σ-additive. However, it is possible to define a new witness for T_0 w.r.t. σ, say wtn'', which is σ-additive. wtn'' is defined by: $wtn''(\phi) = \phi$ if ϕ is a conjunction that includes some disequality $x \neq y$ for some x, y. Otherwise, $wtn''(\phi) = wtn'(\phi)$.

The following definition generalizes the theory T_0.

Definition 11 (*Existential Theory*) We say a Σ-theory T is *existential* if there exists a sentence of the form $\phi = \exists \overline{x}.\varphi$ where φ is quantifier-free, such that T consists of all the Σ-structures that satisfy ϕ.

T_0 is an *existential* theory, with the sentence $\exists x, y \,.\, x \neq y$. Similarly, minimal finite cardinality constraints can be axiomatized with an existential sentence. The construction of wtn'' above can be generalized to any *existential* theory. Such theories are also smooth w.r.t. any set of sorts and so *existential* theories are additively (and thus strongly) polite:

Proposition 2 *If T is existential then it is strongly polite w.r.t. S.*

Proof Let φ be the formula whose existential closure defines T. Define a function wtn_T by

$$wtn_T(\phi) = \begin{cases} \phi & \text{if } \phi = \phi' \wedge \varphi' \\ \phi \wedge \varphi'' & \text{otherwise} \end{cases}$$

where ϕ' is a quantifier-free formula, φ' is obtained from φ by replacing its variables with variables not in $vars(\phi')$, and φ'' is obtained from φ by replacing its variables with variables not in $vars(\phi)$. By construction, wtn_T is S-additive: once an instance of φ' was added to the formula, further applications of wtn_T will not change the input formula. wtn_T is also a witness for T w.r.t. S: if $wtn_T(\phi) = \phi$ then the equivalence requirement is trivial. Otherwise, ϕ is T-equivalent to $\exists \overline{w}.wtn_T(\phi)$, where $\overline{w}$ are the fresh variables that were introduced, since the latter only adds an existential formula that is T-valid. Further, if we restrict the domain of a T-interpretation that satisfies $wtn_T(\phi)$, we still obtain a T-interpretation, as the existential closure of φ logically follows from any instance of φ.

For smoothness, let ϕ be a quantifier-free formula, $\mathcal{A}$ a T-interpretation that satisfies ϕ, and κ a function as in Definition 7. Since T is defined by the existential closure of a quantifier-free formula φ, augmenting $\sigma^{\mathcal{A}}$ for each $\sigma \in S$ so that its cardinality matches $\kappa(\sigma)$ results in another T-interpretation satisfying ϕ. □

The notion of additive witnesses is useful for proving that a polite theory is strongly polite. In particular, the witnesses for the theories of equality, arrays, sets, and multisets from [23] are all additive, and so strong politeness of these theories follows from their politeness. The same will hold later, when we conclude strong politeness of theories of algebraic datatypes from their politeness.

4 Politeness for the SMT-LIB 2 Theory of Datatypes

Let Σ be a datatypes signature with $\mathcal{S}_\Sigma = \mathbf{Elem} \uplus \mathbf{Struct}$ and $\mathcal{F}_\Sigma = \mathcal{CO} \uplus \mathcal{SE}$. In this section, we prove that $\mathcal{T}_\Sigma$ is strongly polite with respect to **Elem**. In Sect. 4.1, we consider theories with only inductive sorts, and consider theories with only finite sorts in Sect. 4.2. We combine them in Sect. 4.3, where arbitrary theories of datatypes are considered. This separation is only needed for finite witnessability, but not for smoothness:

Lemma 1 *$\mathcal{T}_\Sigma$ is smooth w.r.t.* **Elem**.

Proof Let ϕ be a quantifier-free Σ formula, and let $\mathcal{A}$ be a $\mathcal{T}_\Sigma$-interpretation that satisfies ϕ. Let κ be a function from **Elem** to the class of cardinals such that $\kappa(\sigma) \geq \left|\sigma^{\mathcal{A}}\right|$ for every $\sigma \in$ **Elem**. Then let $\mathcal{A}'$ be augmented from $\mathcal{A}$ by adding elements to $\sigma^{\mathcal{A}}$ to match $\kappa(\sigma)$. This is possible because the sorts of **Elem** are never the range of any constructor. Such an $\mathcal{A}'$ exists so that the interpretations of variables and selectors in ϕ remain intact, and for such $\mathcal{A}'$, we have $\mathcal{A}' \models \phi$. □

4.1 Inductive Datatypes

In this section, we assume that all sorts in **Struct** are inductive.

To prove finite witnessability, we now introduce an additive witness function. Following arguments from [23], it suffices to define the witness only for conjunctions of flat literals. A complete witness can then use the restricted one by first transforming the input formula to flat DNF form and then creating a disjunction where each disjunct is the result of applying the witness on the corresponding disjunct. Similarly, it suffices to show that $wtn(\phi)$ is finitely witnessed for ϕ which is a conjunction of flat literals. Essentially, our witness guesses possible constructors for variables whose constructors are not explicit in the input formula.

Definition 12 (*A Witness for* $\mathcal{T}_\Sigma$) Let ϕ be a quantifier-free conjunction of flat Σ-literals. $wtn_i(\phi)$ is obtained from ϕ by performing the following steps:

1. For any literal of the form $y = s_{c,i}(x)$ such that $x = d(\overrightarrow{u_d})$ does not occur in ϕ for any d and $\overrightarrow{u_d}$, we conjunctively add $x = c(\overrightarrow{u_1}, y, \overrightarrow{u_2}) \vee (\bigvee_{d \neq c, d \in \mathcal{CO}} x = d(\overrightarrow{u_d}))$ where u_1 is a list of i-1 fresh variables, u_2 is a list of $n - i$ fresh variables with n being the number of arguments of c, u_d is a list of m fresh variables with m being the number of arguments of d for each d, and y is a fresh variable. All fresh variables are sorted according to the arities of c and the d's.
2. For any literal of the form $is_c(x)$ such that $x = c(\overrightarrow{u})$ does not occur in ϕ for any $\overrightarrow{u}$, we conjunctively add $x = c(\overrightarrow{u})$ with fresh $\overrightarrow{u}$.
3. For any literal of the form $\neg is_c(x)$ such that $x = d(\overrightarrow{u_d})$ does not occur in ϕ for any $d \neq c$ and $\overrightarrow{u_d}$, we conjunctively add $\bigvee_{d \neq c} x = d(\overrightarrow{u_d})$, with fresh $\overrightarrow{u_d}$.
4. For any sort $\sigma \in$ **Elem** such that ϕ does not include a variable of sort σ we conjunctively add a literal $x = x$ for a fresh variable x of sort σ.

Example 5 Let ϕ be the Σ_{list}-formula $y = cdr(x) \wedge y' = cdr(x) \wedge is_{cons}(y)$. $wtn_i(\phi)$ is $\phi \wedge (x = nil \vee x = cons(e, y)) \wedge (x = nil \vee x = cons(e', y')) \wedge y = cons(e'', z) \wedge e''' = e'''$ where e, e', e'', e''', z are fresh.

in Definition 12, Item 1 guesses the constructor of the argument for the selector. Items 2 and 3 correspond to the semantics of testers. Item 4 is meant to ensure that we can construct a finite witness with non-empty domains. The requirement for absence of literals before adding literals or disjunctions to ϕ is used to ensure additivity of wtn_i. And indeed:

Lemma 2 *wtn_i is* **Elem**-*additive for T_Σ.*

Proof For input formulas that are conjunctions of flat literals, this follows from the construction of wtn_i. For arbitrary quantifier-free formulas, as mentioned before Definition 12, wtn_i is extended from conjunctions of flat literals to arbitrary quantifier-free formulas by transforming the input formula to flat DNF form and then applying the witness on each disjunct of the DNF, taking the disjunction of these applications. We prove that this extension preserves additivity.

Let ϕ be a quantifier-free Σ-formula, $D_1 \vee \ldots \vee D_m$ its flat DNF form, and φ a conjunction of flat literals such that every term in φ is a variable whose sort is in **Elem**. By the above, $wtn_i(wtn_i(\phi) \wedge \varphi) = wtn_i(wtn_i(D_1 \vee \ldots \vee D_m) \wedge \varphi) = wtn_i((wtn_i(D_1) \vee \ldots \vee wtn_i(D_m)) \wedge \varphi)$. For each $1 \leq i \leq m$, let $E_i^1 \vee \ldots \vee E_i^{k_i}$ be the flat DNF form of $wtn_i(D_i)$. Since wtn_i does not introduce non-flat literals, no new variables are introduced in the transformation from $wtn_i(D_i)$ to $E_i^1 \vee \ldots \vee E_i^{k_i}$, but only propositional transformations are employed. The equation list above can continue with $wtn_i((E_1^1 \wedge \varphi) \vee \ldots \vee (E_m^{k_m} \wedge \varphi)) = wtn_i(E_1^1 \wedge \varphi) \vee \ldots \vee wtn_i(E_m^{k_m} \wedge \varphi)$. Now, for each $1 \leq i \leq m$ and $1 \leq j \leq k_i$, E_i^j is a conjunction of flat literals in the DNF form of $wtn_i(D_i)$. By the construction of wtn_i, each such $E_i^j \wedge \varphi$ does not satisfy any of the preconditions in wtn_i for the addition of any formula: the literals already exist in E_i^j after the first application of wtn_i over D_i. In addition, φ does not contain any constructors and testers. Also, each conjunction in the DNF includes at least one variable of each **Elem**-sort. Thus $wtn_i(E_i^j \wedge \varphi) = E_i^j \wedge \varphi$. This means that $wtn_i(wtn_i(\phi) \wedge \varphi) = (E_1^1 \wedge \varphi) \vee \ldots \vee (E_m^{k_m} \wedge \varphi)$.

Similarly, $wtn_i(\phi) \wedge \varphi = (wtn_i(D_1) \vee \ldots wtn_i(D_m)) \wedge \varphi$, which is logically equivalent to $(E_1^1 \vee \ldots \vee E_m^{k_m}) \wedge \varphi$, and hence to $(E_1^1 \wedge \varphi) \vee \ldots \vee (E_m^{k_m} \wedge \varphi)$, which by the above is equivalent to $wtn_i(wtn_i(\phi) \wedge \varphi)$. Further, since the second application of wtn_i does not introduce anything new, the set of **Elem**-variables is the same in both formulas.

□

Further, the equivalence constraint is satisfied:

Lemma 3 *Let ϕ be a conjunction of flat literals. ϕ and $\exists \overrightarrow{w}\,.\,\Gamma$ are $\mathcal{T}_\Sigma$-equivalent, where $\Gamma = wtn_i(\phi)$ and $\overrightarrow{w} = vars(\Gamma) \setminus vars(\phi)$.*

Proof Each variable in $\overrightarrow{w}$ occurs exactly once in Γ. Let Γ' be obtained from $\exists \overrightarrow{w}.\Gamma$ by pushing each existential quantifier to the literal that contains its corresponding quantified variable. Clearly, $\exists \overrightarrow{w}\,\Gamma$ and Γ' are logically equivalent, and in particular they are $\mathcal{T}_\Sigma$-equivalent. Γ' contains all the conjuncts of ϕ as top-level conjuncts. Hence clearly every $\mathcal{T}_\Sigma$-interpretation that satisfies Γ' also satisfies ϕ. For the converse, let $\mathcal{A}$ be a $\mathcal{T}_\Sigma$-interpretation that satisfies ϕ and Δ a top-level conjunct of Γ'.

- If Δ is also a literal of ϕ then $\mathcal{A} \models \Delta$.
- If Δ corresponds to a formula that was added by Item 1 of Definition 12, then it has the form $(\exists \overrightarrow{u_1}\, y \overrightarrow{u_2}.x = c(\overrightarrow{u_1}, y, \overrightarrow{u_2})) \vee (\bigvee_{d \neq c} \exists \overrightarrow{u_d}.x = d(\overrightarrow{u_d}))$ and $y = s_{c,i}(x)$ is a literal of ϕ. $\mathcal{A} \models y = s_{c,i}(x)$. If $\mathcal{A} \models is_c(x)$ then it must satisfy the first disjunct of Δ. Otherwise, $\mathcal{A}$ must satisfy one of the other disjuncts. In both cases, $\mathcal{A} \models \Delta$.
- If Δ corresponds to a formula that was added by Item 1 of Definition 12 then it has the form $\exists \overrightarrow{u}\,.x = c(\overrightarrow{u})$ and $is_c(x)$ is a literal of ϕ. Since $\mathcal{A} \models is_c(x)$, we must have $\mathcal{A} \models \Delta$.
- If Δ corresponds to a formula that was added by Item 2 of Definition 12 then it has the form $\bigvee_{d \neq c} \exists \overrightarrow{u}\,.x = d(\overrightarrow{u})$ and $\neg is_c(x)$ is in ϕ. Since $\mathcal{A} \not\models is_c(x)$, we must have $\mathcal{A} \models \Delta$.
- If Δ corresponds to a formula that was added by Item 3 then it is trivially satisfied.

□

The remainder of this section is dedicated to the proof of the following lemma:

Lemma 4 (Finite Witnessability) *Let ϕ be a conjunction of flat literals. Then, $\Gamma = wtn_i(\phi)$ is finitely witnessed for $\mathcal{T}_\Sigma$ with respect to* **Elem**.

Suppose that Γ is $\mathcal{T}_\Sigma$-satisfiable, and let $\mathcal{A}$ be a satisfying $\mathcal{T}_\Sigma$-interpretation. We define a $\mathcal{T}_\Sigma$-interpretation $\mathcal{B}$ as follows and then show that $\mathcal{B}$ is a finite witness of Γ for $\mathcal{T}_\Sigma$ w.r.t. **Elem**.

4.1.1 Construction of $\mathcal{B}$

We start by defining an interpretation $\mathcal{B}$. For every $\sigma \in$ **Elem** we set:

$$\sigma^{\mathcal{B}} = vars_{\sigma}(\Gamma)^{\mathcal{A}} \tag{1}$$

For every variable $e \in vars_{\sigma}(\Gamma)$ with $\sigma \in$ **Elem**, we set:

$$e^{\mathcal{B}} = e^{\mathcal{A}} \tag{2}$$

The interpretations of **Struct**-sorts, testers, and constructors are uniquely determined by the theory, as they are generated by the signature and the interpretation of **Elem** in $\mathcal{A}$.

It is therefore left to define new values for the interpretations of **Struct**-variables in $\mathcal{B}$, as well as the interpretation of the selectors. For the former, they might have been interpreted in $\mathcal{A}$ using (discarded) constructors. For the latter, their interpretations need to correspond to the new interpretations in $\mathcal{B}$. We do this in several steps:

Step 1—Simplifying Γ since ϕ is a conjunction of flat literals, Γ is a conjunction whose conjuncts are either flat literals or disjunctions of flat literals (introduced in Items 1 and 3 of Definition 12). Since $\mathcal{A} \models \Gamma$, $\mathcal{A}$ satisfies at least one disjunct of each such disjunction. By the definition of wtn_i, exactly one such disjunct is satisfied. We can thus obtain a formula Γ_1 from Γ by replacing every disjunction with the unique disjunct that is satisfied by $\mathcal{A}$. Notice that $\mathcal{A} \models \Gamma_1$ and that it is a conjunction of flat literals. Let Γ_2 be obtained from Γ_1 by removing any literal of the form $is_c(x)$ and any literal of the form $\neg is_c(x)$. Let Γ_3 be obtained from Γ_2 by removing any literal of the form $x = s_{c,i}(y)$. For convenience, we denote Γ_3 by Γ'. Note that the predicate and selector terms are redundant for Γ since they have been expanded to constructor terms by the witness function. Obviously, $\mathcal{A} \models \Gamma'$, and Γ' is a conjunction of flat literals without selectors and testers.

Step 2—Working with Equivalence Classes We would like to preserve equalities between **Struct**-variables from $\mathcal{A}$. To this end, we group all variables in $vars(\Gamma)$ to equivalence classes according to their interpretation in $\mathcal{A}$. Let $\equiv_{\mathcal{A}}$ denote the equivalence relation over $vars(\Gamma)$ such that $x \equiv_{\mathcal{A}} y$ iff $x^{\mathcal{A}} = y^{\mathcal{A}}$. We denote by $[x]$ the equivalence class of x. Let α be an equivalence class, thus $\alpha^{\mathcal{A}} = \left\{x^{\mathcal{A}} \mid x \in \alpha\right\}$ is a singleton. Identifying this singleton with its only element, we have that $\alpha^{\mathcal{A}}$ denotes $a^{\mathcal{A}}$ for an arbitrary element a of the equivalence class α.

Step 3—Ordering Equivalence Classes We would also like to preserve disequalities between **Struct**-variables from $\mathcal{A}$. Thus we introduce a relation $\prec$ over the equivalence classes: $\alpha \prec \beta$ if $y = c(w_1, \ldots, w_n)$ occurs as one of the conjuncts in Γ' for some $w_1, \ldots, w_n$ and $c \in \mathcal{CO}$ such that $w_k \in \alpha$ for some $k \in [1, n]$ and $y \in \beta$.

An equivalence class α is *nullary* if $\mathcal{A} \models is_c(x)$ for some $x \in \alpha$ and nullary constructor c. An equivalence class α is *minimal* if $\beta \not\prec \alpha$ for every β. Notice that each nullary equivalence class is minimal. The relation $\prec$ induces a directed acyclic graph (DAG), denoted as G. The vertices are the equivalence classes. Whenever $\alpha \prec \beta$, we draw an edge from vertex α to β.

Step 4—Interpretation of Equivalence Classes We next define $\alpha^{\mathcal{B}}$ for every equivalence class α. Then, for every **Struct**-variable x, we set:

$$x^{\mathcal{B}} = [x]^{\mathcal{B}} \tag{3}$$

The idea for defining $\alpha^{\mathcal{B}}$ goes as follows. Nullary classes are assigned according to $\mathcal{A}$, because nullary constructors are interpreted as themselves, and hence there is only one way to interpret them. Other minimal classes are assigned arbitrarily, but it is important to assign different classes to terms whose depths are far enough from each other to ensure that the disequalities in $\mathcal{A}$ are preserved. Non-minimal classes are uniquely determined after minimal ones are assigned.

Formally, let m be the number of equivalence classes, l the number of minimal equivalence classes, r the number of nullary equivalence classes, and $\alpha_1, \ldots, \alpha_m$ a topological sort of G, such that all minimal classes occur before all others, and the first r classes are nullary. Let d be the length of the longest path in G. We define $\alpha_i^{\mathcal{B}}$ by induction on i. In the definition, we use $\mathcal{B}_{\mathbf{Elem}}$ to denote the **Elem**-sorted set assigning $\sigma^{\mathcal{B}}$ to every $\sigma \in \mathbf{Elem}$.

1. If $0 < r$ and $i \leq r$ then α_i is a nullary class and so we set:
$$\alpha_i^{\mathcal{B}} = \alpha_i^{\mathcal{A}} \tag{4}$$
2. If $r < i \leq l$ then α_i is minimal and not nullary. Let σ be the sort of variables in α_i. If $\sigma \in \mathbf{Elem}$, then all variables in the class have already been defined. Otherwise, $\sigma \in \mathbf{Struct}$. In this case, we set:
$$\alpha_i^{\mathcal{B}} = a \tag{5}$$
such that a is some arbitrary element of $T_\sigma(\Sigma_{|\mathcal{CO}}, \mathcal{B}_{\mathbf{Elem}})$ that has depth strictly greater than $\max\left\{depth(\alpha_j^{\mathcal{B}}) \mid 0 < j < i\right\} + d$ (here $\max \emptyset = 0$).
3. If $i > l$ then we set:
$$\alpha_i^{\mathcal{B}} = c(\beta_1^{\mathcal{B}}, \ldots, \beta_n^{\mathcal{B}}) \tag{6}$$
for the unique equivalence classes $\beta_1, \ldots, \beta_n \subseteq \{\alpha_1, \ldots, \alpha_{i-1}\}$ and c such that $y = c(x_1, \ldots, x_n)$ occurs in Γ' for some $y \in \alpha_i$ and $x_1 \in \beta_1, \ldots, x_n \in \beta_n$.

Example 6 Let Γ be the following Σ_{list}-formula: $x_1 = cons(e_1, x_2) \wedge x_3 = cons(e_2, x_4) \wedge x_2 \neq x_4$. Then $\Gamma' = \Gamma$. We have the following satisfying interpretation $\mathcal{A}$: $\mathbf{elem}^{\mathcal{A}} = \{1, 2, 3, 4\}$, $e_1^{\mathcal{A}} = 1$, $e_2^{\mathcal{A}} = 2$, $x_1^{\mathcal{A}} = [1, 2, 3]$, $x_2^{\mathcal{A}} = [2, 3]$, $x_3^{\mathcal{A}} = [2, 2, 4]$, $x_4^{\mathcal{A}} = [2, 4]$.

The construction above yields the following interpretation $\mathcal{B}$: $\mathbf{elem}^{\mathcal{B}} = \{1, 2\}$, $e_1^{\mathcal{B}} = 1$, $e_2^{\mathcal{B}} = 2$. For **list**-variables, we proceed as follows. The equivalence classes of **list**-variables are $[x_1], [x_2], [x_3], [x_4]$, with $[x_2] \prec [x_1]$ and $[x_4] \prec [x_3]$. The length of the longest path in G is 1.

Assuming $[x_2]$ comes before $[x_4]$ in the topological sort, $x_2^{\mathcal{B}}$ will get an arbitrary list over $\{1, 2\}$ with length greater than 1 (the depth of $e_2^{\mathcal{B}}$ plus the length of the longest path), say, $[1, 1, 1]$. $x_4^{\mathcal{B}}$ will then get an arbitrary list of length greater than 4 (the depth of $x_2^{\mathcal{B}}$ plus the length of the longest path). Thus we could have $x_4^{\mathcal{B}} = [1, 1, 1, 1, 1]$. Then, $x_1^{\mathcal{B}} = [1, 1, 1, 1]$ and $x_3^{\mathcal{B}} = [2, 1, 1, 1, 1, 1]$.

Lemma 5 *$\alpha_i^{\mathcal{B}}$ is well defined.*

Proof The case of nullary and minimal constructors is clearly well defined. Suppose α_i is not minimal. Then the sort of its variables is in **Struct**. We prove that there is a unique list $\beta_1, \ldots, \beta_n$, of equivalence classes, all elements of $\{\alpha_1, \ldots, \alpha_{i-1}\}$ and a unique constructor c such that $y = c(x_1, \ldots, x_n)$ occurs in Γ' for some $y \in \alpha_i$ and $x_1 \in \beta_1, \ldots, x_n \in \beta_n$. **Existence:** α_i is not minimal. Hence, there exists some β_1 such that $\beta_1 \prec \alpha_i$. Hence w.l.g. there exists some $y \in \alpha_i$ and some $x_1 \in \beta_1$ such that $y = c(x_1, x_2, \ldots, x_n)$ is in Γ' for some $x_2, \ldots, x_n$ and c. By definition, this means that $[x_2], \ldots, [x_n] \prec \alpha_i$ as well, and thus $[x_j]$ must occur before α_i in the topological ordering for every $1 \leq j \leq n$, hence $[x_j] \in \{\alpha_1, \ldots, \alpha_{i-1}\}$ for each j. **Uniqueness:** Suppose there are also equivalence classes $\beta_1', \ldots, \beta_m'$, all elements of $\{\alpha_1, \ldots, \alpha_{i-1}\}$, and a constructor c' such that $y' = c'(x_1', \ldots, x_m')$ occurs in Γ' for some

$y' \in \alpha_i$ and $x'_1 \in \beta'_1, \ldots, x'_m \in \beta'_m$. Since $y' = c'(x'_1, \ldots, x'_m)$ and $y = c(x_1, \ldots, x_n)$ both occur in Γ' and are thus satisfied by $\mathcal{A}$, and $[y] = [y']$, we must have $c = c'$, $n = m$, and $\mathcal{A} \models x_j = x'_j$ for every j; otherwise, it would contradict $[y] = [y']$. Hence $[x_j] = [x'_j]$, so that $\beta'_j = \beta_j$ for every j. □

Step 5—Interpretation of Selectors Let $s_{c,i} \in \mathcal{SE}$ for $c : \sigma_1 \times \cdots \times \sigma_n \to \sigma$, $1 \leq i \leq n$ and $a \in \sigma^{\mathcal{B}}$. If $a \in is_c^{\mathcal{B}}$, we must have $a = c(a_1, \ldots, a_n)$ for some $a_1 \in \sigma_1^{\mathcal{B}}, \ldots, a_n \in \sigma_n^{\mathcal{B}}$. We then set:

$$s_{c,i}^{\mathcal{B}}(a) = a_i \tag{7}$$

Otherwise, we consider two cases. If $x^{\mathcal{B}} = a$ for some $x \in vars(\Gamma)$ such that $y = s_{c,i}(x)$ occurs in Γ_2 for some y, we set:

$$s_{c,i}^{\mathcal{B}}(a) = y^{\mathcal{B}} \tag{8}$$

Otherwise, $s_{c,i}^{\mathcal{B}}(a)$ is set arbitrarily.

4.1.2 $\mathcal{B}$ is a Finite Witness of Γ

Now that $\mathcal{B}$ is defined using Eqs. 1–8, we show that it is a finite witness of Γ for $\mathcal{T}_\Sigma$ w.r.t. **Elem**. By construction, $\sigma^{\mathcal{B}} = vars_\sigma(\Gamma)^{\mathcal{B}}$ for every $\sigma \in$ **Elem**. Hence it is left to show that $\mathcal{B} \models \Gamma$. We start by showing that $\mathcal{B}$ preserves the equalities and disequalities in $\mathcal{A}$:

Lemma 6 *$x^{\mathcal{A}} = y^{\mathcal{A}}$ iff $x^{\mathcal{B}} = y^{\mathcal{B}}$ for every $x, y \in vars(\Gamma)$.*

Proof The left-to-right direction follows directly from the definition of $\mathcal{B}$ that does not distinguish distinct elements inside a single equivalence class of $\equiv_{\mathcal{A}}$. For the converse, we prove that $\alpha_1^{\mathcal{B}}, \ldots, \alpha_p^{\mathcal{B}}$ are pairwise distinct for every $1 \leq p \leq m$ by induction on p. From this, the claim follows: if $x^{\mathcal{A}} \neq y^{\mathcal{A}}$, then $[x] = \alpha_p$ and $[y] = \alpha_q$ for some $p \neq q$, and therefore $x^{\mathcal{B}} = [x]^{\mathcal{B}} \neq [y]^{\mathcal{B}} = y^{\mathcal{B}}$.

The induction base corresponds to the first l classes (minimal classes).

1. For all the equivalence classes of **Elem**-sorted variables, as they are also minimal, and the definition is the same as in $\mathcal{A}$, their interpretations are distinct by definition.
2. For the nullary classes, the definition is also the same as in $\mathcal{A}$, thus they have distinct interpretations.
3. For the equivalence classes of minimal non-nullary **Struct**-sorted variables, they have different interpretations with the nullary classes, as their interpretations all have the depth more than d. And among themselves, the depths of the interpretations of these classes correspond to a strongly increasing monotonic sequence by definition.

For the induction step, assume the claim for p ($l \leq p < m$) vertices. It is sufficient to prove that α_{p+1} has a different interpretation from all the previous vertices. Assume otherwise, and let $i \leq p$ with $\alpha_i^{\mathcal{B}} = \alpha_{p+1}^{\mathcal{B}}$.

α_{p+1} is not minimal. Since α_{p+1} cannot be nullary, $\alpha_i^{\mathcal{B}} = \alpha_{p+1}^{\mathcal{B}}$ cannot be nullary, thus we have $i > r$. Recall that the first r classes are nullary as defined in Step 4. Then let us consider two cases.

1. α_i is not minimal: There must be a constructor c such that $\alpha_i^{\mathcal{B}} = c(\beta_1^{\mathcal{B}}, \ldots, \beta_n^{\mathcal{B}})$ and $\alpha_{p+1}^{\mathcal{B}} = c(\hat{\beta}_1^{\mathcal{B}}, \ldots, \hat{\beta}_n^{\mathcal{B}})$ for some equivalence classes $\beta_1, \ldots, \beta_n$ and $\hat{\beta}_1, \ldots, \hat{\beta}_n$. Then from $\alpha_i^{\mathcal{B}} = \alpha_{p+1}^{\mathcal{B}}$, we have $\beta_k^{\mathcal{B}} = \hat{\beta}_k^{\mathcal{B}}$ for $k = 1, \ldots, n$. Also, note that

$\beta_1, \ldots, \beta_n, \hat{\beta}_1, \ldots, \hat{\beta}_n \in \{\alpha_1, \ldots, \alpha_p\}$. Let $1 \leq k \leq n$. By the induction hypothesis, either $\beta_k = \hat{\beta}_k$ or $\beta_k^{\mathcal{B}} \neq \hat{\beta}_k^{\mathcal{B}}$. By the above, the former must hold. So that $\beta_k^{\mathcal{A}} = \hat{\beta}_k^{\mathcal{A}}$ for $k = 1, \ldots, n$, thus we get $\alpha_i^{\mathcal{A}} = \alpha_{p+1}^{\mathcal{A}}$. Since the equivalence classes are defined by $\equiv_{\mathcal{A}}$, we have $[\alpha_i] = [\alpha_{p+1}]$, thus $i = p + 1$. This contradicts the fact that $i < p + 1$.

2. α_i is minimal: An equivalence class β is said to be a source of α_{p+1}, if there is a path from β to α_{p+1} in G and β is minimal.
   If α_{p+1} has a source vertex β such that $depth(\beta^{\mathcal{B}}) \geq depth(\alpha_i^{\mathcal{B}})$, then we have $depth(\alpha_{p+1}^{\mathcal{B}}) > depth(\beta^{\mathcal{B}}) \geq depth(\alpha_i^{\mathcal{B}})$.
   Otherwise $depth(\alpha_i^{\mathcal{B}}) > 0$ since by construction, α_i is not an **Elem**-sorted variable, and for any other minimal class α', either $depth(\alpha'^{\mathcal{B}}) > depth(\alpha_i^{\mathcal{B}})$ or $depth(\alpha'^{\mathcal{B}}) < depth(\alpha_i^{\mathcal{B}}) - d$. So for any source vertex β of α_{p+1}, we have $depth(\beta^{\mathcal{B}}) < depth(\alpha_i^{\mathcal{B}}) - d$. Since d is the length of the longest path, we obtain $depth(\alpha_{p+1}^{\mathcal{B}}) \leq depth(\beta^{\mathcal{B}}) + d < depth(\alpha_i^{\mathcal{B}})$. Therefore, $\alpha_i^{\mathcal{B}} \neq \alpha_{p+1}^{\mathcal{B}}$, which makes the contradiction. □

By considering every shape of a literal in Γ', we can prove that $\mathcal{B} \models \Gamma'$. Then, our interpretation of the selectors ensures the following:

Lemma 7 $\mathcal{B} \models \Gamma$.

Proof We start by proving that $\mathcal{B} \models \Gamma'$. Γ' is a conjunction of flat literals without selectors and testers. We consider each type of conjunct separately.

- Literals of the form $x = y$ or $x \neq y$: By Lemma 6, and the fact that $\mathcal{A} \models \Gamma'$, these literals hold in interpretation $\mathcal{B}$.
- Literals of the form $x = c$, where c is a nullary constructor: In this case, $x^{\mathcal{B}}$ is defined as $x^{\mathcal{A}}$, see Eq. 4. Since $\mathcal{A} \models \Gamma'$, we have $\mathcal{B} \models x = c$.
- Literals of the form $x = c(w_1, \ldots, w_n)$ for some constructor c: Since $\mathcal{A} \models \Gamma'$, c is the only constructor that construct x in Γ'. From the definition of $\mathcal{B}$, $x^{\mathcal{B}} = c(d_1^{\mathcal{B}}, \ldots, d_n^{\mathcal{B}})$ for some $d_1, \ldots, d_n$, see Eq. 6. And by Lemma 5, we have $[w_k] = [d_k]$ for $k = 1, \ldots, n$. So we have $x^{\mathcal{B}} = c(w_1^{\mathcal{B}}, \ldots, w_n^{\mathcal{B}})$.

Next, we prove that $\mathcal{B} \models \Gamma_2$. Γ_2 is a conjunction of the literals of Γ', together with literals of the form $y = s_{c,i}(x)$ from Γ. Let $y = s_{c,i}(x)$ be such a conjunct of Γ_2. Then by the definition of wtn_i and Γ', there are two cases:

- $x = c(\ldots, y, \ldots)$ is in Γ'. Thus $[y] \prec [x]$ and $x^{\mathcal{B}} = c(\ldots, y^{\mathcal{B}}, \ldots)$ by the definition of $\mathcal{B}$. In particular, $x^{\mathcal{B}} \in is_c^{\mathcal{B}}$. In this case, $s_{c,i}(x)^{\mathcal{B}}$ is set to $y^{\mathcal{B}}$ by the definition of $\mathcal{B}$.
- $x = d(\ldots)$ is in Γ' for some $d \neq c$. We consider the following subcases.
  - If d is nullary then $[x]$ is nullary. In this case, $x^{\mathcal{B}} = x^{\mathcal{A}}$. $\mathcal{A} \models \Gamma'$ and hence $x^{\mathcal{A}} \in is_d^{\mathcal{A}}$, which means that $x^{\mathcal{B}} \in is_d^{\mathcal{B}}$ as well. In particular, $x^{\mathcal{B}} \notin is_c^{\mathcal{B}}$. Since $y = s_{c,i}(x)$ occurs in Γ_2, $s_{c,i}(x)^{\mathcal{B}}$ is set to be $y^{\mathcal{B}}$.
  - If d is not nullary then $[x]$ cannot be minimal, and hence $x^{\mathcal{B}} \in is_d^{\mathcal{B}}$ by the definition of $\mathcal{B}$. In particular, $x^{\mathcal{B}} \notin is_c^{\mathcal{B}}$. Since $y = s_{c,i}(x)$ occurs in Γ_2, $s_{c,i}(x)^{\mathcal{B}}$ is set to be $y^{\mathcal{B}}$ in this case.

Hence $\mathcal{B} \models \Gamma_2$.

Next, we show that $\mathcal{B} \models \Gamma_1$, which is obtained from Γ_2 by the addition of conjunctions of the form $is_c(x)$ and $\neg is_c(x)$. Let $is_c(x)$ be such a literal in Γ_1. Then it is also a literal of Γ. Then by the definition of wtn_i and of Γ', this means that Γ' contains a literal of the form

$x = c(y_1, \ldots, y_n)$. Since $\mathcal{B} \models \Gamma'$, we have $\mathcal{B} \models is_c(x)$. Now let $\neg is_c(x)$ be a literal of Γ_1. Then it is also a literal of Γ. By the definition of wtn_i and Γ', the latter contains a literal of the form $x = d(t_1, \ldots, t_n)$ for some $d \neq c$. Since $\mathcal{B} \models \Gamma'$, we have $\mathcal{B} \models \neg is_c(x)$.

Finally, we have seen that $\mathcal{B}$ satisfies a disjunct in every disjunction of Γ, as well as all of the top-level literals of Γ, which means that $\mathcal{B} \models \Gamma$. □

Lemmas 3 and 7, together with the definition of the domains of $\mathcal{B}$, give us that $\mathcal{B}$ is a finite witness of Γ for $\mathcal{T}_\Sigma$ w.r.t. **Elem**, and so Lemma 4 is proven. As a consequence of Lemmas 1, 2 and 4, strong politeness is obtained.

Theorem 2 *If Σ is a datatypes signature and all sorts in* **Struct**$_\Sigma$ *are inductive, then $\mathcal{T}_\Sigma$ is strongly polite w.r.t.* **Elem**$_\Sigma$.

4.2 Finite Datatypes

In this section, we assume that all sorts in **Struct** are finite.

For finite witnessability, we define the following witness that guesses the construction of each **Struct**-variables until a fixpoint is reached.

Definition 13 (*A Witness for $\mathcal{T}_\Sigma$*) For every quantifier-free conjunction of flat Σ-literals ϕ, define the sequence $\phi_0, \phi_1, \ldots$, such that $\phi_0 = \phi$, and for every $i \geq 0$, ϕ_{i+1} is obtained from ϕ_i by conjuncting it with a disjunction $\bigvee_{c \in \mathcal{CO}} x = c(w_1^c, \ldots, w_{n_c}^c)$ for fresh $w_1^c, \ldots, w_{n_c}^c$, where x is some arbitrary **Struct**-variable in ϕ_i such that there is no literal of the form $x = c(y_1, \ldots, y_n)$ in ϕ_i for any constructor $c \in \mathcal{CO}$ and variables $y_1, \ldots, y_n$. Since **Struct** only has finite sorts, there is necessarily a minimal k such that $\phi_k = \phi_{k+1}$ and $wtn_f(\phi)$ is defined to be ϕ_k.

Example 7 Let ϕ be the Σ_{pair}-formula $x = \mathit{first}(y) \wedge x' = \mathit{first}(y') \wedge x \neq x'$. $wtn_f(\phi)$ is $\phi \wedge y = pair(e_1, e_2) \wedge y' = pair(e_3, e_4)$.

Similarly to the case of inductive datatypes presented in Sect. 4.1, we have:

Lemma 8 *wtn_f is* **Elem**-*additive for $\mathcal{T}_\Sigma$.*

Proof We proceed just like in the proof of Lemma 2. Let φ be a conjunction of flat literals such that every term in φ is a variable whose sort is in **Elem**. By construction of wtn_f, $wtn_f(\phi) \wedge \varphi$ does not satisfy the precondition in wtn_f for the addition of any formula since φ does not contain any constructors. Thus, $wtn_f(wtn_f(\phi) \wedge \varphi) = wtn_f(\phi) \wedge \varphi$. □

Lemma 9 *ϕ and $\exists \overrightarrow{w} \,.\, wtn_f(\phi)$ are $\mathcal{T}_\Sigma$-equivalent, where $\overrightarrow{w} = vars(wtn_f(\phi)) \setminus vars(\phi)$.*

Proof Similarly to the proof of Lemma 3, we know that $\exists \overrightarrow{w_i} \,.\, \phi_i$ and $\exists \overrightarrow{w_{i+1}} \,.\, \exists \phi_{i+1}$ are $\mathcal{T}_\Sigma$-equivalent, where $\overrightarrow{w_i} = vars(wtn_f(\phi_i)) \setminus vars(\phi)$, $\overrightarrow{w_{i+1}} = vars(wtn_f(\phi_{i+1})) \setminus vars(\phi)$. Also since $\phi_0 = \phi$, we have that ϕ and $\exists \overrightarrow{w} \,.\phi_k$ are $\mathcal{T}_\Sigma$-equivalent, where $\overrightarrow{w} = vars(\phi_k) \setminus vars(\phi)$, for the minimal k such that $\phi_k = \phi_{k+1}$. □

We now prove the following lemma:

Lemma 10 (Finite Witnessability) *Let ϕ be a conjunction of flat literals. Then, $wtn_f(\phi)$ is finitely witnessed for $\mathcal{T}_\Sigma$ with respect to* **Elem**.

Proof Suppose $\Gamma = wtn_f(\phi)$ is $\mathcal{T}_\Sigma$-satisfiable, and let $\mathcal{A}$ be a satisfying $\mathcal{T}_\Sigma$-interpretation. We define a $\mathcal{T}_\Sigma$-interpretation $\mathcal{B}$ which is a finite witness of Γ for $\mathcal{T}_\Sigma$ w.r.t. **Elem**. We set $\sigma^{\mathcal{B}} = vars_\sigma(\Gamma)^{\mathcal{A}}$ for every $\sigma \in \mathbf{Elem}$, $e^{\mathcal{B}} = e^{\mathcal{A}}$, for every variable $e \in vars_{\mathbf{Elem}}(\Gamma)$ and $x^{\mathcal{B}} = x^{\mathcal{A}}$ for every variable $x \in vars_{\mathbf{Struct}}(\Gamma)$. Selectors are also interpreted as they are interpreted in $\mathcal{A}$. This is well defined: for any **Struct**-variable x, every element in $\sigma^{\mathcal{A}}$ for $\sigma \in \mathbf{Elem}$ that occurs in $x^{\mathcal{A}}$ has a corresponding variable e in Γ such that $e^{\mathcal{A}}$ is that element. This holds by the finiteness of the sorts in **Struct** and the definition of wtn_f. Further, for any **Struct**-variable x such that $s_{c,i}(x)$ occurs in Γ, we must have that it occurs in some literal of the form $y = s_{c,i}(x)$ of Γ. Similarly to the above, all elements that occur in $y^{\mathcal{A}}$ and $x^{\mathcal{A}}$ have corresponding variables in Γ. Therefore, $\mathcal{B} \models \Gamma$ is a trivial consequence of $\mathcal{A} \models \Gamma$. By the definition of its domains, $\mathcal{B}$ is a finite witness of Γ for $\mathcal{T}_\Sigma$ w.r.t. **Elem**. □

By Lemmas 1, 8, 9 and 10, strong politeness is obtained.

Theorem 3 *If Σ is a datatypes signature and all sorts in* $\mathbf{Struct}_\Sigma$ *are finite, then $\mathcal{T}_\Sigma$ is strongly polite w.r.t.* $\mathbf{Elem}_\Sigma$.

4.3 Combining Finite and Inductive Datatypes

Now we consider the general case. Let Σ be an arbitrary datatypes signature. We prove that $\mathcal{T}_\Sigma$ is strongly polite w.r.t. **Elem**. We show that there are datatypes signatures $\Sigma_1, \Sigma_2 \subseteq \Sigma$ such that $\mathcal{T}_\Sigma = \mathcal{T}_{\Sigma_1} \oplus \mathcal{T}_{\Sigma_2}$, and then use Theorem 1. In Σ_1, inductive sorts are excluded, while in Σ_2, finite sorts are considered to be element sorts:

Theorem 4 *If Σ is a datatypes signature then $\mathcal{T}_\Sigma$ is strongly polite w.r.t.* $\mathbf{Elem}_\Sigma$.

Proof Set Σ_1 as follows: $\mathbf{Elem}_{\Sigma_1} = \mathbf{Elem}_\Sigma$ and $\mathbf{Struct}_{\Sigma_1} = Fin(\Sigma)$. $\mathcal{F}_{\Sigma_1} = \mathcal{CO}_{\Sigma_1} \uplus \mathcal{SE}_{\Sigma_1}$, where $\mathcal{CO}_{\Sigma_1} = \{c : \sigma_1 \times \cdots \times \sigma_n \to \sigma \mid c \in \mathcal{CO}_\Sigma, \sigma \in \mathbf{Struct}_{\Sigma_1}\}$ and $\mathcal{SE}_{\Sigma_1}$ and $\mathcal{P}_{\Sigma_1}$ are the corresponding selectors and testers. Notice that if σ is finite and $c : \sigma_1 \times \cdots \times \sigma_n \to \sigma$ is in $\mathcal{CO}_\Sigma$, then σ_i must be finite or in $\mathbf{Elem}_\Sigma$ for every $1 \le i \le n$. Next, we set Σ_2 as follows: $\mathcal{S}_{\Sigma_2} = \mathbf{Elem}_{\Sigma_2} \uplus \mathbf{Struct}_{\Sigma_2}$, where $\mathbf{Elem}_{\Sigma_2} = \mathbf{Elem}_\Sigma \cup Fin(\Sigma)$ and $\mathbf{Struct}_{\Sigma_2} = Ind(\Sigma)$. $\mathcal{F}_{\Sigma_2} = \mathcal{CO}_{\Sigma_2} \uplus \mathcal{SE}_{\Sigma_2}$, where $\mathcal{CO}_{\Sigma_2} = \{c : \sigma_2 \times \cdots \times \sigma_n \to \sigma \mid c \in \mathcal{CO}_\Sigma, \sigma \in \mathbf{Struct}_{\Sigma_2}\}$ and $\mathcal{SE}_{\Sigma_2}$ and $\mathcal{P}_{\Sigma_2}$ are the corresponding selectors and testers. Thus, $\mathcal{T}_\Sigma = \mathcal{T}_{\Sigma_1} \oplus \mathcal{T}_{\Sigma_2}$.

Now set $S = \mathbf{Elem}_\Sigma \cup Fin(\Sigma)$, $S_1 = \mathbf{Elem}_\Sigma$, $S_2 = \mathbf{Elem}_\Sigma \cup Fin(\Sigma)$, $T_1 = \mathcal{T}_{\Sigma_1}$, and $T_2 = \mathcal{T}_{\Sigma_2}$. By Theorem 3, T_1 is strongly polite w.r.t. S_1 and by Theorem 2, T_2 is strongly polite w.r.t. S_2. By Theorem 1, $\mathcal{T}_\Sigma$ is strongly polite w.r.t. $\mathbf{Elem}_\Sigma$. □

Remark 1 A concrete witness for $\mathcal{T}_\Sigma$ in the general case that we call wtn_Σ is obtained by first applying the witness from Definition 12 and then applying the witness from Definition 13 on the literals that involve finite sorts. A direct finite witnessability proof can be obtained by using the same arguments from the proofs of Lemmas 4 and 10. This witness is simpler than the one produced in the proof from [16] of Theorem 1 that involves purification and arrangements. In our case, we do not consider arrangements, but instead notice that the resulting function is additive, and hence ensures strong finite witnessability.

5 Axiomatizations

In this section, we discuss the possible connections between $\mathcal{T}_\Sigma$ and some axiomatizations of trees. We show how to get a reduction of any $\mathcal{T}_\Sigma$-satisfiability problem into a satisfiability

$$
\begin{array}{ll}
(\mathit{Inj}) & \{c(X_1,\ldots,X_n) = c(Y_1,\ldots,Y_n) \rightarrow \bigwedge_{i=1}^n X_i = Y_i \mid c \in \mathcal{CO}\} \\
(\mathit{Dis}) & \{c(X_1,\ldots,X_n) \neq d(Y_1,\ldots,Y_m) \mid c,d \in \mathcal{CO}, c \neq d\} \\
(\mathit{Proj}) & \{s_{c,i}(c(X_1,\ldots,X_n)) = X_i \mid c \in \mathcal{CO}, i \in [1,n]\} \\
(\mathit{Is}_1) & \{is_c(c(X_1,\ldots,X_n)) \mid c \in \mathcal{CO}\} \\
(\mathit{Is}_2) & \{\neg is_c(d(X_1,\ldots,X_n)) \mid c,d \in \mathcal{CO}, c \neq d\} \\
(\mathit{Acyc}) & \{X \neq t[X] \mid t \text{ is a non-variable } \Sigma_{|\mathcal{CO}}\text{-term that contains } X\ \} \\
(\mathit{Ext}_1) & \{\bigvee_{c:\sigma_1\times\ldots\times\sigma_n\rightarrow\sigma\in\mathcal{CO}} is_c(X) \mid \sigma \in \mathbf{Struct}\} \\
(\mathit{Ext}_2) & \{\exists \overrightarrow{y}\,.\, is_c(X) \rightarrow X = c(\overrightarrow{y}) \mid c \in \mathcal{CO}\}
\end{array}
$$

Fig. 1 Axioms for TREE_Σ and TREE^*_Σ

problem modulo an axiomatized theory of trees. The latter can be decided using syntactic unification.

Let Σ be a datatypes signature. The set TREE^*_Σ of axioms is defined as the union of all the sets of axioms in Fig. 1 (where upper case letters denote implicitly universally quantified variables). Let TREE_Σ be the set obtained from TREE^*_Σ by dismissing Ext_1 and Ext_2. Note that because of *Acyc*, we have that TREE_Σ is infinite (that is, consists of infinitely many axioms) unless all sorts in **Struct** are finite. TREE_Σ is a generalization of the theory of Absolutely Free Data Structures (AFDS) from [11] to many-sorted signatures with selectors and testers. In what follows we identify TREE_Σ (and TREE^*_Σ) with the class of structures that satisfy them when there is no ambiguity. It is routine to verify that the axioms are sound, and hence we have:

Proposition 3 *Every TREE^*_Σ-unsatisfiable formula is $\mathcal{T}_\Sigma$-unsatisfiable.*

5.1 A Satisfiability Procedure for TREE_Σ

Using an approach à la Shostak [3, 11, 18, 20, 28, 33], it is possible to get a satisfiability procedure for the $\Sigma_{|\mathcal{CO}}$-reduct of TREE_Σ. Consider the $\Sigma_{|\mathcal{CO}}$-theory FT defined from TREE_Σ by dismissing *Proj*, Is_1 and Is_2. As shown below, FT is a Shostak theory for which there exists a solver computing solved forms. A conjunction of equalities Γ of the form $\bigwedge_{k\in K} x_k = t_k$ is said to be a *solved form* if for each $k \in K$, x_k is a variable occurring only once in Γ. A theory T whose signature does not contain any predicate symbol is said to be a *Shostak* theory if

- T is convex, meaning that for any conjunction of literals φ over the signature of T, $T \cup \{\varphi\}$ does not entail any disjunction of equalities without entailing one of the equalities itself.
- T admits a solver $solve_T$ and a canonizer $canon_T$:
  - $solve_T$ computes, for any conjunction of equalities Φ, a formula Γ such that Γ is a solved form T-equivalent to Φ if Φ is T-satisfiable; otherwise Γ is the unsatisfiable formula $\perp$.
  - $canon_T$ is a computable idempotent mapping from terms to terms such that $T \models s = t$ iff $canon_T(s) = canon_T(t)$.

A substitution can be associated with any solved form. Formally, a substitution is defined in the usual way as an endomorphism of the structure of terms with only finitely many variables not mapped to themselves. In the case of a solved form $\Gamma = (\bigwedge_{k\in K} x_k = t_k)$, the associated substitution is $\mu = \{x_k \mapsto t_k\}_{k\in K}$. Application of the substitution μ to a term t is the term written $\mu(t)$ which is obtained from t by replacing x_k with t_k for each $k \in K$. The substitution associated with a solved form is useful to express a T-satisfiability procedure for a Shostak theory T.

Lemma 11 ([33]) *Let T be a Shostak theory. Assume Φ is any conjunction of equalities and Δ is any conjunction of disequalities such that $\Phi \wedge \Delta$ is built over the signature of T. $\Phi \wedge \Delta$ is T-satisfiable iff*

- *$solve_T$ computes, for the input Φ, a solved form $\Gamma = (\bigwedge_{k \in K} x_k = t_k)$,*
- *and for the substitution $\mu = \{x_k \mapsto t_k\}_{k \in K}$ and any $v \neq w$ in Δ, we have $canon_T(\mu(v)) \neq canon_T(\mu(w))$.*

A solver for FT is given by a syntactic unification algorithm [2], which can be viewed as a satisfiability procedure for the $\Sigma_{|\mathcal{CO}}$-structure of $\Sigma_{|\mathcal{CO}}$-trees over a countable $\mathcal{S}_\Sigma$-sorted set of variables V (cf. Definition 1), also denoted by $T(\Sigma_{|\mathcal{CO}}, V)$. Given any conjunction of $\Sigma_{|\mathcal{CO}}$-equalities Φ, a syntactic unification algorithm computes a formula Γ such that $T(\Sigma_{|\mathcal{CO}}, V) \models \Phi \Leftrightarrow \Gamma$ and Γ is either the unsatisfiable formula $\bot$ or a solved form.

Lemma 12 *FT is a Shostak theory where the solver is provided by a syntactic unification algorithm and the canonizer is the identity mapping.*

Proof Theories defined by Horn clauses are known to be convex [30]. Consequently, FT is convex.

Consider any conjunction of $\Sigma_{|\mathcal{CO}}$-equalities Φ. A syntactic unification algorithm with Φ as input computes a formula Γ such that $T(\Sigma_{|\mathcal{CO}}, V) \models \Phi \Leftrightarrow \Gamma$. The formula Γ can be obtained by a sequence of inferences, where each of these inferences corresponds to an equivalence that holds both in $T(\Sigma_{|\mathcal{CO}}, V)$ and in FT. Hence, $FT \models \Phi \Leftrightarrow \Gamma$. If Γ is a solved form, then both Γ and Φ are FT-satisfiable; otherwise, Γ is the unsatisfiable formula $\bot$ and both Γ and Φ are FT-unsatisfiable.

Let us now show that $FT \models s = t$ iff $s = t$. The "if" direction is obvious. For the "only-if" direction, we use that $T(\Sigma_{|\mathcal{CO}}, V) \models FT$. Thus, $FT \models s = t$ implies $T(\Sigma_{|\mathcal{CO}}, V) \models s = t$. Then, it suffices to remark that $T(\Sigma_{|\mathcal{CO}}, V) \models s = t$ iff $s = t$. □

TREE_Σ is not a Shostak theory because Σ includes some predicate symbols. However, TREE_Σ and FT coincide on $\Sigma_{|\mathcal{CO}}$-sentences.

Lemma 13 *For any $\Sigma_{|\mathcal{CO}}$-sentence φ, $\mathit{TREE}_\Sigma \models \varphi$ iff $FT \models \varphi$.*

Proof The "if" direction is a consequence of the fact that $\mathit{TREE}_\Sigma \models FT$. For the "only-if" direction, it is easy to show that any model of FT falsifying φ can be extended to a model of TREE_Σ falsifying φ. □

As a direct application of Lemmas 11, 12 and 13, it is possible to decide the TREE_Σ-satisfiability of any conjunction of $\Sigma_{|\mathcal{CO}}$-literals:

Lemma 14 *Assume Φ is any conjunction of $\Sigma_{|\mathcal{CO}}$-equalities and Δ is any conjunction of $\Sigma_{|\mathcal{CO}}$-disequalities. If a syntactic unification algorithm computes, for the input Φ, the unsatisfiable formula $\bot$, then $\Phi \wedge \Delta$ is TREE_Σ-unsatisfiable. Otherwise, it computes a solved form $\Gamma = (\bigwedge_{k \in K} x_k = t_k)$ and we have that:*

1. *$\Gamma \wedge \Delta$ is TREE_Σ-equivalent to $\Phi \wedge \Delta$,*
2. *$\Gamma \wedge \Delta$ is TREE_Σ-satisfiable iff for the substitution $\mu = \{x_k \mapsto t_k\}_{k \in K}$ and any $v \neq w$ in Δ, we have $\mu(v) \neq \mu(w)$.*

Remark 2 Along the lines of [1], a superposition calculus can be also applied to get a TREE_Σ-satisfiability procedure. Such a calculus has been used in [8, 11] for a theory of trees with selectors but no testers. To handle testers, one can use a classical encoding of predicates into first-order logic with equality, by representing an atom $is_c(x)$ as a flat equality $Is_c(x) = \mathbb{T}$ where Is_c is now a unary function symbol and $\mathbb{T}$ is a constant. Then, a superposition calculus dedicated to $TREE_\Sigma$ can be obtained by extending the standard superposition calculus [1] with some expansion rules, one for each axiom of $TREE_\Sigma$ [11]. For the axioms Is_1 and Is_2, the corresponding expansion rules are, respectively, $x = c(x_1, \ldots, x_n) \vdash Is_c(x) = \mathbb{T}$ if$c \in \mathcal{CO}$, and $x = d(x_1, \ldots, x_n) \vdash Is_c(x) \neq \mathbb{T}$ if$c, d \in \mathcal{CO}, c \neq d$.

We do not detail further the above superposition-based satisfiability procedure. Actually, Lemma 14 is sufficient to get a $\mathcal{T}_\Sigma$-satisfiability procedure based on a reduction to $TREE_\Sigma$-satisfiability of conjunctions of $\Sigma_{|\mathcal{CO}}$-literals.

5.2 A Satisfiability Procedure for $\mathcal{T}_\Sigma$

In the following, we show that any $\mathcal{T}_\Sigma$-satisfiability problem can be reduced to a $TREE_\Sigma$-satisfiability problem. Using a $TREE_\Sigma$-satisfiability procedure, this leads to a $\mathcal{T}_\Sigma$-satisfiability procedure.

Lemma 15 *Let Σ be a datatypes signature and φ any conjunction of flat Σ-literals including an arrangement over the variables in φ. Then, there exists a Σ-formula φ' such that:*

1. *φ and $\exists \overrightarrow{w} \,.\, \varphi'$ are $\mathcal{T}_\Sigma$-equivalent, where $\overrightarrow{w} = vars(\varphi') \backslash vars(\varphi)$.*
2. *φ' is $\mathcal{T}_\Sigma$-satisfiable iff φ' is $TREE_\Sigma$-satisfiable.*

Proof The proof is divided in two parts. First, we show how to construct the formula φ'. Second, we prove the equivalence between $\mathcal{T}_\Sigma$-satisfiability and $TREE_\Sigma$-satisfiability for φ'. To prove this equivalence, we construct a $\mathcal{T}_\Sigma$-interpretation satisfying φ' when φ' is $TREE_\Sigma$-satisfiable.

1. Construction of a $\mathcal{T}_\Sigma$-equivalent formula.

To define φ', let us first introduce the notion of is-constraint. Given a finite set of **Struct**-sorted variables V, an is-constraint over V is a conjunction ρ of literals $is_c(x)$ such that $x \in V$, $c : \sigma_1 \times \cdots \times \sigma_n \to \sigma \in \mathcal{CO}$ if x is of sort σ; and for every $x \in V$, there exists a unique c for which $is_c(x)$ occurs in ρ. The set of is-constraints over V is denoted by $IS(V)$. Given an is-constraint ρ, ρ_{eq} denotes a conjunction of equalities $x = c(y_1, \ldots, y_n)$ such that $is_c(x)$ occurs in ρ; all the variables $y_1, \ldots, y_n$ are distinct and fresh; and for every $is_c(x)$ in ρ, there exists a unique equality of the form $x = c(\ldots)$ in ρ_{eq}.

Assume φ is any conjunction of flat Σ-literals including an arrangement over the variables in φ. Consider the set of variables $GV(\varphi)$ defined as

$$\{x \mid is_c(x) \in \varphi\} \cup \{x \mid \neg is_c(x) \in \varphi\} \cup \{y \mid x = s_{c,i}(y) \in \varphi, s_{c,i} \in \mathcal{SE}\}$$

excluding all the variables in

$$\{y \mid x = s_{c,i}(y), y = d(\ldots) \in \varphi, s_{c,i} \in \mathcal{SE}, d \in \mathcal{CO}, d \neq c\}.$$

We want to build a formula equivalent to φ but including at least one σ-sorted variable for each $\sigma \in$ **Elem**. For this reason, let us denote φ_{te} a conjunction of trivial equalities $x_\sigma = x_\sigma$, one for every $\sigma \in$ **Elem** such that $vars_\sigma(\varphi) = \emptyset$, x_σ being a fresh σ-sorted variable. If $GV(\varphi) = \emptyset$, define $\varphi_1 = \varphi \wedge \varphi_{te}$. Otherwise, define φ_1 as follows:

$$\varphi_1 = \bigvee_{\rho \in IS(GV(\varphi))} w(\varphi, \rho_{eq}) \wedge \rho_{eq} \wedge \varphi_{te}$$

where $w(\varphi, \rho_{eq})$ is built in a inductive way by first considering the case of any Σ-literal l:

1. if $l = is_c(x)$ and $x = c(y_1, \ldots, y_n)$ occurs in ρ_{eq}, then $w(l, \rho_{eq}) = \top$;
2. if $l = \neg is_c(x)$ and $x = c(y_1, \ldots, y_n)$ occurs in ρ_{eq}, then $w(l, \rho_{eq}) = \bot$;
3. if $l = is_d(x)$, $x = c(y_1, \ldots, y_n)$ occurs in ρ_{eq} and $c \neq d$, then $w(l, \rho_{eq}) = \bot$;
4. if $l = \neg is_d(x)$, $x = c(y_1, \ldots, y_n)$ occurs in ρ_{eq} and $c \neq d$, then $w(l, \rho_{eq}) = \top$;
5. if $l = (y = s_{c,i}(x))$ and $x = c(y_1, \ldots, y_n)$ occurs in ρ_{eq}, then $w(l, \rho_{eq}) = (y = y_i)$;
6. otherwise, $w(l, \rho_{eq}) = l$.

If φ is any non-empty conjunction of Σ-literals of the form $l \wedge \varphi_r$ where l is a Σ-literal, then $w(\varphi, \rho_{eq})$ is obtained from $w(l, \rho_{eq}) \wedge w(\varphi_r, \rho_{eq})$ by simplifying the latter thanks to the properties that $\bot$ is absorbing for $\wedge$ and $\top$ is the identity for $\wedge$. Otherwise, φ is the empty conjunction $\top$, and $w(\varphi, \rho_{eq}) = \top$.

Note that the above construction is similar to the one given in [11] (see Proposition 4 in [11]). One can observe that $\varphi \wedge \rho_{eq}$ and $w(\varphi, \rho_{eq}) \wedge \rho_{eq}$ are $TREE^*_\Sigma$-equivalent. In particular, for the case (5.) above, it follows from the projection axiom *Proj* in $TREE^*_\Sigma$. In addition the guessing of is-constraint preserves the $TREE^*_\Sigma$-equivalence since $TREE^*_\Sigma$ includes the extensionality axioms Ext_1 and Ext_2. Thus φ and $\exists \overrightarrow{w}.\varphi_1$ are $TREE^*_\Sigma$-equivalent for $\overrightarrow{w} = vars(\varphi_1) \backslash vars(\varphi)$.

For any conjunction of literals ϕ, let us define

$$Min(\phi) = vars(\phi) \backslash \{x \mid x = c(\ldots) \text{ occurs in } \phi\}.$$

Starting from φ_1, consider the following sequences of formulas, obtained by guessing is-constraints for "minimal" variables of finite sorts:

$$\varphi_{j+1} = \bigvee_{\rho \in IS(\bigcup_{\sigma \in Fin(\Sigma)} Min_\sigma(\varphi_j))} \varphi_j \wedge \rho_{eq}$$

By definition of $Fin(\Sigma)$, there exists necessarily some j' such that the set of variables $\bigcup_{\sigma \in Fin(\Sigma)} Min_\sigma(\varphi_{j'})$ is empty. In that case, let us define $\varphi' = \varphi_{j'}$.

It is routine to show that φ and $\exists \overrightarrow{w}.\varphi'$ are $\mathcal{T}_\Sigma$-equivalent for the set of fresh variables $\overrightarrow{w} = vars(\varphi') \backslash vars(\varphi)$, using the following facts:

- all the sentences in $TREE^*_\Sigma$ are true in all the $\mathcal{T}_\Sigma$-interpretations,
- as shown above, φ and $\exists \overrightarrow{w}.\varphi_1$ are $TREE^*_\Sigma$-equivalent for the set of fresh variables $\overrightarrow{w} = vars(\varphi_1) \backslash vars(\varphi)$,
- φ_j and $\exists \overrightarrow{w}.\varphi_{j+1}$ are $TREE^*_\Sigma$-equivalent, for $\overrightarrow{w} = vars(\varphi_{j+1}) \backslash vars(\varphi_j)$ and any $j = 1, \ldots, j' - 1$, since $TREE^*_\Sigma$ includes the extensionality axioms Ext_1 and Ext_2.

2. Construction of a $\mathcal{T}_\Sigma$-interpretation.

Let us now show that φ' is $\mathcal{T}_\Sigma$-satisfiable iff φ' is $TREE_\Sigma$-satisfiable.

($\Rightarrow$) directly follows from Proposition 3.

($\Leftarrow$) If φ' is $TREE_\Sigma$-satisfiable, there exists a $TREE_\Sigma$-interpretation $\mathcal{A}$ and a disjunct ψ of φ' such that $\mathcal{A} \models \psi$. By construction of φ', ψ is a conjunction $\psi_{\mathcal{CO}} \wedge \psi_{\mathcal{SE}}$ where

- $\psi_{\mathcal{CO}}$ is a conjunction of $\Sigma_{|\mathcal{CO}}$-literals,
- $\psi_{\mathcal{SE}}$ is a conjunction of equalities of the form $x = s_{c,i}(y)$.

Since ψ holds in a $TREE_\Sigma$-interpretation, the conjunction of $\Sigma_{|\mathcal{CO}}$-equalities in $\psi_{\mathcal{CO}}$ has a most general unifier. By Lemma 14, $\psi_{\mathcal{CO}}$ is $TREE_\Sigma$-equivalent to a conjunction of literals $\Gamma \wedge \Delta$ such that

- Γ is a conjunction of equalities $\bigwedge_{k \in K} x_k = t_k$ such that for each $k \in K$, x_k is a variable occurring only once in Γ,

- Δ is the conjunction of disequalities in ψ,
- given the substitution $\mu = \{x_k \mapsto t_k\}_{k \in K}$, for any $v \neq w$ in Δ, $\mu(v) \neq \mu(w)$.

Consider the set of variables $MV = \{x \in vars_{\mathbf{Struct}}(\varphi') \mid \mu(x) = x\}$. Since the sorts of variables in MV are all inductive, there exists a substitution α from MV to $T(\Sigma_{|\mathcal{CO}}, vars_{\mathbf{Elem}}(\varphi'))$ such that for any $x, y \in MV$, $\alpha(x)^{\mathcal{A}} = \alpha(y)^{\mathcal{A}}$ iff $x = y$. According to this substitution α, we have for any terms $t, u \in T(\Sigma_{|\mathcal{CO}}, MV \cup vars_{\mathbf{Elem}}(\varphi'))$, $\alpha(t)^{\mathcal{A}} = \alpha(u)^{\mathcal{A}}$ iff $t = u$. In particular, we have for any $k, k' \in K$,

$$\alpha(\mu(x_k))^{\mathcal{A}} = \alpha(\mu(x_{k'}))^{\mathcal{A}} \text{ iff } \mu(x_k) = \mu(x_{k'}).$$

It is always possible to choose α such that for any $x, y \in MV$, $x \neq y$, we have

$$|depth(\alpha(x)) - depth(\alpha(y))| > \max\{depth(t_k)\}_{k \in K}.$$

According to the assumption on α, it is impossible to have $\alpha(\mu(x_k))^{\mathcal{A}} = \alpha(\mu(x))^{\mathcal{A}}$ for some $k \in K$ and some $x \in MV$. Consequently, we have for any $x, y \in vars_{\mathbf{Struct}}(\varphi')$,

$$\alpha(\mu(x))^{\mathcal{A}} = \alpha(\mu(y))^{\mathcal{A}} \text{ iff } \mu(x) = \mu(y).$$

Let us now consider $\mathcal{B} \in \mathcal{T}_\Sigma$ such that

- for any $\sigma \in \mathbf{Elem}$, $\sigma^{\mathcal{B}} = \{e^{\mathcal{A}} \mid e \in vars_\sigma(\varphi')\}$,
- for any $x \in vars_{\mathbf{Struct}}(\varphi')$, $x^{\mathcal{B}} = \alpha(\mu(x))^{\mathcal{A}}$,
- for any $e \in vars_{\mathbf{Elem}}(\varphi')$, $e^{\mathcal{B}} = e^{\mathcal{A}}$.

One can observe that $\mathcal{B} \models \Gamma \wedge \Delta$ since

- for any $x_k = t_k$ in Γ, $\mu(x_k) = \mu(t_k)$ and so $x_k^{\mathcal{B}} = t_k^{\mathcal{B}}$,
- for any $v \neq w$ in Δ, $\mu(v) \neq \mu(w)$ and so $v^{\mathcal{B}} \neq w^{\mathcal{B}}$.

Since all the sentences in $TREE_\Sigma^*$ are true in all the $\mathcal{T}_\Sigma$-interpretations and $\Gamma \wedge \Delta$ is $TREE_\Sigma$-equivalent to $\psi_{\mathcal{CO}}$, we have $\mathcal{B} \models \psi_{\mathcal{CO}}$.

Let us now consider the conjunction $\psi_{\mathcal{SE}}$ that contains only equalities of the form $x = s_{c,i}(y)$. By construction of φ', the term $\mu(y)$ is necessarily rooted by a constructor $d \in \mathcal{CO}$, $d \neq c$. Thus $s_{c,i}^{\mathcal{B}}$ can be defined arbitrarily on $y^{\mathcal{B}}$ since $y^{\mathcal{B}}$ is a standard tree rooted by some constructor d different from c. In particular, we can define $s_{c,i}^{\mathcal{B}}$ such that $s_{c,i}^{\mathcal{B}}(y^{\mathcal{B}}) = x^{\mathcal{B}}$. Using this interpretation $\mathcal{B}$ for the selectors, we have $\mathcal{B} \models \psi_{\mathcal{SE}}$.

Since $\mathcal{B} \models \psi_{\mathcal{CO}}$ and $\mathcal{B} \models \psi_{\mathcal{SE}}$, we get $\mathcal{B} \models \psi$. Since ψ is some disjunct of φ', we can conclude that $\mathcal{B} \models \varphi'$. □

Lemma 15 can be easily lifted to any quantifier-free Σ-formula thanks to the following transformations:

- computation of a disjunctive normal form, that is, a disjunction of conjunctions of Σ-literals;
- flattening of each conjunction of Σ-literals;
- for each resulting conjunction of flat Σ-literals, guessing all the possible arrangements over its variables.

Therefore, Lemma 15 leads to:

Theorem 5 *Let Σ be a datatypes signature and φ any quantifier-free Σ-formula. Then, there exists a Σ-formula φ' such that:*

1. φ and $\exists \overrightarrow{w}\,.\,\varphi'$ are $\mathcal{T}_\Sigma$-equivalent, where $\overrightarrow{w} = vars(\varphi') \backslash vars(\varphi)$.

2. *φ' is $\mathcal{T}_\Sigma$-satisfiable iff φ' is $TREE_\Sigma$-satisfiable.*

In both Lemma 15 and Theorem 5, $\exists\overrightarrow{w}\ .\ \varphi'$ and φ are not only $\mathcal{T}_\Sigma$-equivalent but also $TREE^*_\Sigma$-equivalent. As a consequence, both Lemma 15 and Theorem 5 also hold when stated using $TREE^*_\Sigma$ instead of $\mathcal{T}_\Sigma$. This shows that any quantifier-free Σ-formula is $\mathcal{T}_\Sigma$-satisfiable iff it is $TREE^*_\Sigma$-satisfiable.

5.3 Politeness and Axiomatization

We conclude this section with a short discussion on the connection to Sect. 4. Both the current section and Sect. 4 rely on two constructions: (i) A formula transformation (wtn_Σ in Remark 1 of Sect. 4, $\varphi \mapsto \varphi'$ in Lemma 15 of the current section); and (ii) A small model construction (finite witnessability in Sect. 4, equisatisfiability between $\mathcal{T}_\Sigma$ and $TREE$ in Lemma 15). While these constructions are similar in both sections, they are not the same. A nice feature of the constructions of Sect. 4 is that they clearly separate between steps (i) and (ii). The witness is very simple, and amounts to adding to the input formula literals and disjunctions that trivially follow from the original formula in $\mathcal{T}_\Sigma$. Then, the resulting formula is post-processed in step (ii), according to a given satisfying interpretation. Having a satisfying interpretation allows us to greatly simplify the formula, and the simplified formula is useful for the model construction. In contrast, the satisfying $TREE_\Sigma$-interpretation that we start with in step (ii) of the current section is not necessarily a $\mathcal{T}_\Sigma$-interpretation, which makes the approach of Sect. 4 incompatible, compared to the syntactic unification approach that we employ here. For that, some of the post-processing steps of Sect. 4 are employed in step (i) itself, in order to eliminate all testers and as much selectors as possible. In addition, a pre-processing is applied in order to include an arrangement. The constructed interpretation finitely witnesses φ' and so this technique can be used to produce an alternative proof of strong politeness.

6 Conclusion

In this paper we have studied the theory of algebraic datatypes, as it is defined by the SMT-LIB 2 standard. Our investigation included both finite and inductive datatypes. For this theory, we have proved that it is strongly polite, making it amenable for combination with other theories by the polite combination method. Our proofs used the notion of additive witnesses, also introduced in this paper. We concluded by extending existing axiomatizations and a decision procedure for trees to support this theory of datatypes.

There are several directions for further research that we plan to explore. First, we plan to continue to prove that more important theories are strongly polite, with an eye to recent extensions of the datatypes theory, namely datatypes with shared selectors [25] and co-datatypes [24]. Second, we envision to further investigate the possibility to prove politeness using superposition-based satisfiability procedures. Third, we plan to study extensions of the theory of datatypes corresponding to finite trees including function symbols with some equational properties such as associativity and commutativity to model data structures such as multisets [29]. We want to focus on the politeness of such extensions. Initial work in that direction has been done in [7] that we plan to build on.

References

1. Armando, A., Bonacina, M.P., Ranise, S., Schulz, S.: New results on rewrite-based satisfiability procedures. ACM Trans. Comput. Log. **10**(1), 4:1-4:51 (2009)
2. Baader, F., Snyder, W., Narendran, P., Schmidt-Schauß, M., Schulz, K.U.: Unification theory. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning (in 2 Volumes), pp. 445–532. Elsevier/MIT Press, New York (2001)
3. Barrett, C.W., Dill, D.L., Stump, A.: A generalization of shostak's method for combining decision procedures. In: A. Armando (ed.) Frontiers of Combining Systems, 4th International Workshop, FroCoS 2002, Santa Margherita Ligure, Italy, April 8–10, 2002, Proceedings, Lecture Notes in Computer Science, vol. 2309, pp. 132–146. Springer (2002)
4. Barrett, C.W., Shikanian, I., Tinelli, C.: An abstract decision procedure for a theory of inductive data types. J. Satisfiab. Boolean Model. Comput. **3**(1–2), 21–46 (2007)
5. Barrett, C., Conway, C.L., Deters, M., Hadarean, L., Jovanović, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Proceedings of the 23rd International Conference on Computer Aided Verification, CAV'11, pp. 171–177. Springer (2011). http://dl.acm.org/citation.cfm?id=2032305.2032319
6. Barrett, C., Fontaine, P., Tinelli, C.: The SMT-LIB Standard: Version 2.6. Tech. rep., Department of Computer Science, The University of Iowa (2017). Available at www.SMT-LIB.org
7. Berthon, R., Ringeissen, C.: Satisfiability modulo free data structures combined with bridging functions. In: T. King, R. Piskac (eds.) Proceedings of SMT@IJCAR 2016, CEUR Workshop Proceedings, vol. 1617, pp. 71–80. CEUR-WS.org (2016)
8. Bonacina, M.P., Echenim, M.: Rewrite-based satisfiability procedures for recursive data structures. Electron. Notes Theor. Comput. Sci. **174**(8), 55–70 (2007)
9. Bonacina, M.P., Fontaine, P., Ringeissen, C., Tinelli, C.: Theory combination: Beyond equality sharing. In: C. Lutz, U. Sattler, C. Tinelli, A. Turhan, F. Wolter (eds.) Description Logic, Theory Combination, and All That - Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday, Lecture Notes in Computer Science, vol. 11560, pp. 57–89. Springer (2019)
10. Casal, F., Rasga, J.: Many-sorted equivalence of shiny and strongly polite theories. J. Autom. Reason. **60**(2), 221–236 (2018)
11. Chocron, P., Fontaine, P., Ringeissen, C.: Politeness and combination methods for theories with bridging functions. J. Autom. Reason. **64**(1), 97–134 (2020)
12. Enderton, H.B.: A Mathematical Introduction to Logic. Academic Press, New York (2001)
13. Fontaine, P.: Combinations of theories for decidable fragments of first-order logic. In: S. Ghilardi, R. Sebastiani (eds.) Frontiers of Combining Systems, 7th International Symposium, FroCoS 2009, Trento, Italy, September 16–18, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5749, pp. 263–278. Springer (2009)
14. Gutiérrez, R., Meseguer, J.: Variant-based decidable satisfiability in initial algebras with predicates. In: F. Fioravanti, J.P. Gallagher (eds.) Logic-Based Program Synthesis and Transformation—27th International Symposium, LOPSTR 2017, Namur, Belgium, October 10–12, 2017, Revised Selected Papers, Lecture Notes in Computer Science, vol. 10855, pp. 306–322. Springer (2017)
15. Hojjat, H., Rümmer, P.: Deciding and interpolating algebraic data types by reduction. In: T. Jebelean, V. Negru, D. Petcu, D. Zaharie, T. Ida, S.M. Watt (eds.) 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2017, Timisoara, Romania, September 21–24, 2017, pp. 145–152. IEEE Computer Society (2017)
16. Jovanovic, D., Barrett, C.W.: Polite theories revisited. In: C.G. Fermüller, A. Voronkov (eds.) Logic for Programming, Artificial Intelligence, and Reasoning—17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10–15, 2010. Proceedings, *Lecture Notes in Computer Science*, vol. 6397, pp. 402–416. Springer (2010). Extended technical report is available at http://theory.stanford.edu/~barrett/pubs/JB10-TR.pdf
17. Kovács, L., Robillard, S., Voronkov, A.: Coming to terms with quantified reasoning. In: G. Castagna, A.D. Gordon (eds.) Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18–20, 2017, pp. 260–270. ACM (2017)
18. Krstic, S., Conchon, S.: Canonization for disjoint unions of theories. Inf. Comput. **199**(1–2), 87–106 (2005)
19. Krstic, S., Goel, A., Grundy, J., Tinelli, C.: Combined satisfiability modulo parametric theories. In: O. Grumberg, M. Huth (eds.) Tools and Algorithms for the Construction and Analysis of Systems, 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24–April 1, 2007, Proceedings, Lecture Notes in Computer Science, vol. 4424, pp. 602–617. Springer (2007)

20. Manna, Z., Zarba, C.G.: Combining decision procedures. In: B.K. Aichernig, T.S.E. Maibaum (eds.) Formal Methods at the Crossroads. From Panacea to Foundational Support, 10th Anniversary Colloquium of UNU/IIST, the International Institute for Software Technology of The United Nations University, Lisbon, Portugal, March 18–20, 2002, Revised Papers, Lecture Notes in Computer Science, vol. 2757, pp. 381–422. Springer (2002)
21. Meseguer, J.: Variant-based satisfiability in initial algebras. Sci. Comput. Program. **154**, 3–41 (2018)
22. Nelson, G., Oppen, D.C.: Simplification by cooperating decision procedures. ACM Trans. Program. Lang. Syst. **1**(2), 245–257 (1979)
23. Ranise, S., Ringeissen, C., Zarba, C.G.: Combining data structures with nonstably infinite theories using many-sorted logic. In: B. Gramlich (ed.) Frontiers of Combining Systems, 5th International Workshop, FroCoS 2005, Vienna, Austria, September 19–21, 2005, Proceedings, Lecture Notes in Computer Science, vol. 3717, pp. 48–64. Springer (2005). Extended technical report is available at https://hal.inria.fr/inria-00070335/
24. Reynolds, A., Blanchette, J.C.: A decision procedure for (co)datatypes in SMT solvers. J. Autom. Reason. **58**(3), 341–362 (2017)
25. Reynolds, A., Viswanathan, A., Barbosa, H., Tinelli, C., Barrett, C.W.: Datatypes with shared selectors. In: D. Galmiche, S. Schulz, R. Sebastiani (eds.) Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14–17, 2018, Proceedings, Lecture Notes in Computer Science, vol. 10900, pp. 591–608. Springer (2018)
26. Sheng, Y., Zohar, Y., Ringeissen, C., Lange, J., Fontaine, P., Barrett, C.W.: Politeness for the theory of algebraic datatypes. In: IJCAR (1), Lecture Notes in Computer Science, vol. 12166, pp. 238–255. Springer (2020)
27. Sheng, Y., Zohar, Y., Ringeissen, C., Reynolds, A., Barrett, C.W., Tinelli, C.: Politeness and stable infiniteness: Stronger together. In: CADE, Lecture Notes in Computer Science, vol. 12699, pp. 148–165. Springer (2021)
28. Shostak, R.E.: A practical decision procedure for arithmetic with function symbols. J. ACM **26**(2), 351–360 (1979)
29. Sofronie-Stokkermans, V.: Locality results for certain extensions of theories with bridging functions. In: R.A. Schmidt (ed.) Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2–7, 2009. Proceedings, Lecture Notes in Computer Science, vol. 5663, pp. 67–83. Springer (2009)
30. Tinelli, C.: Cooperation of background reasoners in theory reasoning by residue sharing. J. Autom. Reason. **30**(1), 1–31 (2003)
31. Tinelli, C., Zarba, C.G.: Combining decision procedures for sorted theories. In: J.J. Alferes, J.A. Leite (eds.) Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Lisbon, Portugal, September 27–30, 2004, Proceedings, Lecture Notes in Computer Science, vol. 3229, pp. 641–653. Springer (2004)
32. Tinelli, C., Zarba, C.G.: Combining nonstably infinite theories. J. Autom. Reason. **34**(3), 209–238 (2005)
33. Tran, D., Ringeissen, C., Ranise, S., Kirchner, H.: Combination of convex theories: modularity, deduction completeness, and explanation. J. Symb. Comput. **45**(2), 261–286 (2010)