Politeness for the Theory of Algebraic Datatypes

Ying Sheng[1], Yoni Zohar[1(✉)], Christophe Ringeissen[2], Jane Lange[1], Pascal Fontaine[2,3], and Clark Barrett[1]

[1] Stanford University, Stanford, USA
yoni206@gmail.com
[2] Université de Lorraine, CNRS, Inria, LORIA, 54000 Nancy, France
[3] Université de Liège, Liège, Belgium

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 also how it can be combined with other arbitrary disjoint theories using polite combination. Our results cover both inductive and finite datatypes, as well as their union. The combination method uses a new, simple, and natural notion of additivity, that enables deducing strong politeness from (weak) politeness.

1 Introduction

Algebraic datatypes such as 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 combination methods, and focuses on parametric datatypes (or *generic* datatypes as they are called in the programming languages community).

The traditional combination method of Nelson and Oppen [20] 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

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).

N. Peltier and V. Sofronie-Stokkermans (Eds.): IJCAR 2020, LNAI 12166, pp. 238–255, 2020.
https://doi.org/10.1007/978-3-030-51074-9_14

fixed-width bit-vectors, which is commonly used for modeling and verifying both hardware and software. To be able to perform combinations with such theories, a more general combination method was designed [21], 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* [14], which also account for possible arrangements of the variables in the formula. 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 [21]), strong politeness of algebraic datatypes remained an unanswered question.

The main contribution of this paper is an affirmative answer to this question. We introduce a *witness* function that essentially "guesses" the right constructors of variables without 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, which allows us to prove politeness and conclude strong politeness. We further study the theory of datatypes beyond politeness and extend a decision procedure for a subset of this theory presented in [9] 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 [3]. 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 [23] and co-datatypes [22]. More recent approaches for solving formulas about datatypes use, e.g., theorem provers [15], variant satisfiability [12,19], and reduction-based decision procedures [1,6,13].

In this paper, we focus on polite theory combination. Other combination methods for non stably infinite theories include shiny theories [27], gentle theories [11], and parametric theories [17]. The politeness property was introduced in [21], 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 [14], and a corrected definition (here called *strong politeness*) was introduced. Strongly polite theories were further studied in [8], where the authors proved their equivalence with shiny theories.

More recently, it was proved [9] 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 [7]. Actually, it was unclear to the authors of [7] 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 and polite theories, and it introduces our working definition of the theory of datatypes, which is based on SMT-LIB 2. 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.

2 Preliminaries

2.1 Signatures and Structures

We briefly review usual definitions of many-sorted first-order logic with equality (see [10,26] for more details). For any set S, an *S-sorted set* A is a function from S to $\mathcal{P}(X) \setminus \{\emptyset\}$ for some set X (i.e., A assigns a non-empty set to every element of S), such that $A(s) \cap A(s') = \emptyset$ whenever $s \neq s'$. We use A_s to denote $A(s)$ for every $s \in S$, and call the elements of S *sorts*. When there is no ambiguity, we sometimes treat sorted sets as sets (e.g., when writing expressions like $x \in A$). Given a set S (of sorts), the *canonical S-sorted set*, denoted $[[S]]$, satisfies $[[S]]_s = \{s\}$ for every $s \in S$. 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 $\sigma_1 \times \ldots \times \sigma_n \to \sigma$, and predicate symbols have arities of the form $\sigma_1 \times \ldots \times \sigma_n$, with $\sigma_1, \ldots, \sigma_n, \sigma \in \mathcal{S}_\Sigma$. For each sort $\sigma \in \mathcal{S}_\Sigma$, $\mathcal{P}_\Sigma$ includes an *equality symbol* $=_\sigma$ of arity $\sigma \times \sigma$. 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 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 $x, y, x_1, \ldots, x_n$ and function and predicate symbols f and 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, and interpretations to the function and predicate symbols of Σ. We further require that $=_\sigma$ is interpreted as the identity relation over A_σ for every $\sigma \in \mathcal{S}_\Sigma$. A *$\Sigma$-interpretation* $\mathcal{A}$ is an extension

of a Σ-structure with interpretations to some set of variables. For any Σ-term α, $\alpha^{\mathcal{A}}$ denotes the interpretation of α in $\mathcal{A}$. When α is a set of Σ-terms, $\alpha^{\mathcal{A}} = \{x^{\mathcal{A}} \mid x \in \alpha\}$. 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 [3], but is adjusted to suit our investigation of politeness.

Definition 1. *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.*
- $T_{\sigma,i+1}(\Sigma, A) = T_{\sigma,i}(\Sigma, A) \cup \{c(t_1, \ldots, t_n) \mid c : \sigma_1 \times \ldots \times \sigma_n \to \sigma \in \mathcal{F}_\Sigma, t_j \in T_{\sigma_j,i}(\Sigma, A)$ *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-ary 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-ary 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 $A = \{a\}$. For the **elem** sort, $T_{\mathbf{elem}}(\Sigma, A)$ is the singleton $A = \{a\}$ and the Σ-tree a is of depth 0. For the **struct** sort, $T_{\mathbf{struct}}(\Sigma, A)$ 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. *A finite signature Σ is called a* datatypes signature *if $\mathcal{S}_\Sigma$ is the disjoint union of two sets of sorts $\mathcal{S}_\Sigma = \mathbf{Elem}_\Sigma \uplus \mathbf{Struct}_\Sigma$ and $\mathcal{F}_\Sigma$ is the disjoint union of two sets of function symbols $\mathcal{F}_\Sigma = \mathcal{CO}_\Sigma \uplus \mathcal{SE}_\Sigma$, such that $\mathcal{SE}_\Sigma = \{s_{c,i} : \sigma \to \sigma_i \mid c \in \mathcal{CO}_\Sigma, c : \sigma_1, \ldots, \sigma_n \to \sigma, 1 \leq i \leq n\}$ and $\mathcal{P}_\Sigma = \{is_c : \sigma \mid c \in \mathcal{CO}_\Sigma, c : \sigma_1, \ldots, \sigma_n \to \sigma\}$. We denote by $\Sigma_{|\mathcal{CO}}$ the signature with*

the same sorts as Σ, no predicate symbols (except $=_\sigma$ for $\sigma \in \mathcal{S}_\Sigma$), and whose function symbols are $\mathcal{CO}_\Sigma$. We further require the following well-foundedness requirement: $T_\sigma(\Sigma_{|\mathcal{CO}}, [[\mathbf{Elem}_\Sigma]]) \neq \emptyset$ 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 $[[\mathbf{Elem}_\Sigma]]$, $\mathcal{CO}$ rather than $\mathcal{CO}_\Sigma$) whenever Σ is clear from the context. Notice that Definition 2 remains equivalent if we replace $[[\mathbf{Elem}]]$ by any (non-empty) **Elem**-sorted set A. The set $[[\mathbf{Elem}]]$ has been chosen since this minimal **Elem**-sorted set is sufficient.

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*. 0-ary constructors 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 $(\mathbf{elem} \times \mathbf{list}) \rightarrow \mathbf{list}$, *nil* of arity **list**, *car* of arity $\mathbf{list} \rightarrow \mathbf{elem}$ and *cdr* of arity $\mathbf{list} \rightarrow \mathbf{list}$. 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 $(\mathbf{elem} \times \mathbf{elem}) \rightarrow \mathbf{pair}$ and $first$ and *second* of arity $\mathbf{pair} \rightarrow \mathbf{elem}$. 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**. Its function symbols are *cons* of arity $(\mathbf{pair} \times \mathbf{list}) \rightarrow \mathbf{list}$, *car* of arity $\mathbf{list} \rightarrow \mathbf{pair}$, as well as $nil, cdr, first, second$ with arities as above. Its predicate symbols are is_{pair}, is_{cons} and *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. *A sort $\sigma \in \mathbf{Struct}$ is* 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. *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* $(\sigma_1 \times \ldots \times \sigma_n) \to \sigma$ *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* $(\sigma_1 \times \ldots \times \sigma_n) \to \sigma$, $t_1 \in \sigma_1^{\mathcal{A}}, \ldots, t_n \in \sigma_n^{\mathcal{A}}$ *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* $(\sigma_1 \times \ldots \times \sigma_n) \to \sigma$.

$\mathcal{A}$ 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 $\mathbf{list}^{\mathcal{A}}$ is the set of terms constructed from $\mathbf{elem}^{\mathcal{A}}$ and *cons*, plus *nil*. If $\mathbf{elem}^{\mathcal{A}}$ is the set of natural numbers, then $\mathbf{list}^{\mathcal{A}}$ 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 $\mathbf{pair}^{\mathcal{A}}$ is the set of terms of the form $pair(a, b)$ with $a, b \in \mathbf{elem}^{\mathcal{A}}$. If $\mathbf{elem}^{\mathcal{A}}$ is again interpreted as the set of natural numbers, $\mathbf{pair}^{\mathcal{A}}$ 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, $\mathbf{pair}^{\mathcal{A}}$ 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 $T_1 \cup T_2$-satisfiability problems using the satisfiability procedures known for T_1 and T_2. Assuming that T_1 and T_2 have disjoint signatures is not sufficient to get a complete combination method for the satisfiability problem. The reason is that T_1 and T_2 may share sorts, and the equality symbol on these shared sorts. 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 built over the equality symbol. These two requirements can be easily 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 V be a finite set of variables whose sorts are in S and $\{V_\sigma \mid \sigma \in S\}$ a 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 both T_1 and T_2 are stably infinite with disjoint signatures, and let V be the finite set of variables shared by T_1 and T_2. Under this assumption, T_1 and T_2 can agree on an infinite cardinality, and guessing an arrangement of V 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 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 [14,21] in order to distinguish between politeness and strong politeness (in terms of [8]) 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 $\kappa(\sigma) \geq \left|\sigma^{\mathcal{A}}\right|$ for every $\sigma \in S$ there exists a Σ-interpretation $\mathcal{A}'$ that satisfies ϕ with $\left|\sigma^{\mathcal{A}'}\right| = \kappa(\sigma)$ 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. 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. ϕ is *strongly finitely witnessed for T w.r.t. S* if $\phi \wedge \delta_V$ is finitely witnessed for T w.r.t. S for every arrangement δ_V of V, where V is any set of variables whose sorts are in S. We say that a function $wtn : QF(\Sigma) \to QF(\Sigma)$ 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.[1]

[1] 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.

Definition 8 (Finitely Witnessable). *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.*

Finally, we recall the following theorem from [14].

Theorem 1 ([14]). *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 [14] that politeness is not sufficient for the proof of the polite combination method from [21]. Strong politeness was introduced to fix the problem. It is unknown, however, whether there are theories that are polite but not strongly polite. 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 [14] using notions of the current paper, shows that the 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 $=_\sigma$), T_0 the Σ_0-theory consisting of all Σ_0-structures $\mathcal{A}$ with $\left|\sigma^{\mathcal{A}}\right| \geq 2$, ϕ the formula $x = x \wedge w = w$, and δ the arrangement $(x = w)$ of $\{x, w\}$. Then $\phi \wedge \delta$ is T_0-satisfiable, but every interpretation $\mathcal{A}$ with $\sigma^{\mathcal{A}} = \{x, w\}^{\mathcal{A}}$ that satisfies it has only one element in $\sigma^{\mathcal{A}}$ and so ϕ is not strongly finitely witnessed for T_0 w.r.t. σ. It is straightforward to show, however, that ϕ is finitely witnessed for T_0 w.r.t. σ. Moreover, 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. This does not show, however, that T_0 is not strongly polite. In fact, it is indeed 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.

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 $f : QF(\Sigma) \to QF(\Sigma)$. 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 just 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.*

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.*[2]

Corollary 1. *Suppose T is additively polite w.r.t. S. Then it is strongly polite w.r.t. S.*

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

T_0 is an *existential theory*: it consists of all the structures that satisfy an existential sentence (in this case, $\exists x, y \,.\, x \neq y$). The construction of wtn'' 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 polite.

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 [21] 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. For smoothness, however, it is straightforward to show that the **Elem** domain of a given interpretation can always be augmented without changing satisfiability of quantifier-free formulas.

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

Lemma 1 holds for any datatypes signature.

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 [21], 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

[2] Due to lack of space, some proofs have been omitted. They can be found in an extended version at https://arxiv.org/abs/2004.04854.

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 11 (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 $x = c(\overrightarrow{u_1}, y, \overrightarrow{u_2})$ does not occur in ϕ and $x = d(\overrightarrow{u_d})$ does not occur in ϕ for any $\overrightarrow{u_1}, \overrightarrow{u_2}, \overrightarrow{u_d}$, we conjunctively add $x = c(\overrightarrow{u_1}, y, \overrightarrow{u_2}) \vee (\bigvee_{d \neq c} x = d(\overrightarrow{u_d}))$ with fresh $\overrightarrow{u_1}, \overrightarrow{u_2}, \overrightarrow{u_d}$, where c and d range over $\mathcal{CO}$.*
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 11, 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.*

Further, it can be verified that:

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 $\overrightarrow{w} = vars(\Gamma) \setminus vars(\phi)$.*

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**. First for every $\sigma \in$ **Elem** we set $\sigma^{\mathcal{B}} = vars_\sigma(\Gamma)^{\mathcal{A}}$, and for every variable $e \in vars_\sigma(\Gamma)$, we set $e^{\mathcal{B}} = e^{\mathcal{A}}$. The interpretations of **Struct**-sorts, testers and constructors are uniquely determined by the theory. It is left to define the interpretation of **Struct**-variables in $\mathcal{B}$, as well as the interpretation of the selectors (the interpretation of selectors is fixed by the theory only when applied to the "right" constructor). 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 11). Since $\mathcal{A} \models \Gamma$, $\mathcal{A}$ satisfies exactly one disjunct of each such disjunction. We can thus obtain a formula Γ_1 from Γ by replacing every disjunction with the 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'. 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 an 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 $\alpha^{\mathcal{A}} = \{x^{\mathcal{A}} \mid x \in \alpha\}$ 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, such that $\alpha \prec \beta$ 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 such that $w_k \in \alpha$ for some $y \in \beta$, $c \in \mathcal{CO}$, and k. Call an equivalence class α *nullary* if $\mathcal{A} \models is_c(x)$ for some $x \in \alpha$ and nullary constructor c. Call an equivalence class α *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 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 define $\alpha^{\mathcal{B}}$ for every equivalence class α. Then, $x^{\mathcal{B}}$ is simply defined as $[x]^{\mathcal{B}}$, for every **Struct**-variable x. The idea goes as follows. Nullary classes are assigned according to $\mathcal{A}$. 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 $\alpha_i^{\mathcal{B}} = \alpha_i^{\mathcal{A}}$.
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 define $\alpha_i^{\mathcal{B}}$ to be an arbitrary element of $T_\sigma(\Sigma_{|\mathcal{CO}}, \mathcal{B}_{\mathbf{Elem}})$ that has depth strictly greater than $\max\{depth(\alpha_j^{\mathcal{B}}) \mid 0 < j < i\} + d$ (here $\max \emptyset = 0$).

3. If $i > l$ then we set $\alpha_i^{\mathcal{B}} = c(\beta_1^{\mathcal{B}}, \ldots, \beta_n^{\mathcal{B}})$ 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 $x_1 \in \beta_1, \ldots, x_n \in \beta_n$.

Since Σ is a datatypes signature in which all **Struct**-sorts are inductive, the second case of the definition is well-defined. Further, the topological sort ensures $\beta_1, \ldots, \beta_n$ exist, and the partition to equivalence classes ensures that they are unique. Hence:

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

Step 5 – Interpretation of Selectors: Let $s_{c,i} \in \mathcal{SE}$ for $c : \sigma_1 \times \ldots \times \sigma_n \to \sigma$, $1 \leq i \leq n$ 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 $s_{c,i}^{\mathcal{B}}(a) = a_i$. Otherwise, we consider two cases. If $x^{\mathcal{B}} = a$ for some $x \in \mathit{vars}(\Gamma)$ such that $y = s_{c,i}(x)$ occurs in Γ_2 for some y, we set $s_{c,i}^{\mathcal{B}}(a) = y^{\mathcal{B}}$. Otherwise, $s_{c,i}^{\mathcal{B}}(a)$ is set arbitrarily.

Example 6. Let Γ be the following Σ_{list}-formula: $x_1 = \mathit{cons}(e_1, x_2) \wedge x_3 = \mathit{cons}(e_2, x_4) \wedge x_2 \neq x_4$. 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 $x_3^{\mathcal{B}} = [2, 1, 1, 1, 1, 1]$.

Now that $\mathcal{B}$ is defined, it is left to show that it is a finite witness of Γ for $\mathcal{T}_\Sigma$ w.r.t. **Elem**. By construction, $\sigma^{\mathcal{B}} = \mathit{vars}_\sigma(\Gamma)^{\mathcal{B}}$ for every $\sigma \in \mathbf{Elem}$. $\mathcal{B}$ also preserves the equalities and disequalities in $\mathcal{A}$, and by considering every shape of a literal in Γ' we can prove that $\mathcal{B} \models \Gamma'$. Our interpretation of the selectors then ensures that:

Lemma 6. *$\mathcal{B} \models \Gamma$.*

Lemma 6, together with the definition of the domains of $\mathcal{B}$, gives 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 corollary of Lemmas 1, 2 and 4, strong politeness is obtained.

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

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. 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 and variables $y_1, \ldots, y_n$, if such x exists. Since **Struct** only has finite sorts, this sequence becomes constant at some ϕ_k.

Definition 12 (A Witness for $\mathcal{T}_\Sigma$). *$wtn_f(\phi)$ is ϕ_k for the minimal k such that $\phi_k = \phi_{k+1}$.*

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

Similarly to Sect. 4.1, we have:

Lemma 7. *wtn_f is* **Elem***-additive.*

Lemma 8. *ϕ and $\exists \overrightarrow{w} \,.\, wtn_f(\phi)$ are $\mathcal{T}_\Sigma$-equivalent, where $\overrightarrow{w} = vars(wtn_f(\phi)) \setminus vars(\phi)$.*

We now prove the following lemma:

Lemma 9 (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***.*

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**, and so Lemma 9 is proven. Then, by Lemmas 1 7 and 9 , 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 a 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.

Formally, we set Σ_1 as follows: where $\mathbf{Elem}_{\Sigma_1} = \mathbf{Elem}_\Sigma$ and $\mathbf{Struct}_{\Sigma_1} = \mathit{Fin}(\Sigma)$. $\mathcal{F}_{\Sigma_1} = \mathcal{CO}_{\Sigma_1} \uplus \mathcal{SE}_{\Sigma_1}$, where $\mathcal{CO}_{\Sigma_1} = \{c : \sigma_1 \times \ldots \times \sigma_n \to \sigma \mid c \in \mathcal{CO}_\Sigma, \sigma \in \mathbf{Struct}_{\Sigma_1}\}$ and $\mathcal{SE}_{\Sigma_1}$ and $\mathcal{P}_{\Sigma_1}$ are the corresponding selectors and testers. Notice that if σ is finite and $c : \sigma_1 \times \ldots \times \sigma_n \to \sigma$ is in $\mathcal{CO}_\Sigma$, then σ_i must be finite or in $\mathbf{Elem}_\Sigma$ for every $1 \leq i \leq n$. Next, we set Σ_2 as follows: $\mathcal{S}_{\Sigma_2} = \mathbf{Elem}_{\Sigma_2} \uplus \mathbf{Struct}_{\Sigma_2}$, where $\mathbf{Elem}_{\Sigma_2} = \mathbf{Elem}_\Sigma \cup \mathit{Fin}(\Sigma)$ and $\mathbf{Struct}_{\Sigma_2} = \mathit{Ind}(\Sigma)$. $\mathcal{F}_{\Sigma_2} = \mathcal{CO}_{\Sigma_2} \uplus \mathcal{SE}_{\Sigma_2}$, where $\mathcal{CO}_{\Sigma_2} = \{c : \sigma_2 \times \ldots \times \sigma_n \to \sigma \mid c \in \mathcal{CO}_\Sigma, \sigma \in \mathbf{Struct}_{\Sigma_2}\}$ 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 $S = \mathbf{Elem}_\Sigma \cup \mathit{Fin}(\Sigma)$, $S_1 = \mathbf{Elem}_\Sigma$, $S_2 = \mathbf{Elem}_\Sigma \cup \mathit{Fin}(\Sigma)$, $T_1 = \mathcal{T}_{\Sigma_1}$, 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 we have:

Theorem 4. *If Σ is a datatypes signature then $\mathcal{T}_\Sigma$ 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 11 and then applying the witness from Definition 12 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 9. This witness is simpler than the one produced in the proof from [14] 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 the politeness of $\mathcal{T}_\Sigma$ and some axiomatizations of trees. We show how to get a reduction of any $\mathcal{T}_\Sigma$-satisfiability problem into a satisfiability 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 [9] 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.

$$
\begin{array}{ll}
(Inj) & \{c(X_1,\ldots,X_n) = c(Y_1,\ldots,Y_n) \rightarrow \bigwedge_{i=1}^{n} X_i = Y_i \mid c \in \mathcal{CO}\} \\
(Dis) & \{c(X_1,\ldots,X_n) \neq d(Y_1,\ldots,Y_m) \mid c,d \in \mathcal{CO}, c \neq d\} \\
(Proj) & \{s_{c,i}(c(X_1,\ldots,X_n)) = X_i \mid c \in \mathcal{CO}, i \in [1,n]\} \\
(Is_1) & \{is_c(c(X_1,\ldots,X_n)) \mid c \in \mathcal{CO}\} \\
(Is_2) & \{\neg is_c(d(X_1,\ldots,X_n)) \mid c,d \in \mathcal{CO}, c \neq d\} \\
(Acyc) & \{X \neq t[X] \mid t \text{ is a non-variable } \Sigma_{|\mathcal{CO}}\text{-term that contains } X\ \} \\
(Ext_1) & \{\bigvee_{c:\sigma_1\times\ldots\times\sigma_n\rightarrow\sigma\in\mathcal{CO}} is_c(X) \mid \sigma \in \mathbf{Struct}\} \\
(Ext_2) & \{\exists \overrightarrow{y}\ .\ is_c(X) \rightarrow X = c(\overrightarrow{y}) \mid c \in \mathcal{CO}\}
\end{array}
$$

Fig. 1. Axioms for $TREE_\Sigma$ and $TREE^*_\Sigma$

Proposition 2. *Every $TREE^*_\Sigma$-unsatisfiable formula is $\mathcal{T}_\Sigma$-unsatisfiable.*

Remark 2. Along the lines of [1], a superposition calculus can be applied to get a $TREE_\Sigma$-satisfiability procedure. Such a calculus has been used in [6,9] 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 $TREE_\Sigma$ [9]. 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 $c,d \in \mathcal{CO}, c \neq d$. Further, consider the theory of finite trees defined from $TREE_\Sigma$ by dismissing $Proj, Is_1$ and Is_2. Being defined by Horn clauses, it is convex. Further, it is a Shostak theory [16,18,24] admitting a solver and a canonizer [9]. The solver is given by a syntactic unification algorithm [2] and the canonizer is the identity function. The satisfiability procedure built using the solver and the canonizer can be applied to decide $TREE_\Sigma$-satisfiability problems containing $\Sigma_{|\mathcal{CO}}$-atoms.

The following result shows that any $\mathcal{T}_\Sigma$-satisfiability problem can be reduced to a $TREE_\Sigma$-satisfiability problem. This leads to a $\mathcal{T}_\Sigma$-satisfiability procedure.

Proposition 3. *Let Σ be a finite 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.*

Proposition 3 can be easily lifted to any conjunction of Σ-literals φ by flattening and then guessing all possible arrangements over the variables. Further, $\exists \overrightarrow{w}\ .\ \varphi'$ and φ are not only $\mathcal{T}_\Sigma$-equivalent but also $TREE^*_\Sigma$-equivalent. As a consequence, Proposition 3 also holds when stated using $TREE^*_\Sigma$ instead of $\mathcal{T}_\Sigma$.

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 Sect. 4, $\varphi \mapsto \varphi'$ in the current section); and (*ii*) A small model construction (finite witnessability in Sect. 4, equisatisfiability between $\mathcal{T}_\Sigma$ and *TREE* in Proposition 3). 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 of 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 [23] and co-datatypes [22]. 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 [25]. We want to focus on the politeness of such extensions. Initial work in that direction has been done in [5], that we plan to build on.

Acknowledgments. We are thankful to the anonymous reviewers for their comments.

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 and MIT Press (2001)
3. Barrett, C., Fontaine, P., Tinelli, C.: The SMT-LIB Standard: Version 2.6. Technical report, Department of Computer Science, The University of Iowa (2017). www.SMT-LIB.org
4. Barrett, C.W., Shikanian, I., Tinelli, C.: An abstract decision procedure for a theory of inductive data types. J. Satisfiability Boolean Model. Comput. **3**(1–2), 21–46 (2007)
5. Berthon, R., Ringeissen, C.: Satisfiability modulo free data structures combined with bridging functions. In: King, T., Piskac, R. (eds.) Proceedings of SMT@IJCAR 2016. CEUR Workshop Proceedings, vol. 1617, pp. 71–80. CEUR-WS.org (2016)
6. Bonacina, M.P., Echenim, M.: Rewrite-based satisfiability procedures for recursive data structures. Electron. Notes Theor. Comput. Sci. **174**(8), 55–70 (2007)
7. Bonacina, M.P., Fontaine, P., Ringeissen, C., Tinelli, C.: Theory combination: beyond equality sharing. In: Lutz, C., Sattler, U., Tinelli, C., Turhan, A.-Y., Wolter, F. (eds.) Description Logic, Theory Combination, and All That. LNCS, vol. 11560, pp. 57–89. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-22102-7_3
8. Casal, F., Rasga, J.: Many-sorted equivalence of shiny and strongly polite theories. J. Autom. Reasoning **60**(2), 221–236 (2018)
9. Chocron, P., Fontaine, P., Ringeissen, C.: Politeness and combination methods for theories with bridging functions. J. Autom. Reasoning **64**(1), 97–134 (2020)
10. Enderton, H.B.: A Mathematical Introduction to Logic. Academic Press (2001)
11. Fontaine, P.: Combinations of theories for decidable fragments of first-order logic. In: Ghilardi, S., Sebastiani, R. (eds.) FroCoS 2009. LNCS (LNAI), vol. 5749, pp. 263–278. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04222-5_16
12. Gutiérrez, R., Meseguer, J.: Variant-based decidable satisfiability in initial algebras with predicates. In: Fioravanti, F., Gallagher, J.P. (eds.) LOPSTR 2017. LNCS, vol. 10855, pp. 306–322. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94460-9_18
13. Hojjat, H., Rümmer, P.: Deciding and interpolating algebraic data types by reduction. In: Jebelean, T., Negru, V., Petcu, D., Zaharie, D., Ida, T., Watt, S.M. (eds.) 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2017, Timisoara, Romania, 21–24 September 2017, pp. 145–152. IEEE Computer Society (2017)
14. Jovanović, D., Barrett, C.: Polite theories revisited. In: Fermüller, C.G., Voronkov, A. (eds.) LPAR 2010. LNCS, vol. 6397, pp. 402–416. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16242-8_29
15. Kovács, L., Robillard, S., Voronkov, A.: Coming to terms with quantified reasoning. In: Castagna, G., Gordon, A.D. (eds.) Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, 18–20 January 2017, pp. 260–270. ACM (2017)
16. Krstic, S., Conchon, S.: Canonization for disjoint unions of theories. Inf. Comput. **199**(1–2), 87–106 (2005)

17. Krstić, S., Goel, A., Grundy, J., Tinelli, C.: Combined satisfiability modulo parametric theories. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 602–617. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71209-1_47
18. Manna, Z., Zarba, C.G.: Combining decision procedures. In: Aichernig, B.K., Maibaum, T. (eds.) Formal Methods at the Crossroads. From Panacea to Foundational Support. LNCS, vol. 2757, pp. 381–422. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-40007-3_24
19. Meseguer, J.: Variant-based satisfiability in initial algebras. Sci. Comput. Program. **154**, 3–41 (2018)
20. Nelson, G., Oppen, D.C.: Simplification by cooperating decision procedures. ACM Trans. Program. Lang. Syst. **1**(2), 245–257 (1979)
21. Ranise, S., Ringeissen, C., Zarba, C.G.: combining data structures with nonstably infinite theories using many-sorted logic. In: Gramlich, B. (ed.) FroCoS 2005. LNCS, vol. 3717, pp. 48–64. Springer, Heidelberg (2005). https://doi.org/10.1007/11559306_3. extended technical report is available at https://hal.inria.fr/inria-00070335/
22. Reynolds, A., Blanchette, J.C.: A decision procedure for (co)datatypes in SMT solvers. J. Autom. Reasoning **58**(3), 341–362 (2017)
23. Reynolds, A., Viswanathan, A., Barbosa, H., Tinelli, C., Barrett, C.: Datatypes with shared selectors. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) IJCAR 2018. LNCS (LNAI), vol. 10900, pp. 591–608. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94205-6_39
24. Shostak, R.E.: A practical decision procedure for arithmetic with function symbols. J. ACM **26**(2), 351–360 (1979)
25. Sofronie-Stokkermans, V.: Locality results for certain extensions of theories with bridging functions. In: Schmidt, R.A. (ed.) CADE 2009. LNCS (LNAI), vol. 5663, pp. 67–83. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02959-2_5
26. Tinelli, C., Zarba, C.G.: Combining decision procedures for sorted theories. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS (LNAI), vol. 3229, pp. 641–653. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30227-8_53
27. Tinelli, C., Zarba, C.G.: Combining nonstably infinite theories. J. Autom. Reasoning **34**(3), 209–238 (2005)