Theory Combination: Beyond Equality Sharing

Maria Paola Bonacina[1], Pascal Fontaine[2],
Christophe Ringeissen[2], and Cesare Tinelli[3]

[1] Università degli Studi di Verona, Verona, Italy
[2] Université de Lorraine and INRIA & LORIA, Nancy, France
[3] The University of Iowa, Iowa City, USA

Dedicated to
Franz Baader,
friend and colleague.

Abstract. Satisfiability is the problem of deciding whether a formula has a model. Although it is not even semidecidable in first-order logic, it is decidable in some first-order theories or fragments thereof (e.g., the quantifier-free fragment). *Satisfiability modulo a theory* is the problem of determining whether a quantifier-free formula admits a model that is a model of a given theory. If the formula mixes theories, the considered theory is their union, and *combination of theories* is the problem of combining decision procedures for the individual theories to get one for their union. A standard solution is the *equality-sharing method* by Nelson and Oppen, which requires the theories to be *disjoint* and *stably infinite*. This paper surveys selected approaches to the problem of reasoning in the union of disjoint theories, that aim at going beyond equality sharing, including: *asymmetric* extensions of equality sharing, where some theories are unrestricted, while others must satisfy stronger requirements than stable infiniteness; *superposition-based* decision procedures; and current work on *conflict-driven satisfiability* (CDSAT).

1 Introduction

Since the early 1980s, it was understood that combination of theories is of paramount importance for software verification [99,97,105,98], because program checking requires inferences about diverse domains such as arithmetic, data structures, and free predicate and function symbols [116,51,20]. Reasoning about disjunction is just as basic, since the different paths that a program execution may take are logically connected by disjunction. The problem known as *satisfiability modulo theories* (SMT) refers to the problem of determining the satisfiability of an arbitrary (usually quantifier-free) formula modulo a union of theories [115,14,16]. Several solvers for SMT have been developed in the last 20 years or so that support a combination of two or more theories. These include (in alphabetical order) Alt-Ergo [46], Boolector [41], CVC4 [12], MathSAT [45], OpenSMT [42], Simplify [56], veriT [37], Yices [57], and Z3 [50]. Because of their

power and efficiency in practice, SMT solvers have been interfaced with or integrated in a large number of tools including theorem provers [28,36,109], proofs assistants [1,17], and tools for the analysis, verification, and synthesis of software (see [52] for a survey).

Deductive problems in combination of theories can be formulated as *modularity problems*, or how to get a decision procedure for a problem in a union of theories, given decision procedures for that problem in the component theories. Franz Baader was a pioneer in the study of modularity problems [6,7,8,9,5], propounding the importance of theory combination in automated reasoning.

For the problem of determining the *satisfiability of sets of literals* in a combined theory, an answer to the quest for modularity is offered by the popular *equality sharing method*, by Nelson and Oppen, also known as the *Nelson-Oppen scheme* [99,119,110]. This method combines decision procedures for theories that are *disjoint* and *stably infinite*. In an unsorted setting, this means that the theories' signatures share only the equality symbol and free constants and every quantifier-free formula satisfiable in one of the theories is satisfiable in a model with a countably infinite domain. The Nelson-Oppen scheme *separates* terms that mix function or predicate symbols from different theories by allowing each theory to view maximal *alien* subterms as free constants. The component decision procedures cooperate by agreeing on an *arrangement* of their shared free constants, that is, a complete and consistent set of equalities and disequalities between those constants. For SMT, an equality-sharing decision procedure for a union $\mathcal{T}$ of theories is integrated with a propositional satisfiability (SAT) solver, based on the DPLL/CDCL[4] procedure [49,93,94], according to the DPLL($\mathcal{T}$) framework and its extensions [103,13,85,38,36].

For the problem of *unification modulo theories*, a milestone on the road towards modularity is the *Baader-Schulz combination method* [7]. Unification modulo theories considers *equational theories*, which are presented by sets of universally quantified equalities. The union of two equational theories is the theory presented by the union of the component theories' presentations. Unification is a satisfiability problem that restricts formulas to conjunctions of equations, and models to *Herbrand interpretations*, that is, interpretations where the domain is the universe of terms, and constant and function symbols are interpreted as themselves. In unification *modulo* a theory $\mathcal{T}$, the universe of terms is partitioned into congruence classes induced by the equalities in the presentation of $\mathcal{T}$.

The Baader-Schulz scheme combines decision procedures that allow the addition of free symbols and cooperate by sharing information including an *identification* of free constants, that is, a set of equalities telling which free constants are equal. Both the Baader-Schulz and the Nelson-Oppen schemes require the theories to be disjoint, and feature a separation phase; also, variable identification is a form of arrangement. On the other hand, the Baader-Schulz method does not deal with inequalities, and does not need stable infiniteness.

[4] DPLL stands for Davis-Putnam-Logemann-Loveland and CDCL stands for Conflict-Driven Clause Learning. The CDCL procedure is an extension and improvement of the DPLL procedure.

The *word problem* in an equational theory $\mathcal{T}$ asks whether a universally quantified equality is valid in the theory, that is, satisfied in every model of $\mathcal{T}$. The *Baader-Tinelli combination method* establishes a modularity result for the word problem [9].

For theories with a finite presentation, these problems can be treated also as *refutational theorem-proving* problems, applying a *superposition-based strategy* (e.g., [82,73,74,71,72,10,34,22]) to the union of the presentation and the negation of the conjecture. For the word problem, a conjecture $\forall \bar{x}.\ s \simeq t$, where $\bar{x}$ contains all variables occurring in $s \simeq t$, is negated into $\exists \bar{x}.\ s \not\simeq t$, whose Skolemization yields a ground target inequality $\hat{s} \not\simeq \hat{t}$, where the hat means that all variables are replaced by Skolem constants. A refutation is reached if $\hat{s}$ and $\hat{t}$ get rewritten to the same term. For the unification problem, a conjecture $\exists \bar{x}.\ s \simeq t$, is negated into $\forall \bar{x}.\ s \not\simeq t$, yielding a non-ground target inequality $s \not\simeq t$ with all variables implicitly universally quantified. Superposition also applies into the target inequality (an inference also known as *narrowing*), and a refutation is reached if s and t get reduced to syntatically unifiable terms, leading to *completion-based approaches to unification modulo theories* (e.g., [62,102,89]). Between the word problem and the general validity problem lies the *clausal validity problem*, which queries whether a clause φ, that is, a universally quantified disjunction of literals $\forall \bar{x}.\ l_1 \vee \ldots \vee l_k$, is valid in a theory. The conjecture φ is negated into $\exists \bar{x}.\ \neg l_1 \wedge \ldots \wedge \neg l_k$, whose Skolemization yields a set of ground literals $Q = \{\neg \hat{l}_1, \ldots, \neg \hat{l}_k\}$: φ is valid in the theory if and only if Q is unsatisfiable in the theory. Superposition-based strategies terminate and therefore are *decision procedures* for the satisfiability of sets of ground literals in several theories [4,2,3,26].

In this paper we survey selected approaches to go beyond the standard represented by equality sharing for SMT in unions of theories. We begin with methods that generalize equality sharing to *asymmetric combinations*, where some theories are not stably infinite, provided the others are either *shiny* [122], *gentle* [60], or *polite* [108,76], which means that they are more flexible than stably-infinite theories cardinality-wise. Then we consider superposition, whose application to unions of theories is also formulated as a *modularity problem*, namely *modularity of termination*: knowing that superposition terminates on satisfiability problems in each component theory, show that it terminates on satisfiability problems in their union [2,3]. This modularity results also allows one to understand the relation between equality sharing and superposition [29,3]. We conclude with a brief description of a new paradigm for SMT in unions of theories, named CDSAT, for *Conflict-Driven SATisfiability*, which generalizes equality sharing in several ways, including lifting stable infiniteness [31,32,33]. The interested reader may find additional material in complementary sources (e.g., [92,64,66,67]).

The paper is organized as follows. After providing basic definitions and notations in Section 2, we present the equality-sharing method in Section 3, including a result showing that the decidability of unions of disjoint decidable theories depends on cardinality requirements. The next three sections (4-6) are dedicated to shiny, gentle, and polite theories, respectively. Section 7 surveys the appli-

cation of superposition-based strategies to SMT. Section 8 gives an overview of CDSAT, and Section 9 closes the paper with a discussion.

2 Background Definitions

A *first-order language*, or *signature*, is a tuple $\mathcal{L} = \langle \mathcal{S}, \mathcal{F}, \mathcal{P}, \mathcal{V} \rangle$, where $\mathcal{S}$ is a finite set of disjoint sorts, $\mathcal{F}$ is a finite set of function symbols, $\mathcal{P}$ is a finite set of predicate symbols, including an equality symbol for each sort, and $\mathcal{V}$ is a set that contains a denumerable amount of variables for each sort. Every variable, predicate, and function symbol is assigned a sort in $\mathcal{S}$, $\mathcal{S}^n$, and $\mathcal{S}^{n+1}$ respectively, where n is the arity of the symbol. *Propositions* are nullary predicate symbols, and *constants* are nullary function symbols. $\mathcal{L}$ is *one-sorted* if $\mathcal{S}$ is a singleton, *many-sorted* otherwise. As decision procedures may extend the given language $\mathcal{L}$ with a finite set C of *new* constant symbols, $\mathcal{L}^C$ denotes the extended language $\langle \mathcal{S}, \mathcal{F} \cup C, \mathcal{P}, \mathcal{V} \rangle$. Terms over $\mathcal{L}$, or $\mathcal{L}$-*terms*, are defined as usual. A *compound term* contains at least an occurrence of a function symbol. A *context* is a term with a hole: the notation $t[l]$ represents a term where l appears as subterm in context t.

An atomic formula, or *atom*, is a predicate symbol applied to as many terms as its arity. A *literal* is either an atom or the negation of an atom. *Formulas* are built as usual from atoms, connectives ($\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\equiv$), and quantifiers ($\forall$, $\exists$). A *sentence* is a formula where all variables are quantified. A *clause* is a disjunction of literals, where all variables are implicitly universally quantified. A quantifier-free formula is in *conjunctive normal form* (CNF), if it is a conjunction, or a set, of disjunctions of literals; in *disjunctive normal form* (DNF) if it is a disjunction of conjunctions, or sets, of literals. Through Skolemization, every formula can be reduced to an equisatisfiable conjunction, or set, of clauses (*clausal form*). A term is *ground* if it does not contain variables, and the same applies to literals, clauses, and formulas. The set of variables occurring in a term t is denoted by $Var(t)$. Atoms, literals, sentences, and formulas over $\mathcal{L}$ are called $\mathcal{L}$-*atoms*, $\mathcal{L}$-*literals*, $\mathcal{L}$-*sentences*, and $\mathcal{L}$-*formulas*, respectively.

An *interpretation* $\mathcal{M}$ of $\mathcal{L}$, or $\mathcal{L}$-*interpretation*, defines non-empty pairwise disjoint domains $\mathcal{M}[s]$ for all $s \in \mathcal{S}$, a sort- and arity-matching total function $\mathcal{M}[f]$ for all $f \in \mathcal{F}$, a sort- and arity-matching relation $\mathcal{M}[p]$ for all $p \in \mathcal{P}$, and an element $\mathcal{M}[x] \in \mathcal{M}[s]$ for all $x \in \mathcal{V}$ of sort s. $\mathcal{M}$ designates a value in $\mathcal{M}[s]$ for every term of sort s, and a truth value for every formula, with equality interpreted as identity in each domain. An $\mathcal{L}$-*structure* is an $\mathcal{L}$-interpretation over an empty set of variables. A *model* of an $\mathcal{L}$-formula φ is an $\mathcal{L}$-interpretation where φ is true, written $\mathcal{M} \models \varphi$, and read $\mathcal{M}$ satisfies φ. A formula is *satisfiable* if it has a model, *unsatisfiable* otherwise. A model is finite if for all $s \in \mathcal{S}$ the cardinality $|\mathcal{M}[s]|$ is finite.

In this paper we adopt the syntactic definition of a theory. Given a first-order language $\mathcal{L}$, an $\mathcal{L}$-*theory* $\mathcal{T}$ is a set of $\mathcal{L}$-sentences, called *axioms* or $\mathcal{T}$-*axioms*. One can write $\mathcal{T}$-atom, $\mathcal{T}$-literal, or $\mathcal{T}$-formula in place of $\mathcal{L}$-atom, $\mathcal{L}$-literal, or $\mathcal{L}$-formula. Symbols that do not appear in $\mathcal{T}$-axioms are called *free* or *uninterpreted*.

An $\mathcal{L}$-theory $\mathcal{T}$ determines the set $Mod(\mathcal{T})$ of its models, or $\mathcal{T}$-*models*, that is, those $\mathcal{L}$-structures $\mathcal{M}$ such that $\mathcal{M} \models \mathcal{T}$, which means that $\mathcal{M} \models \varphi$ for all φ in $\mathcal{T}$. In turn, a set $\mathcal{C}$ of $\mathcal{L}$-structures determines the set $Th(\mathcal{C})$ of its theorems, that is, those $\mathcal{L}$-sentences that are true in all structures in $\mathcal{C}$. If $\mathcal{C} = Mod(\mathcal{T})$, one can write $Th(\mathcal{T})$, in place of $Th(\mathcal{C})$, for the set of theorems of $\mathcal{T}$ or $\mathcal{T}$-*theorems*. With respect to $Mod(\mathcal{T})$ or $Th(\mathcal{T})$, $\mathcal{T}$ is called *axiomatization* or *presentation*. If $\mathcal{T}$ is finite, the theory is said to be *finitely axiomatized*. Given an $\mathcal{L}_1$-theory $\mathcal{T}_1$ and an $\mathcal{L}_2$-theory $\mathcal{T}_2$, their *union* is the $\mathcal{L}_1 \cup \mathcal{L}_2$-theory $\mathcal{T}_1 \cup \mathcal{T}_2$. Two languages are *disjoint* if their sets of non-nullary functions and predicates are pairwise disjoint, and two theories are *disjoint* if their languages are.

Whenever equational reasoning is built into an inference system or algorithm, the axioms of equality are omitted from $\mathcal{T}$. If all axioms are built into the inference system or algorithm, a finite axiomatization $\mathcal{T}$ may not be given, and an $\mathcal{L}$-theory, for $\mathcal{L} = \langle \mathcal{S}, \mathcal{F}, \mathcal{P}, \mathcal{V} \rangle$, may be characterized by a set $\mathcal{C}$ of $\mathcal{L}$-structures. The implicit, and usually infinite, axiomatization $\mathcal{T}$ of the theory is given by $\mathcal{T} = Th(\mathcal{C})$, so that the structures in $\mathcal{C}$ are still called $\mathcal{T}$-models and $\mathcal{T}$ is still used as the name of the theory. Given another language $\mathcal{L}' = \langle \mathcal{S}', \mathcal{F}', \mathcal{P}', \mathcal{V}' \rangle$ such that $\mathcal{S} \subseteq \mathcal{S}'$, $\mathcal{F} \subseteq \mathcal{F}'$, and $\mathcal{P} \subseteq \mathcal{P}'$, an $\mathcal{L}'$-structure $\mathcal{M}'$ is a $\mathcal{T}$-*model over* $\mathcal{L}'$, if the $\mathcal{L}$-structure $\mathcal{M}$ defined by the $\mathcal{M}'$-interpretation of $\mathcal{L}$-symbols is a $\mathcal{T}$-model, that is, if $\mathcal{M} \in \mathcal{C}$, or, equivalently, $\mathcal{M} \models \mathcal{T}$. Given an $\mathcal{L}_1$-theory $\mathcal{T}_1$ and an $\mathcal{L}_2$-theory $\mathcal{T}_2$ characterized in this style, their *union* $\mathcal{T}_1 \cup \mathcal{T}_2$ is the $\mathcal{L}_1 \cup \mathcal{L}_2$-theory characterized by the class of $\mathcal{L}_1 \cup \mathcal{L}_2$-structures $\mathcal{M}$ that are simultaneously $\mathcal{T}_1$-models over $\mathcal{L}_1 \cup \mathcal{L}_2$ and $\mathcal{T}_2$-models over $\mathcal{L}_1 \cup \mathcal{L}_2$.

A formula φ is $\mathcal{T}$-*satisfiable* if it has a $\mathcal{T}$-model, $\mathcal{T}$-*unsatisfiable* otherwise. A formula φ is $\mathcal{T}$-*valid* if it is true in all $\mathcal{T}$-models, $\mathcal{T}$-invalid otherwise. Since no interpretation satisfies both φ and $\neg\varphi$, a formula φ is $\mathcal{T}$-valid if and only if $\neg\varphi$ is $\mathcal{T}$-unsatisfiable, and φ is $\mathcal{T}$-satisfiable if and only if $\neg\varphi$ is $\mathcal{T}$-invalid. Thus, $\mathcal{T}$-satisfiability is decidable if and only if $\mathcal{T}$-validity is decidable. The $\mathcal{T}$-validity of φ is approached refutationally by proving that $\neg\varphi$ is $\mathcal{T}$-unsatisfiable: for this purpose, $\neg\varphi$ is typically reduced to clausal form. If φ is a clause (*clausal validity problem*), the clausal form of $\neg\varphi$ is a set of ground unit clauses or, equivalently, ground literals. For many theories of interest only the quantifier-free fragment is decidable (e.g., [39], Chapter 3). Let φ be a quantifier-free formula and $\bar{x}$ the tuple of its free variables: φ is $\mathcal{T}$-satisfiable if and only if its existential closure $\exists\bar{x}.\ \varphi$ is $\mathcal{T}$-satisfiable; φ is $\mathcal{T}$-valid if and only if its universal closure $\forall\bar{x}.\ \varphi$ is $\mathcal{T}$-valid if and only if $\exists\bar{x}.\ \neg\varphi$ is $\mathcal{T}$-unsatisfiable. Through Skolemization, both problems reduce to the $\mathcal{T}$-satisfiability of a ground formula, which can be reduced to either CNF, yielding a set of ground clauses, or DNF. CNF is generally preferred, especially if the original problem is a $\mathcal{T}$-validity problem, as most refutational calculi work with clausal form. If the original problem is a $\mathcal{T}$-satisfiability problem, DNF offers the advantage that a DNF formula is satisfiable if and only if at least one of its sets of literals is. In summary, a theory $\mathcal{T}$ is $\exists$-*decidable*, if the $\mathcal{T}$-satisfiability of finite sets of ground literals is decidable, and $\exists_\infty$-*decidable*, if it is $\exists$-decidable and the satisfiability of finite sets of ground literals in infinite $\mathcal{T}$-models is also decidable.

3 The Equality Sharing Method

SMT solvers are generally built around a decision procedure for quantifier-free formulas, whose Boolean structure is handled by the underlying SAT-solver. As a consequence, theory reasoning is only concerned with conjunctions or (finite) sets of literals. In most SMT solvers, the theory reasoners handle a union of theories. The equality-sharing method by Nelson and Oppen [99,105,97,98,110,119] is a means to build a decision procedure for satisfiability of sets of literals in a union of disjoint theories from decision procedures for satisfiability of sets of literals in each theory. For example, consider the set of literals

$$Q = \{a \leq b,\ b \leq (a + f(a)),\ P(h(a) - h(b)),\ \neg P(0),\ f(a) \simeq 0\}.$$

The first step is to identify the involved theories. Suppose a specification tells us that $\leq$, $+$, $-$, and 0 are to be interpreted over the integers, while the symbols P, f, h, a, and b are free. Since there is no occurrence of product, for the integers it suffices to consider *linear integer arithmetic* (LIA). For the free symbols, the relevant theory is the *theory of (equality with) uninterpreted function symbols* (abbreviated as EUF or UF). Thus, the problem requires the union of LIA and UF sharing the sort int of the integers. However, UF only has equality as predicate, and therefore the problem gets rewritten in the equisatifiable form

$$Q = \{a \leq b,\ b \leq (a + f(a)),\ f_P(h(a) - h(b)) \simeq \bullet,\ f_P(0) \not\simeq \bullet,\ f(a) \simeq 0\}.$$

Let prop be the sort interpreted by all interpretations as the set $\{\mathsf{true}, \mathsf{false}\}$ of the propositional, or Boolean, values. Then, the language of UF has sorts int and prop, function symbols $f, h\colon \mathsf{int} \to \mathsf{int}$ and $f_P\colon \mathsf{int} \to \mathsf{prop}$, and constant symbols a and b of sort int, and $\bullet$ of sort prop. The language of LIA has sorts int and prop, predicate symbol $\leq$ of sort $\mathsf{int} \times \mathsf{int}$ for the ordering, function symbols $+, -\colon \mathsf{int} \times \mathsf{int} \to \mathsf{int}$ for addition and subtraction, and the constant 0 of sort int. Let $\mathcal{T}_1$ be LIA and $\mathcal{T}_2$ be UF. For the separation phase of the equality-sharing method, Q is *separated* into a set of $\mathcal{T}_1$-literals Q_1 and a set of $\mathcal{T}_2$-literals Q_2 by introducing fresh free constants[5] to produce the equisatisfiable problem $Q_1 \cup Q_2$:

$$Q_1 = \{a \leq b,\ b \leq (a + v_1),\ v_2 \simeq v_3 - v_4,\ v_5 \simeq 0,\ v_1 \simeq 0\}$$
$$Q_2 = \{v_1 \simeq f(a),\ f_P(v_2) \simeq \bullet,\ v_3 \simeq h(a),\ v_4 \simeq h(b),\ f_P(v_5) \not\simeq \bullet\}.$$

Q_1 and Q_2 only share equality between terms of sort int and the free constants in the set $C = \{a, b, v_1, v_2, v_3, v_4, v_5\}$. It is not difficult to see that Q is $\mathcal{T}_1 \cup \mathcal{T}_2$-unsatisfiable whereas Q_1 is $\mathcal{T}_1$-satisfiable and Q_2 is $\mathcal{T}_2$-satisfiable. This means that, in general, it is not sufficient for $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiability to let the decision procedures for $\mathcal{T}_1$ and $\mathcal{T}_2$ examine only the satisfiability of their subproblem. The decision procedures need to exchange information about their individual sets of literals. A first key idea in equality sharing is that the decision procedures need to *agree* on an *arrangement* of the shared constants.

[5] Traditionally combination schemes use free variables for this role (e.g., [39]). Since quantified formulas appear in this paper, we choose to use free constants.

Definition 1. *An* arrangement α *of a set of constant symbols* C *is a satisfiable set of sorted equalities and inequalities between elements of* C *such that* $a \simeq b \in \alpha$ *or* $a \not\simeq b \in \alpha$ *for all* a, b *of the same sort in* C*.*

A second key ingredient is that the decision procedures need to *agree* on the *cardinalities* of the shared sorts. The following theorem states these two requirements for completeness (cf. [120,61,121,122,60] for equivalent formulations).

Theorem 1. *Assume* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *are theories over disjoint languages* $\mathcal{L}_1$ *and* $\mathcal{L}_2$*, and* Q_i *(*$i = 1, 2$*) is a set of* $\mathcal{L}_i^C$*-literals. Then* $Q_1 \cup Q_2$ *is* $\mathcal{T}_1 \cup \mathcal{T}_2$*-satisfiable if and only if there exist an arrangement* α *of* C *and a* $\mathcal{T}_i$*-model* $\mathcal{M}_i$ *such that* $\mathcal{M}_i \models \alpha \cup Q_i$ *for* $i = 1, 2$ *and* $|\mathcal{M}_1[s]| = |\mathcal{M}_2[s]|$ *for all sorts* s *common to both languages* $\mathcal{L}_1$ *and* $\mathcal{L}_2$*.*

This theorem can be strengthened by restricting the arrangement α to the constants of C that occur in both Q_1 and Q_2. The ($\Rightarrow$) case of the proof is straightforward, and the ($\Leftarrow$) case is proved by building from the $\mathcal{T}_1$-model and the $\mathcal{T}_2$-model a $\mathcal{T}_1 \cup \mathcal{T}_2$-model (e.g., Theorem 1, [60]). This is possible thanks to the shared arrangement, and because the $\mathcal{T}_1$-model and the $\mathcal{T}_2$-model have the same cardinality for each common sort. Checking the existence of a model is the task of the decision procedures for the component theories. The issue is how to ensure that there are models that agree on the cardinalities of shared sorts.

A theory $\mathcal{T}$ is *stably infinite* if every $\mathcal{T}$-satisfiable quantifier-free $\mathcal{T}$-formula has a $\mathcal{T}$-model such that for all sorts other than prop the domain has cardinality $\aleph_0$, the cardinality of the set $\mathbb{N}$ of the natural numbers. Combining only stably infinite theories is a radical solution to the cardinality requirement of Theorem 1: the cardinality is $\aleph_0$ for all shared theories other than prop. Since both LIA and UF are stably infinite, the set Q of our example is $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable if and only if there exists an arrangement α of the free constants in C such that $\alpha \cup Q_i$ is $\mathcal{T}_i$-satisfiable for i=1, 2. As no such arrangement exists, Q is $\mathcal{T}_1 \cup \mathcal{T}_2$-unsatisfiable. In order to include non-stably-infinite theories, combination schemes can rely on the notion of *spectrum* [60].

Definition 2. *The* spectrum *of a one-sorted theory* $\mathcal{T}$ *is the set of the cardinalities of the* $\mathcal{T}$*-models.*[6]

This notion can be generalized to the many-sorted case by considering tuples of cardinalities, one cardinality for each sort. Using this definition and Theorem 1, it is possible to state completeness requirements for a combination scheme for disjoint theories that are not necessarily stably infinite (cf. Corollary 1, [60]).

Corollary 1. *Given theories* $\mathcal{T}_1$ *and* $\mathcal{T}_2$ *over disjoint languages* $\mathcal{L}_1$ *and* $\mathcal{L}_2$*,* $\mathcal{T}_1 \cup \mathcal{T}_2$ *is* $\exists$*-decidable if, for all sets of* $\mathcal{L}_1^C$*-literals* Q_1 *and* $\mathcal{L}_2^C$*-literals* Q_2*, it is possible to determine whether the intersection of the spectra of* $\mathcal{T}_1 \cup Q_1$ *and* $\mathcal{T}_2 \cup Q_2$ *is non-empty for each sort common to both languages* $\mathcal{L}_1$ *and* $\mathcal{L}_2$*.*

[6] The spectrum of a theory is usually defined as the set of the *finite* cardinalities of its models. We extend the definition slightly for convenience.

For stably infinite theories, if $\mathcal{T}_1 \cup Q_1$ and $\mathcal{T}_2 \cup Q_2$ are satisfiable, the intersection of their spectra contains $\aleph_0$ for each shared sort, and therefore the following classic combination lemma does not need to mention cardinalities.

Combination Lemma 1 (Stably Infinite Theories) *Assume two stably infinite disjoint theories $\mathcal{T}_1$ and $\mathcal{T}_2$ over languages $\mathcal{L}_1$ and $\mathcal{L}_2$, and let Q_i be a set of $\mathcal{L}_i^C$-literals ($i = 1, 2$). Then $Q_1 \cup Q_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable if and only if there exists an arrangement α of C such that $\alpha \cup Q_i$ is T_i-satisfiable for $i = 1, 2$.*

Since the union of disjoint stably infinite theories is stably infinite, the class of $\exists$-decidable stably infinite theories is closed under disjoint union.

Theorem 2. *The union of disjoint, stably infinite, $\exists$-decidable theories is stably infinite and $\exists$-decidable.*

A key result is that some cardinality requirement is necessary for decidability. This finding is a consequence of the following theorem (cf. Proposition 4.1, [29]).

Theorem 3. *There exist an $\exists$-decidable theory that is not $\exists_\infty$-decidable.*

The proof exhibits a theory TM_∞ with language $\mathcal{L}_{TM_\infty}$, including infinitely many nullary predicates $P_{(e,n)}$ for all $e \in \mathbb{N}$ and $n \in \mathbb{N}$. The meaning of $P_{(e,n)}$ is that e is the index of a Turing machine, and n is an input for the Turing machine of index e. The axioms of TM_∞ involve a kind of clauses called *at-most cardinality constraints*. An *at-most cardinality constraint* is a clause containing only non-trivial (i.e., other than $x \simeq x$) equalities between variables. For example, $\forall x, y, z.\ y \simeq x \vee y \simeq z$ is the at-most-2 cardinality constraint: a model of this clause can have at most 2 elements since the clause says that out of 3 variables at least 2 must be equal. In general,

$$\forall x_0, \ldots, x_m. \bigvee_{0 \le i \ne k \le m} x_i \simeq x_k$$

is the *at-most-m cardinality constraint*: a model of this clause can have at most m elements. The axioms of TM_∞ are all the formulas saying that $P_{(e,n)}$ implies the at-most-m cardinality constraint, if Turing Machine e halts on input n in fewer than m steps. The property of being an axiom of TM_∞ is decidable, because it suffices to run the Turing machine and see whether it halts in fewer than m steps. The TM_∞-satisfiability of a finite set Q of ground $\mathcal{L}_{TM_\infty}^C$-literals is also decidable: intuitively, as Q involves finitely many constants, their arrangement dictates the minimum cardinality m of a candidate model; if Q contains both $P_{(e,n)}$ and $\neg P_{(e,n)}$, it is unsatisfiable; otherwise, for each $P_{(e,n)} \in Q$ it suffices to test that Turing Machine e runs on input n for at least m steps. On the other hand, satisfiability in infinite TM_∞-models is undecidable: $Q = \{P_{(e,n)}\}$ is satisfiable in an infinite TM_∞-model if and only if Turing Machine e does not halt on input n, which is undecidable for being the complement of the Halting Problem. While TM_∞ has an infinite language, it is possible to exhibit a theory with the same decidability properties of TM_∞ and a finite language [30]. It follows as a corollary that the union of this theory with any disjoint $\exists$-decidable theory with only infinite models is not $\exists$-decidable (cf. Theorem 4.1, [29]).

Theorem 4. *There exist ∃-decidable disjoint theories whose union is not ∃-decidable.*

This theorem implies that if we want to lift the stable infiniteness requirement on one or more component theories, while maintaining decidability of theory unions, we still need to impose some restrictions on the cardinality of these theories' models. Such restrictions must allow the theories to agree on the cardinality of a joint model, so that Corollary 1 can apply. In the next three sections, we survey combination schemes that achieve precisely such a synchronization of the theories on models' cardinalities without imposing stable infiniteness.

4 Shiny Theories

The theory of uninterpreted symbols (UF) is one of the most useful theories. It is convenient to model arrays, generic functions, or other data structures described by a custom set of axiom handled separately by the reasoner through instantiation or other inferences (e.g., see Sect. 7). This theory is *shiny* [122], a much stronger property than being stably infinite.

Definition 3 (Shiny Theory). *A theory $\mathcal{T}$ over a one-sorted language $\mathcal{L}$ is* shiny *if, for all sets Q of $\mathcal{L}^C$-literals, either the spectrum of $\mathcal{T} \cup Q$ is empty or it is the set of all cardinalities greater than or equal to a finite cardinality $mincard_{\mathcal{T}}(Q)$ computable from Q.*

For UF, if Q is unsatisfiable, the spectrum is empty. If Q is satisfiable, let s be its sort. An arrangement of the finitely many constant symbols that appear in Q determines a finite cardinality for the domain $\mathcal{M}[s]$ of a model $\mathcal{M}$. A model $\mathcal{M}'$ such that $|\mathcal{M}'[s]| > |\mathcal{M}[s]|$ is obtained by taking a non-empty set A disjoint from $\mathcal{M}[s]$, and letting $\mathcal{M}'[s]$ be $\mathcal{M}[s] \cup A$. $\mathcal{M}'$ interprets equality as identity on every pair of elements of A, and is otherwise identical to $\mathcal{M}$. This is the argument showing that UF is stably infinite (e.g., see Example 10.3, [39]) plus the observation that an arrangement yields a finite cardinality.

The spectrum of a shiny theory $\mathcal{T}$ is *upward closed from $mincard_{\mathcal{T}}(Q)$* or *upward closed* for short: if Q has a $\mathcal{T}$-model, it has a $\mathcal{T}$-model for every larger cardinality. A theory with such a spectrum is called *smooth*. A shiny theory also has the *finite-model property*: if a set Q of $\mathcal{L}^C$-literals is $\mathcal{T}$-satisfiable, it is satisfiable in a finite model of $\mathcal{T}$. Consider the union of a shiny theory $\mathcal{T}_1$ and an arbitrary theory $\mathcal{T}_2$ such that $\mathcal{T}_1$ and $\mathcal{T}_2$ are disjoint and share the one sort s of $\mathcal{T}_1$. For $\mathcal{T}_1$ and $\mathcal{T}_2$ to agree on the cardinality of s it suffices that there is a $\mathcal{T}_2$-model that interprets s with a domain of sufficiently large cardinality. An *at-least cardinality constraint* expresses this requirement. An at-least cardinality constraint is the negation of an at-most cardinality constraint (see Sect. 3), hence it is a conjunction of non-trivial inequalities between variables, all existentially quantified. Through Skolemization, the clausal form of an at-least cardinality constraint is a set of inequalities between constants, known as an *all-different constraint*.

Definition 4. *Given a positive integer m and a sort s, an all-different constraint $\delta_s(m)$ for sort s is a set of literals $\{c_i \not\approx c_j \mid 1 \leq i \neq j \leq n\}$, where $c_1, \ldots, c_m$ are distinct fresh free constants of sort s.*

For a theory $\mathcal{T}$, whose language has a sort s, and a set Q of $\mathcal{T}$-literals, Q has a $\mathcal{T}$-model that interprets s with a domain of cardinality at least m if and only if the set of literals $Q \cup \delta_s(m)$ is $\mathcal{T}$-satisfiable. Thus, the union of a shiny theory $\mathcal{T}_1$ and an arbitrary theory $\mathcal{T}_2$, sharing $\mathcal{T}_1$'s only sort s, is handled by testing in this manner that $\mathcal{T}_2$ has a model $\mathcal{M}$ such that $|\mathcal{M}[s]| \geq m$, with m determined by applying $mincard_{\mathcal{T}_1}$ to the set of $\mathcal{T}_1$-literals and the arrangement.

Combination Lemma 2 (Shiny Theory) *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be disjoint languages such that $\mathcal{L}_1$ has only one sort s shared with $\mathcal{L}_2$. Assume a shiny $\mathcal{L}_1$-theory $\mathcal{T}_1$ and an arbitrary $\mathcal{L}_2$-theory $\mathcal{T}_2$, and let Q_i be a set of $\mathcal{L}_i^C$-literals ($i = 1, 2$). Then $Q_1 \cup Q_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable if and only if there exists an arrangement α of C such that $\alpha \cup Q_1$ is $\mathcal{T}_1$-satisfiable and $\alpha \cup Q_2 \cup \delta_s(mincard_{\mathcal{T}_1}(\alpha \cup Q_1))$ is $\mathcal{T}_2$-satisfiable.*

Since one theory is shiny and the other is arbitrary, the combination scheme is *asymmetric*.

Theorem 5. *The union of a shiny $\exists$-decidable theory with an arbitrary $\exists$-decidable theory that shares the single sort of the shiny theory is $\exists$-decidable.*

The generalization of shininess to many-sorted theories leads to *politeness* (see Section 6). Several other theories have a spectrum that is not upward closed, but satisfies other useful properties for asymmetric combination schemes, as captured by the concept of *gentleness* in the next section.

5 Gentle Theories

By weakening the requirements on the spectrum of theories, more theories can take part in asymmetric combinations. *Gentleness* is weaker than shininess and captures several other interesting $\exists$-decidable theories [60].

Definition 5. *A theory $\mathcal{T}$ over a one-sorted language $\mathcal{L}$ is* gentle *if, for all sets Q of $\mathcal{L}^C$-literals, the spectrum of $\mathcal{T} \cup Q$ is computable and is equal to*

1. *Either a finite set of finite cardinalities, or*
2. *A co-finite set of cardinalities given by the union of a finite set of finite cardinalities and the set of all (finite and infinite) cardinalities greater than a computable finite cardinality.*

A gentle theory $\mathcal{T}$ is not necessarily stably infinite, since a $\mathcal{T}$-satisfiable set of literals may have only finite models by Case (1) of Definition 5. A shiny theory is gentle, since it satisfies Case (2) of Definition 5 with an empty set of finite cardinalities. Thus, UF is gentle. Conversely, the spectrum of a gentle

theory is like the spectrum of a shiny theory with the addition of finitely many finite cardinalities. Gentle theories are $\exists$-decidable, and theories in the *Bernays-Schönfinkel-Ramsey class* (axioms of the form $\exists^*\forall^*\varphi$, where φ is quantifier-free and without occurrences of non-nullary function symbols), *Löwenheim class* (axioms in first-order relational monadic logic, i.e., no non-nullary functions and only unary predicates), and FO^2 *class* (axioms with only two variables and no non-nullary functions) are gentle [60].

Theorem 6. *The union of disjoint gentle theories is gentle.*

The proof (see [60]) rests on the observation that the intersection of the spectra of gentle theories is a spectrum that satisfies Definition 5.

Let $mincard_{\mathcal{T}}$ be the partial function from sets of $\mathcal{T}$-literals to cardinal numbers defined by $mincard_{\mathcal{T}}(Q)=k$, if k is the smallest non-zero finite cardinality such that the spectrum of $\mathcal{T} \cup Q$ is upward closed from k. If Q is $\mathcal{T}$-unsatisfiable, or the spectrum of $\mathcal{T} \cup Q$ is bounded and only contains a finite number of finite cardinalities, then $mincard_{\mathcal{T}}(Q)$ is undefined. Let $fincard_{\mathcal{T}}$ be the function that maps a set Q of $\mathcal{T}$-literals to a finite, possibly empty, set of finite cardinalities of $\mathcal{T}$-models of Q as follows: (i) if the spectrum of $\mathcal{T} \cup Q$ is empty, $fincard_{\mathcal{T}}(Q)$ is empty; (ii) if the spectrum of $\mathcal{T} \cup Q$ is finite, $fincard_{\mathcal{T}}(Q)$ is the spectrum of $\mathcal{T} \cup Q$; (iii) otherwise, $fincard_{\mathcal{T}}(Q)$ is the set of the cardinalities in the spectrum of $\mathcal{T} \cup Q$ that are strictly smaller than $mincard_{\mathcal{T}}(Q)$.

Combination Lemma 3 (Gentle Theory) *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be disjoint languages such that $\mathcal{L}_1$ has only one sort s shared with $\mathcal{L}_2$. Assume a gentle $\mathcal{L}_1$-theory $\mathcal{T}_1$ and an arbitrary $\mathcal{L}_2$-theory $\mathcal{T}_2$, and let Q_i be a set of $\mathcal{L}_i^C$-literals ($i = 1, 2$). Then $Q_1 \cup Q_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable if and only if there exists an arrangement α of C such that $\alpha \cup Q_1$ is $\mathcal{T}_1$-satisfiable and*

1. *Either $mincard_{\mathcal{T}_1}(\alpha \cup Q_1)$ is defined and $\alpha \cup Q_2 \cup \delta_s(mincard_{\mathcal{T}_1}(\alpha \cup Q_1))$ is $\mathcal{T}_2$-satisfiable,*
2. *Or there exists a cardinality $k \in fincard_{\mathcal{T}_1}(Q_1 \cup \alpha)$ such that $\alpha \cup Q_2$ is $\mathcal{T}_2$-satisfiable in a $\mathcal{T}_2$-model $\mathcal{M}$ such that $|\mathcal{M}[s]| = k$.*

Condition (1) follows the pattern of Combination Lemma 2. For Condition (2), note that for finitely axiomatized one-sorted theories, it is decidable whether there is a model of a given finite cardinality k. Indeed, there are finitely many (up to isomorphism) interpretations of cardinality k for a finite language, and it takes a finite amount of time to check whether such an interpretation satisfies the axioms. Several widely used theories are not gentle, but they can be combined with gentle theories [60].

Theorem 7. *Given disjoint one-sorted theories $\mathcal{T}_1$ and $\mathcal{T}_2$, where $\mathcal{T}_1$ is gentle, their union $\mathcal{T}_1 \cup \mathcal{T}_2$ is $\exists$-decidable, provided that:*

1. *$\mathcal{T}_2$ also is gentle, or*
2. *$\mathcal{T}_2$ is an $\exists$-decidable finitely axiomatized theory, or*

3. *$\mathcal{T}_2$ is an $\exists$-decidable theory that only admits a fixed finite (possibly empty) known set of finite cardinalities for its models, and possibly infinite models.*

The proof shows that in each case of Theorem 7, either Condition (1) or Condition (2) of Combination Lemma 3 applies, possibly after some preliminary work. For example, if both $\mathcal{T}_1$ and $\mathcal{T}_2$ are gentle, the intersection of their spectra is computed first. In Case (3), either the intersection of the finite sets of finite cardinalities admitted by the two theories is non-empty, or else the Löwenheim-Skolem theorem for first-order logic (if a theory has an infinite model, it has models for every infinite cardinality) is invoked to imply that the spectrum of $\mathcal{T}_2$ is upward closed from some infinite cardinality. For example, the theory of arrays (e.g., [95] and Chapter 3 [39]) belongs to Case (2) of Theorem 7, while LIA and the theory of real closed fields (RCF) fit in Case (3).

Gentleness can be extended to many-sorted theories as done for *$\mathcal{P}$-gentleness*, a generalization of gentleness introduced to handle unions of non-disjoint theories sharing only unary predicates [43].

6 Polite Theories

Politeness can be considered as a many-sorted extension of shininess [108,76]. The concept of politeness is instrumental to combine data-structure theories with an arbitrary theory of elements, as illustrated below for *arrays*, *records*, *sets*, and *multisets*. In this section, we work directly with quantifier-free formulas, instead of sets of literals. A theory is *polite* with respect to a given set $\mathcal{S}$ of sorts, if it is *smooth* and *finitely witnessable* with respect to $\mathcal{S}$. As seen in Section 4 for the one-sorted case, *smooth* means that it is possible to enlarge arbitrarily the $\mathcal{S}$-sorted domains of a model to get another model with the desired cardinalities.

Definition 6 (Smooth Theory). *An $\mathcal{L}$-theory $\mathcal{T}$ is* smooth *with respect to a set $\mathcal{S} = \{s_1, \ldots, s_n\}$ of sorts of its many-sorted language $\mathcal{L}$, if:*

- *For all ground formulas φ in $\mathcal{L}^C$,*
- *For all $\mathcal{T}$-models $\mathcal{M}$ of φ,*
- *For all cardinal numbers $k_1, \ldots, k_n$ such that $k_i \geq |\mathcal{M}[s_i]|$, for $i = 1, \ldots, n$,*

there exists a $\mathcal{T}$-model $\mathcal{N}$ of φ such that $|\mathcal{N}[s_i]| = k_i$ for $i = 1, \ldots, n$.

Finite witnessability complements smoothness by establishing a starting point for the upward movement. The starting point is given by a finite $\mathcal{T}$-model obtained from formulas called *finite witnesses* (see Sects. 6.1 and 6.2 for examples).

Definition 7 (Finite Witness). *Let $\mathcal{S}$ be a set of sorts of a many-sorted language $\mathcal{L}$ and $\mathcal{T}$ an $\mathcal{L}$-theory. Given a ground $\mathcal{L}^C$-formula φ, a ground $\mathcal{L}^D$-formula ψ, where $D \supseteq C$, is a* finite witness of φ in $\mathcal{T}$ with respect to $\mathcal{S}$ *if:*

1. *For all $\mathcal{T}$-models $\mathcal{M}$ over $\mathcal{L}^D$, $\mathcal{M} \models (\varphi \equiv \psi)$, and*

2. *For all arrangements α of all the $\mathcal{S}$-sorted constants in D, if $\{\psi\} \cup \alpha$ is $\mathcal{T}$-satisfiable then there exists a $\mathcal{T}$-model $\mathcal{M}^*$ of $\{\psi\} \cup \alpha$ such that $\mathcal{M}^*[s] = \{\mathcal{M}^*[d] \mid d \in D,\ d \text{ of sort } s\}$, for all $s \in \mathcal{S}$.*

Thanks to Property (2) in this definition, finite witnesses provide the finite $\mathcal{T}$-model $\mathcal{M}^*$ that is the starting point for the upward movement made possible by smoothness: for all $s \in \mathcal{S}$, the domain $\mathcal{M}^*[s]$ comprises precisely the elements used to interpret the constant symbols occuring in the finite witness.

Definition 8 (Finitely Witnessable Theory). *An $\mathcal{L}$-theory $\mathcal{T}$ is* finitely witnessable *with respect to a set $\mathcal{S}$ of sorts of $\mathcal{L}$, if there exists a computable function* witness *such that, for all ground formulas φ in $\mathcal{L}^C$,* witness(φ) *is a finite witness of φ in $\mathcal{T}$ with respect to $\mathcal{S}$.*

Definition 9 (Polite Theory). *An $\mathcal{L}$-theory $\mathcal{T}$ is* polite *with respect to a set of sorts $\mathcal{S}$ of $\mathcal{L}$, if it is smooth and finitely witnessable with respect to $\mathcal{S}$.*

In the classical Nelson-Oppen procedure for disjoint stably infinite theories, it suffices to compute an arrangement of shared constants (see Sect. 3). Polite theories allow us to extend the equality-sharing scheme to non-stably-infinite theories [108], provided the procedure computes an arrangement of a larger set of constants, that includes the new ones introduced by witnesses [76].

Combination Lemma 4 (Polite Theory) *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be disjoint languages sharing a set $\mathcal{S}$ of sorts. Assume an $\mathcal{L}_1$-theory $\mathcal{T}_1$ polite with respect to $\mathcal{S}$ and an arbitrary $\mathcal{L}_2$-theory $\mathcal{T}_2$. Let φ_i be a ground $\mathcal{L}_i^C$-formula ($i = 1, 2$), and* witness(φ_1) *be a ground formula in $\mathcal{L}_1^D$, $D \supseteq C$, that is a finite witness of φ_1 in $\mathcal{T}_1$ with respect to $\mathcal{S}$. Then $\varphi_1 \wedge \varphi_2$ is $\mathcal{T}_1 \cup \mathcal{T}_2$-satisfiable if and only if there exists an arrangement α of all $\mathcal{S}$-sorted constants in D such that $\alpha \wedge$* witness(φ_1) *is $\mathcal{T}_1$-satisfiable and $\alpha \wedge \varphi_2$ is $\mathcal{T}_2$-satisfiable.*

The ($\Rightarrow$) case of the proof is straightforward. For the ($\Leftarrow$) case the reasoning goes as follows. First, by Property (1) of Definition 7, witness(φ_1) is equivalent to φ_1 in $\mathcal{T}_1$. Second, by Property (2) of Definition 7, witness(φ_1) determines finite cardinalities for the shared sorts. Third, by smoothness of $\mathcal{T}_1$, it is possible to scale up these cardinalities to meet those required by the $\mathcal{T}_2$-model. As there is agreement on both shared constants, as provided by the arrangement, and cardinalities of shared sorts, the result follows by Theorem 1. Since one theory is polite and the other is arbitrary, the combination scheme is *asymmetric*.

Theorem 8. *The union of two $\exists$-decidable disjoint theories is $\exists$-decidable, if one of them is polite with respect to the set of sorts shared by the two theories.*

The main difficulty in applying this theorem is to show that a given theory is polite, and especially to show that finite witnesses can be computed for all input formulas. All known polite theories are theories of data structures. We distinguish two classes of polite data-structure theories. The first one comprises theories characterized by sets of *standard interpretations*, and is covered in Sect. 6.1:

the theories of *arrays*, *records*, *sets*, and *multisets* belong to this class. All these theories feature an *extensionality axiom* whereby two data structures are equal if and only if their corresponding elements are. For all these theories, a witness function can be built by using a common principle based on the translation of an inequality of data structures into some constraint on their elements. The second class includes extensions of UF with axioms such as *projection*, *injectivity*, or *acyclicity*, leading to axiomatizations of *recursive data structures*, also known as *algebraic data types*, and is covered in Sect. 6.2. For this second class of theories, a witness function can be defined using the saturated set of clauses computed by a *superposition-based decision procedure* (see Sect. 7).

6.1 Arrays, Records, Sets, and Multisets

The *theory of arrays* $\mathcal{T}_{\mathsf{array}}$ is an especially important example of polite theory. This theory is widely studied and usually presented as an axiomatized theory (e.g., [95,4,2,40,39,3]). Its language $\mathcal{L}_{\mathsf{array}}$ has a sort elem for elements, a sort index for indices, a sort array for arrays, and the function symbols $\mathsf{read}\colon \mathsf{array} \times \mathsf{index} \to \mathsf{elem}$ and $\mathsf{write}\colon \mathsf{array} \times \mathsf{index} \times \mathsf{elem} \to \mathsf{array}$. Semantically, an array is seen as a function $a\colon I \to E$ from some set of *indices* to some set of *elements*. We use the notation E^I for the set of functions from I to E. Consider an $\mathcal{L}_{\mathsf{array}}$-interpretation $\mathcal{M}$ with domains $\mathcal{M}[\mathsf{elem}] = E$ and $\mathcal{M}[\mathsf{index}] = I$. $\mathcal{M}$ interprets a term of sort array as a function $a\colon I \to E$. However, arrays can be updated, and $\mathcal{L}_{\mathsf{array}}$ employs the function write to represent such an update. For the interpretation of write, a function $a_{i \mapsto e}\colon I \to E$ is defined as follows: $a_{i \mapsto e}(i) = e$ and $a_{i \mapsto e}(j) = a(j)$, for $j \neq i$. Then, a *standard* $\mathcal{L}_{\mathsf{array}}$*-interpretation* $\mathcal{M}$ is an $\mathcal{L}_{\mathsf{array}}$-interpretation such that: $\mathcal{M}[\mathsf{array}] = (\mathcal{M}[\mathsf{elem}])^{\mathcal{M}[\mathsf{index}]}$, $\mathcal{M}[\mathsf{read}](a, i) = a(i)$ for all $a \in \mathcal{M}[\mathsf{array}]$ and $i \in \mathcal{M}[\mathsf{index}]$, and $\mathcal{M}[\mathsf{write}](a, i, e) = a_{i \mapsto e}$ for all $a \in \mathcal{M}[\mathsf{array}]$, $i \in \mathcal{M}[\mathsf{index}]$, and $e \in \mathcal{M}[\mathsf{elem}]$. $\mathcal{T}_{\mathsf{array}}$ is the $\mathcal{L}_{\mathsf{array}}$-theory characterized by the *class of all standard* array*-structures*. The following *extensionality axiom* is $\mathcal{T}_{\mathsf{array}}$-valid:

$$\forall x, y\colon \mathsf{array}.\ (x \simeq y) \equiv (\forall z\colon \mathsf{index}.\ \mathsf{read}(x, z) \simeq \mathsf{read}(y, z)).$$

The clausal form of its ($\Leftarrow$) direction is $x \simeq y \vee \mathsf{read}(x, sk(x, y)) \not\simeq \mathsf{read}(y, sk(x, y))$, where the Skolem term $sk(x, y)$ represents the index where x and y differ, the "witness" that the two arrays are different. A finite witness of a set Q of $\mathcal{T}_{\mathsf{array}}$-literals is constructed by replacing each array-sorted inequality $l \not\simeq r$ in Q with $\mathsf{read}(l, i) \not\simeq \mathsf{read}(r, i)$, where i is a new constant symbol of sort index. This transformation originated as a reduction of the satisfiability of sets of ground literals in $\mathcal{T}_{\mathsf{array}}$ with extensionality to the satisfiability of sets of ground literals in $\mathcal{T}_{\mathsf{array}}$ without extensionality [4]: using a Skolem constant i in place of the compound Skolem term $sk(x, y)$ preserves equisatisfiability. Intuitively, adding constants to name the positions where arrays differ suffices to glean from the number of constants occurring in the set of literals the minimum cardinalities for sorts elem and index, leading to the following politeness result [108].

Theorem 9. *$\mathcal{T}_{\mathsf{array}}$ is polite with respect to* $\{\mathsf{elem}, \mathsf{index}\}$.

Basically, $\mathcal{T}_{\mathsf{array}}$ inherits smoothness with respect to $\{\mathsf{elem}, \mathsf{index}\}$ from the shininess of UF. Once extensionality has been eliminated, the reasoning focuses solely on equality of indices and elements. The remaining occurrences of write can be eliminated by a *case analysis* with respect to the read-over-write axioms, on whether the index-sorted argument of read is equal or different from that of the nested write (see Sect. 9.5, [39]). Once all occurrences of write have been eliminated, $\mathcal{T}_{\mathsf{array}}$ essentially reduces to UF, as a term $\mathsf{read}(l, i)$ can be written as $f_a(l)$, by introducing a free function symbol f_a for every array-term l (e.g., Chapter 9, [39]).

Records aggregate attribute-value pairs and resemble arrays if attributes are considered as indices [2,108,3]): if there are n attributes, the set of "indices" has cardinality n. The *theory of records* $\mathcal{T}_{\mathsf{rec}}$ has a language $\mathcal{L}_{\mathsf{rec}}$ with a sort rec for records, a sort elem for values, and a pair of read and write function symbols for each attribute: $\mathsf{read}_i \colon \mathsf{rec} \to \mathsf{elem}$ and $\mathsf{write}_i \colon \mathsf{rec} \times \mathsf{elem} \to \mathsf{rec}$, for $i = 1, \ldots, n$. A *standard* rec*-interpretation* $\mathcal{M}$ is an $\mathcal{L}_{\mathsf{rec}}$-interpretation such that: $\mathcal{M}[\mathsf{rec}] = (\mathcal{M}[\mathsf{elem}])^n$, $\mathcal{M}[\mathsf{read}_i](a) = a(i)$ for all $a \in \mathcal{M}[\mathsf{rec}]$, for $i = 1, \ldots, n$, and $\mathcal{M}[\mathsf{write}](a, i, e) = a_{i \mapsto e}$ for all $a \in \mathcal{M}[\mathsf{rec}]$ and $e \in \mathcal{M}[\mathsf{elem}]$, for $i = 1, \ldots, n$. $\mathcal{T}_{\mathsf{rec}}$ is the $\mathcal{L}_{\mathsf{rec}}$-theory characterized by the *class of all standard* rec*-structures*. The $\mathcal{T}_{\mathsf{rec}}$-valid *extensionality axiom* has the following form:

$$\forall x, y \colon \mathsf{rec}.\ (x \simeq y) \equiv (\mathsf{read}_1(x) \simeq \mathsf{read}_1(y) \wedge \cdots \wedge \mathsf{read}_n(x) \simeq \mathsf{read}_n(y)).$$

Sets also resemble arrays if a set X, $X \subseteq I$, is viewed as its *characteristic function* $X \colon I \to \{0, 1\}$ [4]. The language $\mathcal{L}_{\mathsf{set}}$ of the *theory of sets* $\mathcal{T}_{\mathsf{set}}$ has a sort elem for set elements, a sort set for sets, and the predicate symbol $\in \colon \mathsf{elem} \times \mathsf{set} \to \mathsf{prop}$. A *standard* set*-interpretation* $\mathcal{M}$ is an $\mathcal{L}_{\mathsf{set}}$-interpretation such that $\mathcal{M}[\mathsf{set}] = \mathbf{2}^{\mathcal{M}[\mathsf{elem}]}$ and $\mathcal{M}[\in](e, x) = \mathsf{true}$ if and only if $x(e) = 1$ for all $x \in \mathcal{M}[\mathsf{set}]$ and $e \in \mathcal{M}[\mathsf{elem}]$. $\mathcal{T}_{\mathsf{set}}$ is the $\mathcal{L}_{\mathsf{set}}$-theory characterized by the *class of all standard* set*-structures*. The *extensionality axiom* for $\mathcal{T}_{\mathsf{set}}$ says that two sets are equal if and only if they contain the same elements:

$$\forall x, y \colon \mathsf{set}.\ (x \simeq y) \equiv (\forall e \colon \mathsf{elem}.\ (e \in x) \equiv (e \in y)).$$

Similarly, a multiset X with elements in I is viewed as its *multiplicity function* $X \colon I \to \mathbb{N}$. The *theory of multisets* $\mathcal{T}_{\mathsf{bag}}$ requires a language $\mathcal{L}_{\mathsf{bag}}$ with sorts int^+ for the non-negative integers, elem for multiset elements, bag for multisets, and the function symbol $\mathsf{count} \colon \mathsf{elem} \times \mathsf{bag} \to \mathsf{int}^+$. A *standard* $\mathcal{L}_{\mathsf{bag}}$*-interpretation* $\mathcal{M}$ is an $\mathcal{L}_{\mathsf{bag}}$-interpretation such that: $\mathcal{M}[\mathsf{int}^+] = \mathbb{N}$; $\mathcal{M}[\mathsf{bag}] = \mathbb{N}^{\mathcal{M}[\mathsf{elem}]}$; and $\mathcal{M}[\mathsf{count}](e, x) = x(e)$ for all $e \in \mathcal{M}[\mathsf{elem}]$ and $x \in \mathcal{M}[\mathsf{bag}]$. $\mathcal{T}_{\mathsf{bag}}$ is the $\mathcal{L}_{\mathsf{bag}}$-theory characterized by the *class of all standard* bag*-structures*, with *extensionality axiom*

$$\forall x, y \colon \mathsf{bag}.\ (x \simeq y) \equiv (\forall e \colon \mathsf{elem}.\ \mathsf{count}(e, x) \simeq \mathsf{count}(e, y)).$$

The politeness of these three theories is a corollary [108] of Theorem 9.

Corollary 2. *The theories* $\mathcal{T}_{\mathsf{rec}}$, $\mathcal{T}_{\mathsf{set}}$ *and* $\mathcal{T}_{\mathsf{bag}}$ *are polite with respect to* $\{\mathsf{elem}\}$.

In order to construct finite witnesses for sets of ground literals in these theories, the ($\Leftarrow$) direction of their extensionality axiom is used to translate every inequality between terms of the data-structure sort into some constraint on their elements. For records, a rec-sorted inequality $l \not\simeq r$ is replaced with $\mathsf{read}_i(l) \not\simeq \mathsf{read}_i(r)$ for some attribute i, $1 \leq i \leq n$: the attribute is the "witness" that the records differ. For sets, an inequality $l \not\simeq r$ between terms of sort set is translated into the literal sets $\{e \in l,\ e \notin r\}$ or $\{e \notin l,\ e \in r\}$, where e is a new constant symbol of sort elem denoting an element that differentiates the sets. For multisets, a bag-sorted inequality $l \not\simeq r$ yields $\mathsf{count}(e, l) \not\simeq \mathsf{count}(e, r)$: the new constant symbol e of sort elem denotes an element that occurs with different multiplicities in the two multisets.

6.2 Recursive Data Structures

Theories of *recursive data structures* (RDS) [26,39], also known as theories of *inductive data types* [15], or *algebraic data types*, are convenient to describe several types of data structures commonly used in programming languages. Classical examples are *lists* and *trees*. These theories adopt a language $\mathcal{L}_{\mathsf{rds}}$ with a sort struct for the data structures, and a sort elem for their elements. The set of function symbols of $\mathcal{L}_{\mathsf{rds}}$ is the disjoint union of a set $\mathcal{F}_{\mathsf{c}}$ of *constructors* and a set $\mathcal{F}_{\mathsf{sel}}$ of *selectors*. A constructor symbol $c \in \mathcal{F}_{\mathsf{c}}$ has the sort $c\colon s_1, \ldots, s_n \to \mathsf{struct}$, for $s_1, \ldots, s_n \in \{\mathsf{elem}, \mathsf{struct}\}$, as a constructor takes elements and structures to build more structures. For example, in theories of lists [117,100,4,26,39,3] the constructor cons takes an element and a list, and returns the list with the given element as the head and the given list as the tail. For every constructor $c \in \mathcal{F}_{\mathsf{c}}$ of sort $c\colon s_1, \ldots, s_n \to \mathsf{struct}$, $\mathcal{F}_{\mathsf{sel}}$ contains selector symbols $\mathsf{sel}_i^c\colon \mathsf{struct} \to s_i$ for $i = 1, \ldots, n$. For example, in theories of lists the selectors associated to cons are named car and cdr: the first one applies to a cons-term to return the head; the second one returns the tail.

The axiomatizations of these theories may contain the following sets of axioms, where all variables are implicitly universally quantified:

- *Projection axioms*: $Proj = \{\mathsf{sel}_i^c(c(x_1, \ldots, x_n)) \simeq x_i \mid \mathsf{sel}_i^c \in \mathcal{F}_{\mathsf{sel}}\}$, that show how selectors operate as *projection operators* over selectors;
- *Distinctiveness axioms*: $Dis = \{c(x_1, \ldots, x_{n_c}) \not\simeq d(y_1, \ldots, y_{n_d}) \mid c, d \in \mathcal{F}_{\mathsf{c}}, c \neq d\}$, where n_c and n_d are the arities of constructors c and d, respectively: these axioms state that *distinct* constructors build *distinct* data structures, so that a term whose root symbol is a constructor cannot be equal to a term whose root symbol is a distinct constructor;
- *Acyclicity axioms*: $Acyc = \{x \not\simeq t[x] \mid t$ is an $\mathcal{F}_{\mathsf{c}}$-context$\}$, where an $\mathcal{F}_{\mathsf{c}}$-context is a context made only of symbols in $\mathcal{F}_{\mathsf{c}}$, that is, constructors; these axioms ensures that constructors do not build cyclic structures;
- *Injectivity axioms*: $Inj = \{c(x_1, \ldots, x_{n_c}) \simeq c(y_1, \ldots, y_{n_c}) \Rightarrow \bigwedge_{i=1}^{n_c} x_i \simeq y_i \mid c \in \mathcal{F}_{\mathsf{c}}\}$, where n_c is the arity of constructor c; these axioms stipulate that constructors are to be interpreted as *injective* functions.

The following general politeness theorem holds for these theories [15,118]:

Theorem 10. *For all theories $\mathcal{T}$ included in $Inj \cup Dis$, the theories $\mathcal{T} \cup Proj$ and $\mathcal{T} \cup Acyc \cup Proj$ are polite with respect to* $\{\mathsf{elem}\}$.

For all theories mentioned in Theorem 10 a *superposition-based strategy* (see Sect. 7) is a decision procedure for problems of the form $\mathcal{T} \cup Q$, where $\mathcal{T}$ is the (finite) axiomatization of the theory and Q is a (finite) set of $\mathcal{T}$-literals. If $\mathcal{T} \cup Q$ is satisfiable, the superposition-based strategy returns a set of clauses that is *saturated* (no irredundant inference applies), and contains equalities useful to construct a finite witness of Q. Dedicated tableaux-style decision procedures based on a combination of congruence closure and unification steps also exist [15].

All these theories can be extended with additional axioms defining a *bridging function* [118], such as the *length* of lists or the *size* of trees. Such a function represents a bridge between two theories: for example, a length function for lists is a bridge between a theory of lists and a theory of the integers, since the length of a list is a non-negative integer. Theories of lists and trees with bridging functions are polite theories [44]. These results rely on the characterization of the theory as a set of standard structures satisfying an extensionality axiom (cf. Sect. 6.1). For example, the *theory of possibly empty lists*, with constructors nil and cons, selectors car and cdr, extensionality axiom

$$\forall x : \mathsf{list}.\ x \not\simeq \mathsf{nil} \Rightarrow x \simeq \mathsf{cons}(\mathsf{car}(x), \mathsf{cdr}(x)),$$

and bridging function length is polite, and the bridging function helps the construction of finite witnesses [44].

7 Superposition-Based Decision Procedures

From the perspective of reasoning in a theory $\mathcal{T}$, theorem proving is the problem of $\mathcal{T}$-validity, approached refutationally by proving $\mathcal{T}$-unsatisfiability of the negation of the conjecture. A complete theorem-proving strategy for first-order logic is a *semidecision procedure* for $\mathcal{T}$-validity for all finitely axiomatized first-order theories $\mathcal{T}$: termination with a proof is guaranteed for all unsatisfiable inputs $\mathcal{T} \cup Q$, where $\mathcal{T}$ contains the axioms of the theory in clausal form, and Q is the clausal form of the negation of the conjecture. On the other hand, termination on satisfiable inputs is a challenge. In this section we survey *termination* results that allow one to apply *superposition-based theorem-proving strategies* to decide SMT problems for some theories [2,26,3,27], including most of the *polite theories* described in Section 6. The central result covered in this section is a *modularity theorem for termination* of superposition [2,3], that opened the way to understanding the relationship between superposition and equality sharing [29], and to designing integrations of SMT-solving and theorem proving [24,35,28,36], yielding more decision procedures.

A *theorem-proving strategy* is given by an *inference system*, which is a set of *inference rules*, and a *search plan*, which is an algorithm that controls the application of the inference rules. We consider a class of theorem-proving strategies known in the literature under various names:

– *Resolution-based* or *superposition-based*, to emphasize the main *expansion inference rules*,
– *Rewrite-based* or *simplification-based* or *ordering-based*, to highlight the removal of redundant clauses by *contraction inference rules* based on well-founded orderings, and
– *Completion-based* or *saturation-based*, to convey the overall process of expanding and contracting the existing set of clauses until either a contradiction arises or no more irredundant inferences apply.

In this paper we adopt the name *superposition-based*, because the surveyed results depend mostly on the inference system, and superposition plays the main role, as equality is the only shared symbol among the theories.

$$\textsf{Superposition}\quad \frac{C \vee l[s'] \bowtie r \qquad D \vee s \simeq t}{(C \vee D \vee l[t] \bowtie r)\sigma} \qquad (i), (ii), (iii), (iv)$$

$$\textsf{Reflection}\quad \frac{C \vee s' \not\simeq s}{C\sigma} \qquad \forall l \in C : (s' \not\simeq s)\sigma \not\prec l\sigma$$

$$\textsf{Equational Factoring}\quad \frac{C \vee s \simeq t \vee s' \simeq t'}{(C \vee t \not\simeq t' \vee s \simeq t')\sigma} \qquad (i), \forall l \in \{s' \simeq t'\} \cup C : (s \simeq t)\sigma \not\prec l\sigma$$

where $\bowtie$ stands for either $\simeq$ or $\not\simeq$, σ is the most general unifier (mgu) of s and s', in superposition s' is not a variable, and the following abbreviations hold:

(i) is $s\sigma \not\preceq t\sigma$,
(ii) is $\forall m \in D : (s \simeq t)\sigma \not\preceq m\sigma$,
(iii) is $l[s']\sigma \not\preceq r\sigma$, and
(iv) is $\forall m \in C : (l[s'] \bowtie r)\sigma \not\preceq m\sigma$.

$$\textsf{Simplification}\quad \frac{C[l] \qquad s \simeq t}{C[t\sigma] \qquad s \simeq t}\ l = s\sigma, \quad s\sigma \succ t\sigma, \quad C[l] \succ (s \simeq t)\sigma$$

$$\textsf{Strict Subsumption}\quad \frac{C \qquad D}{C} \qquad D \trianglerighteq C$$

$$\textsf{Deletion}\quad \frac{C \vee t \simeq t}{}$$

where $D \trianglerighteq C$ if $D \trianglerighteq C$ and $C \not\trianglerighteq D$; and $D \trianglerighteq C$ if $C\sigma \subseteq D$ (as multisets) for some substitution σ. Theorem provers also apply subsumption of variants (if $D \trianglerighteq C$ and $C \trianglerighteq D$, the oldest clause is retained) and tautology deletion (that removes clauses such as $C \vee s \simeq t \vee s \not\simeq t$).

Fig. 1. SP: a standard superposition-based inference system.

A standard superposition-based inference system, named SP from superposition (cf. Fig. 1-2, [3], Fig. 1, [27], and Fig. 1, [36]), is reported in Fig. 1:

expansion rules add what is below the single inference line to what is above; contraction rules replace what is above the double inference line by what is below. SP is parametric with respect to a *complete simplification ordering (CSO)* $\succ$ on terms, extended to literals and clauses by multiset extension. A simplification ordering is *stable* ($l \succ r$ implies $l\sigma \succ r\sigma$ for all terms l and r and substitutions σ), *monotonic* ($l \succ r$ implies $t[l] \succ t[r]$ for all terms l and r and contexts t), and has the *subterm property* (i.e., it contains the *strict subterm ordering* $\rhd$: $l \rhd r$ implies $l \succ r$ for all terms l and r). An ordering with these properties is *well-founded*. A CSO is also *total* on ground terms. Definitions, results, and references on orderings are accessible in several surveys (e.g., [54,55]).

The main expansion rule in SP is *superposition*, where $l[s'] \bowtie r$ $(C \vee l[s'] \bowtie r)$ is the literal (clause) *superposed into*, and $s \simeq t$ $(D \vee s \simeq t)$ is the literal (clause) *superposed from*, where $\bowtie$ stands for either $\simeq$ or $\not\simeq$. Depending on whether s is a variable, a constant, or a compound term, superposition is from a variable, a constant, or a compound term. *Reflection* captures ordered resolution with the reflexivity axiom $\forall x.\ x \simeq x$. *Equational factoring* allows the inference system to impose all four ordering-based restrictions listed for superposition [10]. The main contraction rule in SP is *simplification*, that performs rewriting by an equality. *Subsumption* eliminates a clause that is less general than another clause according to the subsumption ordering $\succcurlyeq$ defined in the caption of Fig. 1. *Deletion* removes clauses containing trivial equalities.

Superposition dates back to the late 1960's [112,82]; inference systems of this kind appear in many papers (e.g., [72,113,10,34]); several general treatments or surveys with additional references and historic background are available (e.g., [54,106,19,55,104,22,87,107,21]). Superposition-based strategies yield decision procedures for several fragments of first-order logic (e.g., [63,59] and [58] for a survey), and are implemented in many theorem-provers including, in alphabetical order, E [114], SPASS [124], Vampire [84], WALDMEISTER [70], and Zipperposition [48]. An *SP-derivation* is a series

$$Q_0 \underset{\text{SP}}{\vdash} Q_1 \underset{\text{SP}}{\vdash} \dots Q_i \underset{\text{SP}}{\vdash} Q_{i+1} \underset{\text{SP}}{\vdash} \dots,$$

where for all i, $i \geq 0$, the set of clauses Q_{i+1} is derived from Q_i by applying an SP-inference rule. A derivation is characterized by its *limit* $Q_\infty = \bigcup_{j \geq 0} \bigcap_{i \geq j} Q_i$, that is the set of *persistent* clauses, those that are either input or generated at some stage, and never deleted afterwards. A derivation is a *refutation* if there exists an i such that $\square \in Q_i$, where $\square$ is the empty clause, the contradiction in clausal form. SP is *refutationally complete*: whenever the input set Q_0 is unsatisfiable, then there exist SP-refutations from Q_0. Inference systems are nondeterministic: multiple SP-derivations are possible from a given input set Q_0. The pairing of SP with a search plan yields an *SP-strategy*, and the SP-derivation generated from Q_0 by an SP-strategy is unique.

Refutational completeness of the inference system is not sufficient for the completeness of a strategy: the complementary requirement on the search plan is *fairness*. A derivation is *fair* if it is guaranteed to be a refutation whenever the input set is unsatisfiable. A search plan is *fair* if it generates a fair derivation for

all inputs. A strategy is *fair* if its search plan is. A strategy is *complete* if its inference system is refutationally complete and its search plan is fair. In practice, a derivation that considers eventually all irredundant inferences is fair, and its limit is *saturated*. An inference is *redundant* if it uses or generates redundant clauses, and *irredundant* otherwise. Definitions of redundancy and sufficient conditions for fairness can be given based on well-founded orderings on either clauses [10] or proofs [34,22]. In the sequel SP-strategy stands for *complete SP-strategy*.

In order to prove that an SP-strategy is a *decision procedure* for a certain class of problems, one needs to show that it is guaranteed to *terminate* on inputs in that class. If one shows that SP only generates *finitely many* clauses from such inputs, termination is guaranteed. We begin with $\exists$-*decidability*, that is, we consider $\mathcal{T}$-satisfiability problems $\mathcal{T} \cup Q$, where $\mathcal{T}$ contains the axioms of the theory in clausal form and Q is a set of ground $\mathcal{T}$-literals. SP generates finitely many clauses from such problems in the theories of:

- *Equality*, for which $\mathcal{T}$ is empty [86,4,11],
- *Non-empty possibly cyclic lists* [4] and *possibly empty possibly cyclic lists* [3],
- *Arrays* with or without extensionality [4,2,3],
- Finite *sets* with or without extensionality [4],
- *Records* with or without extensionality [2,3],
- *Integer offsets* and *integer offsets modulo*, a theory useful to describe data structures such as *circular queues* [2,3], and
- *Recursive data structures* with a constructor and k selectors [26], of which integer offsets and *acyclic non-empty lists* are special cases for k=1 and k=2, respectively.

Therefore, these theories are $\exists$-*SP-decidable* [27], meaning that an SP-strategy is a *decision procedure* for $\exists$-decidability in these theories.

For each theory $\mathcal{T}$ the proof of termination rests on an analysis of the possible SP-inferences from an input of the form $\mathcal{T} \cup Q$, showing that there are only finitely many. This analysis assumes that a preprocessing phase *flattens* all literals in Q, by introducing new constant symbols and equalities, in such a way that every positive literal contains at most one occurrence of function symbol, and every negative literal contains no occurrence of function symbols. For example, the literal $f(a) \not\simeq f(b)$ yields the set of flat literals $\{f(a) \simeq a',\ f(b) \simeq b',\ a' \not\simeq b'\}$ by introducing fresh constants a' and b'.

The preprocessing phase may involve some other simple mechanical transformation, called $\mathcal{T}$-*reduction*: for example, for the theories of *arrays with extensionality* [4,2,3] and *records with extensionality* [2,3], $\mathcal{T}$-reduction replaces array-sorted, and rec-sorted, inequalities, via the introduction of "witnesses," as already described in Sect. 6.1, so that the extensionality axiom can be removed. For both theories, $\mathcal{T}$-reduction preserves equisatisfiability also in the presence of free function symbols, which is relevant for their union with the theory of equality, provided the sorts array and rec do not appear in the sorts of the free function symbols [3]. Since the presentation of recursive data structures includes *infinitely many acyclicity axioms* (cf. Sect. 6.2), $\mathcal{T}$-reduction in this case transforms the problem to an equisatisfiable problem with finitely many acyclicity axioms [26,3].

For some theories, the $\mathcal{T}$-reduction is empty and preprocessing consists only of flattening. Also, the CSO $\succ$ employed by SP is required to be *good*, meaning that $t \succ c$ for all ground compound terms t and constants c [3,25,27].

Once termination is established, the *complexity* of the resulting superposition-based decision procedure may be characterized abstractly in terms of *meta-saturation* [88,91]. More concretely, the superposition-based decision procedures for the above mentioned theories are exponential, except for *records without extensionality* and *integer offsets modulo* [26,3]. For the theory of arrays this is unavoidable, because the already mentioned case analysis over whether two indices are equal (see Sect. 6.1) means that $\mathcal{T}_{\mathsf{array}}$-satisfiability is as hard as SAT, and therefore has an exponential lower bound. For the theories of *records with extensionality* and *integer offsets* the superposition-based decision procedures were improved to be *polynomial* [27], showing that there is a big difference between records and arrays complexity-wise.

The *modularity problem* for $\exists$-SP-decidability is the problem of showing that if $\mathcal{T}_1$ and $\mathcal{T}_2$ are $\exists$-SP-decidable, then $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$ also is $\exists$-SP-decidable. Since $\mathcal{T}_i$-reduction applies separately for each theory, and flattening is harmless, the modularity of $\exists$-SP-decidability reduces to that of *termination*. The problem is to show that if SP is guaranteed to generate finitely many clauses from inputs of the form $\mathcal{T}_i \cup Q_i$ $(i = 1, 2)$, where Q_i is a set of ground $\mathcal{T}_i$-literals, then SP is guaranteed to generate finitely many clauses from inputs of the form $\mathcal{T} \cup Q$, where Q is a set of ground $\mathcal{T}$-literals. The issue is to find sufficient conditions for this result. Two conditions are easy: $\mathcal{T}_1$ and $\mathcal{T}_2$ should not share non-nullary function symbols, which is implied by their being disjoint, and the CSO $\succ$ should be good for both theories. The key condition is that $\mathcal{T}_1$ and $\mathcal{T}_2$ are *variable-inactive* [2,3], which prevents *superposition from variables* across theories.

Definition 10. *A clause φ is* variable-inactive *if no $\succ$-maximal literal in φ is an equality $t \simeq x$ where $x \notin Var(t)$. A set of clauses is* variable-inactive *if all its clauses are.*

Variable-inactivity is concerned only with equalities $t \simeq x$ where $x \notin Var(t)$, because if $x \in Var(t)$, the ordering-based restrictions on superposition suffice to bar superposition from x, as $t \succ x$ in any ordering $\succ$ with the subterm property.

Definition 11. *A theory $\mathcal{T}$ is* variable-inactive*, if for all $\mathcal{T}$-satisfiability problems $\mathcal{T} \cup Q$ the limit of every fair SP-derivation from $\mathcal{T} \cup Q$ is variable-inactive.*

The absence of shared non-nullary function symbols prevents superposition from compound terms across theories. Thus, the only superpositions across theories are superpositions from constants into constants, and because there are finitely many constant symbols in the problem, the modularity of termination follows (cf. Theorem 5, [2]; Theorem 4.1 and Corollary 3, [3]).

Theorem 11. *If theories $\mathcal{T}_1$ and $\mathcal{T}_2$ are disjoint, variable-inactive, and $\exists$-SP-decidable, then their union $\mathcal{T}_1 \cup \mathcal{T}_2$ is $\exists$-SP-decidable.*

All the theories above satisfy the hypotheses of this theorem and therefore their unions are $\exists$-SP-decidable (cf. Corollary 1, [2] and Theorem 4.6, [3]).

Superpositions from constants into constants across theories are superpositions from shared constants into shared constants. Also, the proof of the modularity result (see the proof of Theorem 4.1, [3]) shows that the equalities superposed from are equalities between constants. Thus, the only equalities that are active across variable-inactive theories in superposition are *equalities between shared constants*. This suggests an analogy with equality sharing, where the decision procedures can build an arrangement for stably infinite theories by propagating only clauses made of *equalities between shared constants* (e.g., Sect. 10.3, [39]). Theorem-proving strategies do not require stable infiniteness upfront. However, *variable-inactivity implies stable-infiniteness*, a finding that reinforces the analogy between superposition and equality sharing, and the intuition that they capture the same essential features of reasoning in a union of theories. The discovery that variable-inactivity implies stable-infiniteness descends from a lemma showing that superposition has the power of revealing the lack of infinite models by generating at-most cardinality constraints (cf. Lemma 5.2, [29]).

Lemma 1. *A finite satisfiable set of clauses Q admits no infinite models if and only if the limit of every fair SP-derivation from Q contains an at-most cardinality constraint.*

Since an at-most cardinality constraint is *not* variable-inactive, the result that variable-inactivity implies stable-infiniteness follows (cf. Theorem 4.5, [3]).

Theorem 12. *If a theory $\mathcal{T}$ is variable-inactive, then it is stably infinite.*

Indeed, if $\mathcal{T}$ is not stably-infinite, there is a quantifier-free $\mathcal{T}$-satisfiable $\mathcal{T}$-formula φ with no infinite $\mathcal{T}$-model. By the above lemma, the limit of every fair SP-derivation from the clausal form of $\mathcal{T} \cup \{\varphi\}$ contains an at-most cardinality constraint, and $\mathcal{T}$ is not variable-inactive.

We consider next $\mathcal{T}$-satisfiability problems $\mathcal{T} \cup Q$ where $\mathcal{T}$ contains the axioms of the theory in clausal form and Q is a set of ground $\mathcal{T}$-clauses. If SP is guaranteed to generate finitely many clauses from these problems, an SP-strategy is a decision procedure for the $\mathcal{T}$-satisfiability of quantifier-free formulas and theory $\mathcal{T}$ is *SP-decidable*. Results of this kind are obtained by replacing variable inactivity with a stronger property named *subterm inactivity*: the theories of *equality*, *arrays with or without extensionality*, possibly augmented with an *injectivity* predicate, a *swap* predicate, or both, *finite sets with or without extensionality*, *recursive data structures*, and their unions, are shown to be SP-decidable in this manner [25,23]. Other variable-inactive theories, such as *possibly empty lists*, *records*, and *integer offsets modulo*, are not subterm-inactive. By a simpler approach [27] it is possible to show that variable-inactivity alone suffices for SP-decidability (cf. Theorem 3.5, [27]).

Theorem 13. *If a theory $\mathcal{T}$ is variable-inactive and $\exists$-SP-decidable, then it is SP-decidable.*

Furthermore, for *arrays with or without extensionality*, *records with or without extensionality*, *integer offsets*, and their unions, superposition can be a preprocessor for an SMT-solver: the problem is decomposed in such a way that superposition is applied only to the axiomatization $\mathcal{T}$ and ground unit $\mathcal{T}$-clauses, realizing an inference-based reduction of $\mathcal{T}$ to the theory of equality [24,28].

Although the results surveyed in this section were obtained for languages where equality is the only predicate, it is simple to generalize them to languages with more predicate symbols (see [36], Section 3). The discovery that variable-inactivity implies stable infiniteness is rich in implications. Variable-inactivity and meta-saturation [88,91] were used to test for stable infiniteness [81,90]. A superposition-based decision procedure for a variable-inactive theory can be a component of an equality-sharing combination: this is a theoretical underpinning for a method named DPLL($\Gamma + \mathcal{T}$) [35,36] which integrates a superposition-based inference system Γ into the DPLL($\mathcal{T}$) framework for SMT [103,13,85]. DPLL(Γ+$\mathcal{T}$) handles axiomatized theories by superposition and built-in theories by DPLL($\mathcal{T}$). Superposition offers complete reasoning about quantifiers, decision procedures for some axiomatized theories, and it can detect the lack of infinite models. DPLL(Γ+$\mathcal{T}$) also enriches DPLL($\mathcal{T}$) with *speculative inferences* to yield decision procedures for more theories and unions of theories.

8 CDSAT: An Overview

The philosophy of equality sharing is to combine decision procedures as *black-boxes*. Clearly, black-box combination has advantages: existing procedures can be combined without modifying them, and their communication can be minimized. In equality sharing, communication is limited to the propagation of disjunctions of equalities between shared constants (e.g., Sect. 10.3, [39]). The DPLL($\mathcal{T}$) framework for SMT [103,13,85] extends this philosophy to the interaction between the *conflict-driven clause learning* (CDCL) procedure for SAT-solving [49,93,94] and the equality-sharing-based $\mathcal{T}$-decision procedure, where $\mathcal{T}$ is a union of theories. An *abstraction function* maps $\mathcal{T}$-atoms to propositional atoms and its inverse performs the opposite translation. Every disjunction propagated by the $\mathcal{T}$-decision procedure is handled by the CDCL procedure, which searches for a propositional model of the set of clauses. The current candidate model is represented as an assignment, called *trail*:

$$\Gamma = u_1 \leftarrow \mathfrak{b}_1, \ldots, u_m \leftarrow \mathfrak{b}_m,$$

where, for all i, $1 \leq i \leq m$, u_i is a propositional atom and $\mathfrak{b}_i$ is either true or false. The $\mathcal{T}$-decision procedure contributes by signalling that a subset of these Boolean assignments implies in $\mathcal{T}$ either a contradiction ($\mathcal{T}$-*conflict*), or another Boolean assignment to an existing $\mathcal{T}$-atom, which is thus added to the trail ($\mathcal{T}$-*propagation*).

Some decision procedures for fragments of arithmetic are *conflict-driven theory procedures*, in the sense that they generalize features of CDCL to theory reasoning [126,123,96,83,47,78,79,69,80]. They assign values to first-order variables,

like CDCL assign truth values to atoms, and they *explain* theory conflicts by theory inferences, like CDCL explains Boolean conflicts by resolution. A significant difference is that propositional resolution generates resolvents made of input atoms, whereas theory inferences may generate *new* (i.e., non-input) $\mathcal{T}$-atoms. If such a conflict-driven procedure for a single theory is integrated as a black-box, the search for a $\mathcal{T}$-model cannot take direct advantage of its guesses and inferences, and the conflict-driven reasoning remains propositional. The MCSAT method, where MCSAT stands for *Model-Constructing SATisfiability*, shows how to integrate a conflict-driven procedure for one theory [53,127,75,68], or for a specific union of theories [77,18], with CDCL, allowing them to cooperate on a single trail that contains assignments to both Boolean and first-order variables. CDSAT, which stands for *Conflict-Driven SATisfiability* [31,32,33], generalizes MCSAT to *generic unions of disjoint theories*, and generalizes equality sharing to accommodate *both* black-box and conflict-driven theory procedures.

The idea of CDSAT is to open the boxes and let *theory modules*, one for each theory in the union $\mathcal{T}$, cooperate in the search for a $\mathcal{T}$-model. Propositional logic is considered as one of the theories, and the CDCL procedure is its theory module. All theory modules access the same trail

$$\Gamma = u_1 \leftarrow \mathsf{c}_1, \ldots, u_m \leftarrow \mathsf{c}_m,$$

where, for all i, $1 \leq i \leq m$, u_i is a $\mathcal{T}$-term and c_i is a concrete value of the appropriate sort for u_i. If u_i is a Boolean term, that is, a formula, c_i is either true or false. If u_i is a term of sort int, for example, c_i is an integer value, as in $x \leftarrow 3$ or $(y+1) \leftarrow -3$. The notion of *theory extension* is used to make sure that values can be named with fresh constant symbols, which, however, remain separate from the original language and do not occur in terms.

Once first-order (i.e., non-Boolean) assignments are allowed on the trail, there is no reason for barring them from appearing in the input problem, which is also viewed as an assignment. Therefore, CDSAT solves a generalization of SMT dubbed SMA for *satisfiability modulo assignments*. An SMT problem is written as $\{u_1 \leftarrow \mathsf{true}, \ldots, u_m \leftarrow \mathsf{true}\}$, where, for all i, $1 \leq i \leq m$, u_i is a quantifier-free $\mathcal{T}$-formula. An SMA problem is written as $\{u_1 \leftarrow \mathsf{true}, \ldots, u_m \leftarrow \mathsf{true}, u_{m+1} \leftarrow \mathsf{c}_1, \ldots, u_{m+j} \leftarrow \mathsf{c}_j\}$, where $\{u_1 \leftarrow \mathsf{true}, \ldots, u_m \leftarrow \mathsf{true}\}$ is an SMT problem, and, for all i, $m+1 \leq i \leq m+j$, u_i is a first-order term, typically a variable occurring in some of the input formulas. Either way, the trail is initialized with the input problem, and CDSAT works to determine whether it is $\mathcal{T}$-satisfiable.

Since the assignment on the trail mixes symbols and values from the different theories, each theory has its *view* of the trail. Suppose that $\mathcal{T}$ is the union of theories $\mathcal{T}_1, \ldots, \mathcal{T}_n$. The $\mathcal{T}_k$-view $(1 \leq k \leq n)$ includes the pairs $t \leftarrow \mathsf{c}$, where c comes from the extension of $\mathcal{T}_k$, and those equalities and disequalities determined by first-order assignments to terms of a $\mathcal{T}_k$-sort. For example, if the trail contains $\{x \leftarrow 3, y \leftarrow 3, z \leftarrow 2\}$, the theory view of every theory with sort int contains $x \simeq y$, $x \not\simeq z$, and $y \not\simeq z$. Indeed, in the presence of first-order assignments, the truth value of an equality can be determined in two ways: either by assigning it true or false, or by assigning the same or different values to its sides. CDSAT

employs a notion of *relevance* of a term to a theory to determine which theory uses one way or the other. A $\mathcal{T}_k$-model $\mathcal{M}_k$ *satisfies* an assignment if for all pairs $t \leftarrow \mathfrak{c}$ in the $\mathcal{T}_k$-view of the assignment $\mathcal{M}_k$ interprets t and $\mathfrak{c}$ as the same element. An assignment is *$\mathcal{T}$-satisfiable* if there is a $\mathcal{T}$-model that endorses the $\mathcal{T}$-view, or *global view*, which contains everything. Otherwise, the assignment is *$\mathcal{T}$-unsatisfiable*. A $\mathcal{T}$-unsatisfiable subset of the trail represents a *conflict*.

Since the conflict-driven search is provided centrally for all theories by CDSAT, theory modules are *theory inference systems*. Thus, the reasoning in the union of the theories is conflict-driven, and combination of theories becomes *conflict-driven combination of theory inference systems*. A theory decision procedure that is *not* conflict-driven is still incorporated as a black-box, by viewing it as an inference system whose only inference rule invokes the procedure to detect the $\mathcal{T}_k$-unsatisfiability of the $\mathcal{T}_k$-view of the trail.

CDSAT is defined as a *transition system* with *trail rules* and *conflict-state rules*. The trail rules transform the trail with *decisions* or *deductions*, and detect conflicts. A *decision* is a guess of a value for a term: a $\mathcal{T}_k$-module is allowed to decide a value for a term t that is relevant to $\mathcal{T}_k$. CDSAT has a notion of *acceptable* assignment to exclude decisions that are obviously bad, because repetitious or causing trivial conflicts. With a *deduction*, a $\mathcal{T}_k$-module posts on the trail a Boolean assignment inferred from assignments on the trail. As a deduction may bring to the trail a *new* term, all deduced terms must come from a *finite global basis* to avoid jeopardizing termination. The inferred assignment is a *justified assignment*, whose *justification* is the set of premises from which it was inferred. This mechanism encompasses both *$\mathcal{T}_k$-propagations* and *explanations* of $\mathcal{T}_k$-conflicts. All assignments on the trail that are not decisions are justified assignments, including input assignments, that have empty justifications.

The *conflict-state rules* intervene after a conflict has been detected, so that they work on the trail and the conflict, until either the conflict is solved, or the input problem is recognized as $\mathcal{T}$-unsatisfiable. CDSAT applies a form of *resolution* to unfold the conflict, by replacing a justified assignment in the conflict with its justification. This process continues until CDSAT identifies either a first-order decision that needs to be undone, or a Boolean assignment that needs to be flipped: the flipped assignment is also a justified assignment inheriting its justification from the process of unfolding the conflict.

CDSAT is *sound*, *terminating*, and *complete*, under suitable hypotheses on $\mathcal{T}_k$-modules and global basis [31,33]. CDSAT requires the $\mathcal{T}_k$'s to be disjoint, but *not stably infinite*, provided there is a *leading theory* that knows all sorts in $\mathcal{T}$. For completeness, every $\mathcal{T}_k$-module must be *leading-theory-complete*, which ensures that when no trail rule applies to the trail, there is a $\mathcal{T}_k$-model that satisfies the $\mathcal{T}_k$-view of the trail, and agrees with a model of the leading theory on *arrangement of shared terms* and *cardinalities of shared sorts*. Whenever CDSAT terminates without reporting unsatsfiability, the $\mathcal{T}_k$-models can be combined in a $\mathcal{T}$-model satisfying the trail, hence the input assignment. CDSAT is a nondeterministic system, as there is nondeterminism in the CDSAT transition system and in each theory module. A CDSAT procedure is obtained by adding a

search plan, that establishes priorities among CDSAT transition rules, theories, and inference rules within each theory module.

9 Discussion

Reasoning in a union of theories can be approached in several ways: the equality-sharing method and its extensions combine theory decision procedures; theorem-proving strategies unite theory presentations and reason about them; and CDSAT combines in a conflict-driven manner theory inference systems. With respect to lifting stable infiniteness, extensions of equality sharing based on shiny, gentle, or polite theories are asymmetric; the superposition-based methodology is symmetric, as it treats all theories evenly, and it handles cardinality issues seamlessly. CDSAT is also asymmetric, as the leading theory knows more than the other theories, including the cardinalities of the shared sorts. Another way to go beyond equality sharing is to admit combinations of *non-disjoint* theories (e.g., [65,67,101,125,118]). Work on this direction has begun for methods based on gentleness and politeness [43,44], as well as for superposition-based decision procedures [111], while it is a direction of future work for CDSAT.

Acknowledgments The authors thank the co-authors of their papers covered in this survey. Part of this work was done when the first author was visiting LORIA Nancy and the Computer Science Laboratory of SRI International: the support of both institutions is greatly appreciated. This work was funded in part by grant "Ricerca di base 2017" of the Università degli Studi di Verona.

References

1. M. Armand, G. Faure, B. Grégoire, C. Keller, L. Théry, and B. Werner. A modular integration of SAT/SMT solvers to Coq through proof witnesses. In J.-P. Jouannaud and Z. Shao, editors, *Proc. of CPP-1*, pages 135–150. Springer, 2011.
2. A. Armando, M. P. Bonacina, S. Ranise, and S. Schulz. On a rewriting approach to satisfiability procedures: extension, combination of theories and an experimental appraisal. In B. Gramlich, editor, *Proc. of FroCoS-5*, volume 3717 of *LNAI*, pages 65–80. Springer, 2005.
3. A. Armando, M. P. Bonacina, S. Ranise, and S. Schulz. New results on rewrite-based satisfiability procedures. *ACM TOCL*, 10(1):129–179, 2009.
4. A. Armando, S. Ranise, and M. Rusinowitch. A rewriting approach to satisfiability procedures. *Inform. Comput.*, 183(2):140–164, 2003.
5. F. Baader and S. Ghilardi. Connecting many-sorted structures and theories through adjoint functions. In B. Gramlich, editor, *Proc. of FroCoS-5*, volume 3717 of *LNAI*, pages 31–47. Springer, 2005.
6. F. Baader and K. U. Schulz. Combination techniques and decision problems for disunification. *Theor. Comput. Sci.*, 142(2):229–255, 1995.
7. F. Baader and K. U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *J. Symb. Comput.*, 21(2):211–243, 1996.

8. F. Baader and K. U. Schulz. Combination of constraint solvers for free and quasi-free structures. *Theor. Comput. Sci.*, 192(1):107–161, 1998.
9. F. Baader and C. Tinelli. Deciding the word problem in the union of equational theories. *Inform. Comput.*, 178(2):346–390, 2002.
10. L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. Logic and Comput.*, 4(3):217–247, 1994.
11. L. Bachmair, A. Tiwari, and L. Vigneron. Abstract congruence closure. *J. Automat. Reason.*, 31(2):129–168, 2003.
12. C. W. Barrett, C. L. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli. CVC4. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. of CAV-23*, volume 6806 of *LNCS*, pages 171–177. Springer, 2011.
13. C. W. Barrett, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Splitting on demand in SAT modulo theories. In M. Hermann and A. Voronkov, editors, *Proc. of LPAR-13*, volume 4246 of *LNAI*, pages 512–526. Springer, 2006.
14. C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability modulo theories. In A. Biere, M. Heule, H. V. Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, chapter 26, pages 825–886. IOS Press, 2009.
15. C. W. Barrett, I. Shikanian, and C. Tinelli. An abstract decision procedure for satisfiability in the theory of inductive data types. *J. Satisfiability Bool. Model. and Comput.*, 3:21–46, 2007.
16. C. W. Barrett and C. Tinelli. Satisfiability modulo theories. In E. Clarke, H. V. Thomas Henzinger, and R. Bloem, editors, *Handbook of Model Checking*. Springer, 2018.
17. J. C. Blanchette, S. Böhme, and L. C. Paulson. Extending Sledgehammer with SMT solvers. In N. Bjørner and V. Sofronie-Stokkermans, editors, *Proc. of CADE-23*, volume 6803 of *LNAI*, pages 116–130. Springer, 2011.
18. F. Bobot, S. Graham-Lengrand, B. Marre, and G. Bury. Centralizing equality reasoning in MCSAT. Proc of SMT-16, 2018.
19. M. P. Bonacina. A taxonomy of theorem-proving strategies. In M. J. Wooldridge and M. Veloso, editors, *Artificial Intelligence Today – Recent Trends and Developments*, volume 1600 of *LNAI*, pages 43–84. Springer, 1999.
20. M. P. Bonacina. On theorem proving for program checking – Historical perspective and recent developments. In M. Fernàndez, editor, *Proc. of PPDP-12*, pages 1–11. ACM Press, 2010.
21. M. P. Bonacina. Parallel theorem proving. In Y. Hamadi and L. Sais, editors, *Handbook of Parallel Constraint Reasoning*, chapter 6, pages 179–235. Springer, 2018.
22. M. P. Bonacina and N. Dershowitz. Abstract canonical inference. *ACM TOCL*, 8(1):180–208, 2007.
23. M. P. Bonacina and M. Echenim. Generic theorem proving for decision procedures. TR 41/2006, Univ. degli Studi di Verona, 2006. Revised March 2007; available at http://profs.sci.univr.it/~bonacina/rewsat.html.
24. M. P. Bonacina and M. Echenim. $\mathcal{T}$-decision by decomposition. In F. Pfenning, editor, *Proc. of CADE-21*, volume 4603 of *LNAI*, pages 199–214. Springer, 2007.
25. M. P. Bonacina and M. Echenim. Rewrite-based decision procedures. In M. Archer, T. Boy de la Tour, and C. Muñoz, editors, *Proc. of STRATEGIES-6*, volume 174(11) of *ENTCS*, pages 27–45. Elsevier, 2007.
26. M. P. Bonacina and M. Echenim. Rewrite-based satisfiability procedures for recursive data structures. In B. Cook and R. Sebastiani, editors, *Proc. of PDPAR-4*, volume 174(8) of *ENTCS*, pages 55–70. Elsevier, 2007.

27. M. P. Bonacina and M. Echenim. On variable-inactivity and polynomial $\mathcal{T}$-satisfiability procedures. *J. Logic and Comput.*, 18(1):77–96, 2008.
28. M. P. Bonacina and M. Echenim. Theory decision by decomposition. *J. Symb. Comput.*, 45(2):229–260, 2010.
29. M. P. Bonacina, S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Decidability and undecidability results for Nelson-Oppen and rewrite-based decision procedures. In U. Furbach and N. Shankar, editors, *Proc. of IJCAR-3*, volume 4130 of *LNAI*, pages 513–527. Springer, 2006.
30. M. P. Bonacina, S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Decidability and undecidability results for Nelson-Oppen and rewrite-based decision procedures. TR 308-06, Univ. degli Studi di Milano, 2006. Available at http://profs.sci.univr.it/~bonacina/rewsat.html.
31. M. P. Bonacina, S. Graham-Lengrand, and N. Shankar. Satisfiability modulo theories and assignments. In L. de Moura, editor, *Proc. of CADE-26*, volume 10395 of *LNAI*, pages 42–59. Springer, 2017.
32. M. P. Bonacina, S. Graham-Lengrand, and N. Shankar. Proofs in conflict-driven theory combination. In J. Andronick and A. Felty, editors, *Proc. of CPP-7*, pages 186–200. ACM Press, 2018.
33. M. P. Bonacina, S. Graham-Lengrand, and N. Shankar. Conflict-driven satisfiability for theory combination: Transition system and completeness. *J. Automat. Reason.*, In press:1–31, 2019. Available at http://doi.org/10.1007/s10817-018-09510-y.
34. M. P. Bonacina and J. Hsiang. Towards a foundation of completion procedures as semidecision procedures. *Theor. Comput. Sci.*, 146:199–242, 1995.
35. M. P. Bonacina, C. A. Lynch, and L. de Moura. On deciding satisfiability by DPLL($\Gamma + \mathcal{T}$) and unsound theorem proving. In R. A. Schmidt, editor, *Proc. of CADE-22*, volume 5663 of *LNAI*, pages 35–50. Springer, 2009.
36. M. P. Bonacina, C. A. Lynch, and L. de Moura. On deciding satisfiability by theorem proving with speculative inferences. *J. Automat. Reason.*, 47(2):161–189, 2011.
37. T. Bouton, D. Caminha B. de Oliveira, D. Déharbe, and P. Fontaine. veriT: an open, trustable and efficient SMT-solver. In R. A. Schmidt, editor, *Proc. of CADE-22*, volume 5663 of *LNAI*, pages 151–156. Springer, 2009.
38. M. Bozzano, R. Bruttomesso, A. Cimatti, T. Junttila, S. Ranise, R. Sebastiani, and P. van Rossu. Efficient satisfiability modulo theories via delayed theory combination. In K. Etessami and S. Rajamani, editors, *Proc. of CAV-17*, volume 3576 of *LNCS*, pages 335–349. Springer, 2005.
39. A. R. Bradley and Z. Manna. *The Calculus of Computation - Decision Procedures with Applications to Verification.* Springer, 2007.
40. A. R. Bradley, Z. Manna, and H. B. Sipma. What's decidable about arrays? In E. A. Emerson and K. S. Namjoshi, editors, *Proc. of VMCAI-7*, volume 3055 of *LNCS*, pages 427–442. Springer, 2006.
41. R. Brummayer and A. Biere. Boolector: An efficient SMT solver for bit-vectors and arrays. In S. Kowalewski and A. Philippou, editors, *Proc. of TACAS-15*, volume 5505 of *LNCS*, pages 174–177. Springer, 2009.
42. R. Bruttomesso, E. Pek, N. Sharygina, and A. Tsitovich. The OpenSMT solver. In J. Esparza and R. Majumdar, editors, *Proc. of TACAS-16*, volume 6015 of *LNCS*, pages 150–153. Springer, 2010.
43. P. Chocron, P. Fontaine, and C. Ringeissen. A gentle non-disjoint combination of satisfiability procedures. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Proc. of IJCAR-7*, volume 8562 of *LNAI*, pages 122–136. Springer, 2014.

44. P. Chocron, P. Fontaine, and C. Ringeissen. Politeness and combination methods for theories with bridging functions. *J. Automat. Reason.*, 2019. Available at https://doi.org/10.1007/s10817-019-09512-4.
45. A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani. The MathSAT5 SMT solver. In N. Piterman and S. Smolka, editors, *Proc. of TACAS-19*, volume 7795 of *LNCS*, pages 93–107. Springer, 2013.
46. S. Conchon, E. Contejean, and M. Iguernelala. Canonized rewriting and ground AC completion modulo Shostak theories : Design and implementation. *Logical Methods in Computer Science*, 8(3):1–29, 2012.
47. S. Cotton. Natural domain SMT: A preliminary assessment. In K. Chatterjee and T. A. Henzinger, editors, *Proc. of FORMATS-8*, volume 6246 of *LNCS*, pages 77–91. Springer, 2010.
48. S. Cruanes. *Extending superposition with integer arithmetic, structural induction, and beyond*. PhD thesis, École Polytechnique, Univ. Paris-Saclay, 2015.
49. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.
50. L. de Moura and N. Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and J. Rehof, editors, *Proc. of TACAS-14*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
51. L. de Moura and N. Bjørner. Bugs, moles and skeletons: symbolic reasoning for software development. In J. Giesl and R. Hähnle, editors, *Proc. of IJCAR-5*, volume 6173 of *LNAI*, pages 400–411. Springer, 2010.
52. L. de Moura and N. Bjørner. Satisfiability modulo theories: introduction and applications. *Commun. ACM*, 54(9):69–77, 2011.
53. L. de Moura and D. Jovanović. A model-constructing satisfiability calculus. In R. Giacobazzi, J. Berdine, and I. Mastroeni, editors, *Proc. of VMCAI-14*, volume 7737 of *LNCS*, pages 1–12. Springer, 2013.
54. N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 243–320. Elsevier, 1990.
55. N. Dershowitz and D. A. Plaisted. Rewriting. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 9, pages 535–610. Elsevier, 2001.
56. D. L. Detlefs, G. Nelson, and J. B. Saxe. Simplify: a theorem prover for program checking. *J. ACM*, 52(3):365–473, 2005.
57. B. Dutertre. Yices 2.2. In A. Biere and R. Bloem, editors, *Proc. of CAV-26*, volume 8559 of *LNCS*, pages 737–744. Springer, 2014.
58. C. Fermüller, A. Leitsch, U. Hustadt, and T. Tammet. Resolution decision procedures. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 25, pages 1793–1849. Elsevier, 2001.
59. A. Fietzke and C. Weidenbach. Superposition as a decision procedure for timed automata. *Mathematics in Computer Science*, 6(4):409–425, 2012.
60. P. Fontaine. Combinations of theories for decidable fragments of first-order logic. In S. Ghilardi and R. Sebastiani, editors, *Proc. of FroCoS-7*, volume 5749 of *LNAI*, pages 263–278. Springer, 2009.
61. P. Fontaine and E. P. Gribomont. Combining non-stably infinite, non-first order theories. In W. Ahrendt, P. Baumgartner, H. de Nivelle, S. Ranise, and C. Tinelli, editors, *Proc. of PDPAR-2*, volume 125 of *ENTCS*, pages 37–51. Elsevier, 2005.
62. J. H. Gallier and W. Snyder. Designing unification procedures using transformations: a survey. *Bulletin of the EATCS*, 40:273–326, 1990.

63. H. Ganzinger and H. de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *Proc. of LICS-14*. IEEE Computer Society, 1999.
64. H. Ganzinger, H. Rueß, and N. Shankar. Modularity and refinement in inference systems. Technical Report CSL-SRI-04-02, SRI International, 2004.
65. S. Ghilardi, E. Nicolini, and D. Zucchelli. A comprehensive framework for combining decision procedures. In B. Gramlich, editor, *Proc. of FroCoS-5*, volume 3717 of *LNAI*, pages 1–30. Springer, 2005.
66. S. Ghilardi, E. Nicolini, and D. Zucchelli. Recent advances in combined decision problems. In E. Ballo and M. Franchella, editors, *Logic and Philosophy in Italy: Trends and Perspectives*, pages 87–104. Polimetrica, 2006.
67. S. Ghilardi, E. Nicolini, and D. Zucchelli. A comprehensive combination framework. *ACM TOCL*, 9(2):1–54, 2008.
68. S. Graham-Lengrand and D. Jovanović. An MCSAT treatment of bit-vectors. In M. Brain and L. Hadarean, editors, *Proc. of SMT-15*, 2017.
69. L. Haller, A. Griggio, M. Brain, and D. Kroening. Deciding floating-point logic with systematic abstraction. In G. Cabodi and S. Singh, editors, *Proc. of FMCAD-12*. ACM and IEEE, 2012.
70. T. Hillenbrand. Citius, altius, fortius: lessons learned from the theorem prover Waldmeister. In I. Dahn and L. Vigneron, editors, *Proc. of FTP-4*, volume 86 of *ENTCS*. Elsevier, 2003.
71. J. Hsiang and M. Rusinowitch. On word problems in equational theories. In T. Ottman, editor, *Proc. of ICALP-14*, volume 267 of *LNCS*, pages 54–71. Springer, 1987.
72. J. Hsiang and M. Rusinowitch. Proving refutational completeness of theorem proving strategies: the transfinite semantic tree method. *J. ACM*, 38(3):559–587, 1991.
73. G. Huet. A complete proof of correctness of the Knuth–Bendix completion algorithm. *J. Comput. Syst. Sci.*, 23(1):11–21, 1981.
74. J.-P. Jouannaud and H. Kirchner. Completion of a set of rules modulo a set of equations. *SIAM J. Comput.*, 15(4):1155–1194, 1986.
75. D. Jovanović. Solving nonlinear integer arithmetic with MCSAT. In A. Bouajjani and D. Monniaux, editors, *Proc. of VMCAI-18*, volume 10145 of *LNCS*, pages 330–346. Springer, 2017.
76. D. Jovanović and C. W. Barrett. Polite theories revisited. In C. G. Fermüller and A. Voronkov, editors, *Proc. of LPAR-17*, volume 6397 of *LNAI*, pages 402–41. Springer, 2010.
77. D. Jovanović, C. W. Barrett, and L. de Moura. The design and implementation of the model-constructing satisfiability calculus. In B. Jobstman and S. Ray, editors, *Proc. of FMCAD-13*. ACM and IEEE, 2013.
78. D. Jovanović and L. de Moura. Cutting to the chase: solving linear integer arithmetic. In N. Bjørner and V. Sofronie-Stokkermans, editors, *Proc. of CADE-23*, volume 6803 of *LNAI*, pages 338–353. Springer, 2011.
79. D. Jovanović and L. de Moura. Solving non-linear arithmetic. In B. Gramlich, D. Miller, and U. Sattler, editors, *Proc. of IJCAR-6*, volume 7364 of *LNAI*, pages 339–354. Springer, 2012.
80. D. Jovanović and L. de Moura. Cutting to the chase: solving linear integer arithmetic. *J. Automat. Reason.*, 51:79–108, 2013.
81. H. Kirchner, S. Ranise, C. Ringeissen, and D.-K. Tran. Automatic combinability of rewriting-based satisfiability procedures. In M. Hermann and A. Voronkov, editors, *Proc. of LPAR-13*, volume 4246 of *LNAI*, pages 542–556. Springer, 2006.

82. D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Proc. of Computational Problems in Abstract Algebras*, pages 263–298. Pergamon Press, 1970.
83. K. Korovin, N. Tsiskaridze, and A. Voronkov. Conflict resolution. In I. P. Gent, editor, *Proc. of CP-15*, volume 5732 of *LNCS*. Springer, 2009.
84. L. Kovàcs and A. Voronkov. First order theorem proving and Vampire. In N. Sharygina and H. Veith, editors, *Proc. of CAV-25*, volume 8044 of *LNCS*, pages 1–35. Springer, 2013.
85. S. Krstić and A. Goel. Architecting solvers for SAT modulo theories: Nelson-Oppen with DPLL. In F. Wolter, editor, *Proc. of FroCoS-6*, volume 4720 of *LNAI*, pages 1–27. Springer, 2007.
86. D. S. Lankford. Canonical inference. Memo ATP-32, Automatic Theorem Proving Project, Univ. of Texas at Austin, 1975.
87. V. Lifschitz, L. Morgenstern, and D. A. Plaisted. Knowledge representation and classical logic. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, volume 1, pages 3–88. Elsevier, 2008.
88. C. A. Lynch and B. Morawska. Automatic decidability. In G. Plotkin, editor, *Proc. of LICS-17*. IEEE Computer Society, 2002.
89. C. A. Lynch and B. Morawska. Basic syntactic mutation. In A. Voronkov, editor, *Proc. of CADE-18*, volume 2392 of *LNAI*, pages 471–485. Springer, 2002.
90. C. A. Lynch, S. Ranise, C. Ringeissen, and D. Tran. Automatic decidability and combinability. *Inf. Comput.*, 209(7):1026–1047, 2011.
91. C. A. Lynch and D.-K. Tran. Automatic decidability and combinability revisited. In F. Pfenning, editor, *Proc. of CADE-21*, volume 4603 of *LNAI*, pages 328–344. Springer, 2007.
92. Z. Manna and C. G. Zarba. Combining decision procedures. In B. K. Aichernig and T. S. E. Maibaum, editors, *Formal Methods at the Crossroads. From Panacea to Foundational Support*, volume 2757 of *LNCS*, pages 381–422. Springer, 2002.
93. J. Marques Silva and K. A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. on Computers*, 48(5):506–521, 1999.
94. J. P. Marques Silva, I. Lynce, and S. Malik. Conflict-driven clause learning SAT solvers. In A. Biere, M. Heule, H. Van Maaren, and T. Walsh, editors, *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 131–153. IOS Press, 2009.
95. J. W. McCarthy. Towards a mathematical science of computation. In C. M. Popplewell, editor, *Proc. of IFIP 1962*, pages 21–28. North Holland, 1963.
96. K. L. McMillan, A. Kuehlmann, and M. Sagiv. Generalizing DPLL to richer logics. In A. Bouajjani and O. Maler, editors, *Proc. of CAV-21*, volume 5643 of *LNCS*, pages 462–476. Springer, 2009.
97. G. Nelson. Techniques for program verification. Technical Report CSL-81-10, Xerox, Palo Alto Research Center, 1981.
98. G. Nelson. Combining satisfiability procedures by equality sharing. In W. W. Bledsoe and D. W. Loveland, editors, *Automatic Theorem Proving: After 25 Years*, pages 201–211. American Mathematical Society, 1983.
99. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM TOPLAS*, 1(2):245–257, 1979.
100. G. Nelson and D. C. Oppen. Fast decision procedures based on congruence closure. *J. ACM*, 27(2):356–364, 1980.
101. E. Nicolini, C. Ringeissen, and M. Rusinowitch. Data structures with arithmetic constraints: a non-disjoint combination. In S. Ghilardi and R. Sebastiani, editors, *Proc. of FroCoS-7*, volume 5749 of *LNAI*, pages 319–334. Springer, 2009.

102. R. Nieuwenhuis. Decidability and complexity analysis by basic paramodulation. *Inf. Comput.*, 147(1):1–21, 1998.
103. R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT modulo theories: from an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T). *J. ACM*, 53(6):937–977, 2006.
104. R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, chapter 7, pages 371–443. Elsevier, 2001.
105. D. C. Oppen. Complexity, convexity and combinations of theories. *Theor. Comput. Sci.*, 12:291–302, 1980.
106. D. A. Plaisted. Equational reasoning and term rewriting systems. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume I: Logical Foundations, pages 273–364. Oxford Univ. Press, 1993.
107. D. A. Plaisted. Automated theorem proving. *Wiley Interdisc. Rev. Cog. Sci.*, 5(2):115–128, 2014.
108. S. Ranise, C. Ringeissen, and C. G. Zarba. Combining data structures with nonstably infinite theories using many-sorted logic. In B. Gramlich, editor, *Proc. of FroCoS-5*, volume 3717 of *LNAI*, pages 48–64. Springer, 2005.
109. G. Reger, M. Suda, and A. Voronkov. Playing with AVATAR. In A. P. Felty and A. Middeldorp, editors, *Proc. of CADE-25*, volume 9195 of *LNAI*, pages 399–415. Springer, 2015.
110. C. Ringeissen. Cooperation of decision procedures for the satisfiability problem. In F. Baader and K. U. Schulz, editors, *Proc. of FroCoS-1*, Applied Logic, pages 121–140. Kluwer, 1996.
111. C. Ringeissen and V. Senni. Modular termination and combinability for superposition modulo counter arithmetic. In C. Tinelli and V. Sofronie-Stokkermans, editors, *Proc. of FroCoS-8*, volume 6989 of *LNAI*, pages 211–226. Springer, 2011.
112. G. A. Robinson and L. Wos. Paramodulation and theorem-proving in first-order theories with equality. In D. Michie and B. Meltzer, editors, *Machine Intelligence*, volume 4, pages 135–150. Edinburgh Univ. Press, 1969.
113. M. Rusinowitch. Theorem-proving with resolution and superposition. *J. Symb. Comput.*, 11(1 & 2):21–50, 1991.
114. S. Schulz. System description: E 1.8. In K. McMillan, A. Middeldorp, and A. Voronkov, editors, *Proc. of LPAR-19*, volume 8312 of *LNAI*, pages 735–743. Springer, 2013.
115. R. Sebastiani. Lazy satisfiability modulo theories. *J. Satisfiability Bool. Model. and Comput.*, 3:141–224, 2007.
116. N. Shankar. Automated deduction for verification. *ACM Comput. Surv.*, 41(4):40–96, 2009.
117. R. E. Shostak. An algorithm for reasoning about equality. *Commun. ACM*, 21(7):583–585, 1978.
118. V. Sofronie-Stokkermans. Locality results for certain extensions of theories with bridging functions. In R. A. Schmidt, editor, *Proc. of CADE-22*, volume 5663 of *LNAI*, pages 67–83. Springer, 2009.
119. C. Tinelli and M. T. Harandi. A new correctness proof of the Nelson–Oppen combination procedure. In F. Baader and K. U. Schulz, editors, *Proc. of FroCoS-1*, Applied Logic, pages 103–120. Kluwer, 1996.
120. C. Tinelli and C. Ringeissen. Unions of non-disjoint theories and combinations of satisfiability procedures. *Theor. Comput. Sci.*, 290(1):291–353, 2003.

121. C. Tinelli and C. G. Zarba. Combining decision procedures for sorted theories. In J. Alferes and J. Leite, editors, *Proc. of JELIA-9*, volume 3229 of *LNAI*, pages 641–653. Springer, 2004.
122. C. Tinelli and C. G. Zarba. Combining non-stably infinite theories. *J. Automat. Reason.*, 34(3):209–238, 2005.
123. C. Wang, F. Ivančić, M. Ganai, and A. Gupta. Deciding separation logic formulae by SAT and incremental negative cycle elimination. In G. Sutcliffe and A. Voronkov, editors, *Proc. of LPAR-12*, volume 3835 of *LNAI*, pages 322–336. Springer, 2005.
124. C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, and P. Wischnewski. SPASS version 3.5. In R. A. Schmidt, editor, *Proc. of CADE-22*, volume 5663 of *LNAI*, pages 140–145. Springer, 2009.
125. T. Wies, R. Piskac, and V. Kuncak. Combining theories with shared set operations. In S. Ghilardi and R. Sebastiani, editors, *Proc. of FroCoS-7*, volume 5749 of *LNAI*, pages 366–382. Springer, 2009.
126. S. A. Wolfman and D. S. Weld. The LPSAT engine and its application to resource planning. In T. Dean, editor, *Proc. of IJCAI-16*, volume 1, pages 310–316. Morgan Kaufmann, 1999.
127. A. Zeljić, C. M. Wintersteiger, and P. Rümmer. Deciding bit-vector formulas with `mcSAT`. In N. Creignou and D. Le Berre, editors, *Proc. of SAT-19*, volume 9710 of *LNCS*, pages 249–266. Springer, 2016.