$\mathcal{T}$-decision by decomposition

Maria Paola Bonacina[*] and Mnacho Echenim[**]

Dipartimento di Informatica
Università degli Studi di Verona
Strada Le Grazie 15, I-39134 Verona, Italy

Abstract. Much research concerning Satisfiability Modulo Theories is devoted to the design of efficient SMT-solvers that integrate a SAT-solver with $\mathcal{T}$-satisfiability procedures. The rewrite-based approach to $\mathcal{T}$-satisfiability procedures is appealing, because it is general, uniform and it makes combination of theories simple. However, SAT-solvers are unparalleled in handling the large Boolean part of $\mathcal{T}$-decision problems of practical interest. In this paper we present a decomposition framework that combines a rewrite-based theorem prover and an SMT solver in an off-line mode, in such a way that the prover "compiles the theory away," so to speak. Thus, we generalize the rewrite-based approach from $\mathcal{T}$-satisfiability to $\mathcal{T}$-decision procedures, making it possible to use the rewrite-based prover for theory reasoning and the SAT-solver in the SMT-solver for Boolean reasoning. We prove the practicality of this framework by giving decision procedures for the theories of records, integer offsets and arrays.

1 Introduction

Decision procedures are at the heart of formal verification tools, which invoke them to decide the validity of logical formulæ. In software or hardware verification problems, the validity of a formula is to be tested *modulo* a background theory $\mathcal{T}$; such problems are called *$\mathcal{T}$-decision problems*. Relevant theories include linear arithmetic, or fragments thereof, theories of data structures, and *combinations* of simpler theories. Decision procedures for these problems are commonly called *SMT-solvers*, where SMT stands for Satisfiability Modulo Theories.

Due to the typically large Boolean part of the formulæ to be tested, most SMT-solvers combine sophisticated extensions of the so-called *DPLL procedure* (see, e.g., [18]) for propositional satisfiability, with *$\mathcal{T}$-satisfiability procedures*, that decide the $\mathcal{T}$-satisfiability of sets of unit clauses (see, e.g., [13]). A main issue with this approach is the *combination of theories*, that is, the case where $\mathcal{T}$ consists of several simpler theories. Most systems resort to the *Nelson-Oppen combination scheme* [12], so that the theory solver is a Nelson-Oppen reasoner for the combination $\mathcal{T}$ of the smaller theories. However, the Nelson-Oppen scheme works without backtracking only if all involved theories are *convex*, otherwise,

[*] E-mail: mariapaola.bonacina@univr.it
[**] E-mail: echenim@sci.univr.it

also the theory solver requires case analysis by backtracking. This makes the design of an SMT-solver that efficiently overlaps the case analysis required by the theory solver with that required by the DPLL procedure a complex engineering problem. Much work is being invested on this issue, for various theories, combinations of theories and classes of clauses. For instance, several of the systems that take part in the SMT competition[1] showcase clever techniques for this integration.

There are several reasons that make a generic theorem prover attractive to solve SMT problems. From a practical point of view, theorem provers offer a well-balanced trade-off of *robustness, reliability* and *efficiency*, as a result of years of research on data structures and algorithms (e.g., those for indexing techniques). Thus, one can take a theorem prover "off the shelf" to solve SMT problems, without worrying about "false negatives" (lack of soundness) or "false positives" (lack of completeness). From a theoretical point of view, theorem provers are *theory-independent* and *expressive*, since the presentation of the considered theory is part of the input. Combination of theories becomes conceptually simple for the same reason: it suffices to provide the theorem prover with the union of the presentations of the theories to be combined.

The rewrite-based inference system $\mathcal{SP}$ was applied to $\mathcal{T}$-satisfiability in [3, 1, 2, 6], by showing that it is guaranteed to terminate on $\mathcal{T}$-satisfiability problems in several theories. Experimental results and a theorem of *modularity of termination* were also obtained [1, 2]. The modularity theorem shows that if $\mathcal{SP}$ terminates on $\mathcal{T}$-satisfiability problems in each theory, then it terminates also on $\mathcal{T}$-satisfiability problems in their union, provided the theories are *variable-inactive* and do not share function symbols. Under the same assumption of *variable-inactivity*, an approach to generalize the termination results for $\mathcal{SP}$ from $\mathcal{T}$-satisfiability problems to the more general $\mathcal{T}$-decision problems, that involve ground formulæ, was presented in [5]. However, generic theorem provers are not designed to deal with the Boolean part of a formula as efficiently as possible. Hence, they do not seem to be suited to solve $\mathcal{T}$-decision problems that feature a heavy Boolean part, although this is the case for most problems of practical interest. This issue can be addressed by integrating the theorem prover with a SAT solver, as it is done for instance in the haRVey system[2]. However, state-of-the-art SMT-solvers rely on a *tight* integration of the SAT-solver and the $\mathcal{T}$-satisfiability procedures, which is very problematic if the $\mathcal{T}$-satisfiability procedure is a theorem prover with a proof-confluent inference system that does not require search by backtracking. Furthermore, if the integration were tight, the advantage of using the prover "off the shelf" would be lost.

In this paper, we propose a new framework where a $\mathcal{T}$-decision problem is *decomposed* in such a way that it can be solved *by stages*, by pipe-lining a first-order reasoner and an SMT-solver. Intuitively, a ground formula is decomposed into two parts: one that interacts with the theory, and another that contains the Boolean structure of the formula. In some sense, such a decomposition corre-

[1] `http://www.csl.sri.com/users/demoura/smt-comp`

[2] `http://www.loria.fr/equipes/cassis/softwares/haRVey/`

sponds to the separation of the *definitional* part of a program (i.e., the part that consists of statements such as "let $x = f(y)$ in ..."), from its *operational* part. The first-order reasoner is applied to the first part and it "compiles" the theory away, by performing as much theory reasoning as possible. All that remains to do is to test the satisfiability of the union of the boolean part of the formula and the result of this process. In this approach, the call to the theorem prover can be viewed as a *reduction*. However, since this reduction is achieved by generic inferences, it is proof-theoretic in nature and, unlike model-theoretic reductions, it is *independent* of the specific theory or combination of theories under consideration. This new framework may lead to the design of a new generation of SMT-solvers, that will only need to incorporate theory reasoners for the theories left after the reduction, and will thus be easier to implement.

The decomposition framework is general, in that it does not depend on a specific first-order inference system or theorem prover or SMT-solver. In the second part of the paper, we instantiate it to construct $\mathcal{T}$-decision procedures based on $\mathcal{SP}$, and we give specific procedures for the theories of *records*, *integer offsets* and *arrays*. We also show how this scheme enables the system to *postpone* the treatment of a theory: when $\mathcal{T}$ is of the form $\mathcal{T}_1 \cup \mathcal{T}_2$, it is possible to process $\mathcal{T}_1$ first and then deal with $\mathcal{T}_2$. This is especially important when $\mathcal{T}_2$ is a theory that cannot be handled by a theorem prover, such as linear arithmetic: its treatment is passed on to the SMT-solver.

This paper is organized as follows: Section 2 recalls some basic definitions. Section 3 presents the abstract decomposition framework. Section 4 contains its concrete instantiation with the inference system $\mathcal{SP}$. Section 5 shows how the above-mentioned theories of data structures fit into the framework and yield decision procedures. Due to the space limits, all proofs are omitted and can be found in the full version [7].

2 Preliminaries

Given a signature Σ, we assume the standard definitions of Σ-terms, Σ-predicates, Σ-literals and Σ-clauses. As usual, clauses are variable-disjoint. For notation, $\simeq$ is unordered equality, $\bowtie$ is either $\simeq$ or $\not\simeq$, the letters l, r, s, u, v and t denote terms, a, b and c denote constants, w, x, y, z variables, and all other lower-case letters denote constants or function symbols. A theory is presented by a set of sentences, called its *presentation* or *axiomatization*. Given a presentation $\mathcal{T}$, *$\mathcal{T}$-satisfiability* is the problem of deciding whether a set of ground unit clauses is satisfiable in $\mathcal{T}$. The more general *$\mathcal{T}$-decision problem* is the problem of deciding the satisfiability of any ground formula in $\mathcal{T}$. Without loss of generality, we can assume that the considered ground formulæ are sets of clauses.

For sets of clauses S and S', we write $S \equiv_{\mathrm{s}} S'$ to say that S and S' are *equisatisfiable*, that is, S has a model if and only if S' has a model. For a term t, the *depth* of t, denoted by $\mathrm{depth}(t)$, is 0 if t is a constant or variable, and $\mathrm{depth}(f(t_1, \ldots, t_n)) = 1 + \max\{\mathrm{depth}(t_i) \mid i = 1, \ldots, n\}$ otherwise. For literals, we define $\mathrm{depth}(l \bowtie r) = \mathrm{depth}(l) + \mathrm{depth}(r)$. A positive literal is *flat* if its depth is 0 or 1, a negative literal is *flat* if its depth is 0. A literal is *strictly flat* if its depth is 0, and a clause is *flat* (resp. *strictly flat*) if all its literals are.

The operation of *flattening* consists of transforming a finite set of ground clauses S over a signature Σ, into a finite set of ground clauses S' over a signature Σ', in such a way that:

- Σ' is obtained by adding a finite number of constants to Σ;
- every non-unit clause in S' is strictly flat;
- every unit clause in S' is flat and
- for all presentations $\mathcal{T}$, $\mathcal{T} \cup S$ and $\mathcal{T} \cup S'$ are equisatisfiable.

For example, if we flatten the set $\{f(a) \not\simeq f(b) \vee f(a) \not\simeq f(c)\}$, we obtain the equisatisfiable set $\{f(a) \simeq a',\ f(b) \simeq b',\ f(c) \simeq c',\ a' \not\simeq b' \vee a' \not\simeq c'\}$ by introducing fresh constants a', b' and c'.

A *simplification ordering* $\succ$ is an ordering that is *stable*, *monotonic* and contains the *subterm ordering*: if $s \succ t$, then $c[s]\sigma \succ c[t]\sigma$ for any context c and substitution σ, and if t is a subterm of s then $s \succ t$. A *complete simplification ordering*, or CSO, is a simplification ordering that is total on ground terms. We write $t \prec s$ if $s \succ t$. More details on orderings can be found in surveys such as [10]. An *inference system* Inf consists of a set of *inference rules*, separated into *expansion rules*, that generate clauses, and *contraction rules*, that delete or simplify clauses. If the inference system is based on a CSO $\succ$, we write $\mathrm{Inf}_{\succ}$ for Inf equipped with $\succ$. An $\mathrm{Inf}_{\succ}$-*derivation* is a sequence

$$S_0 \vdash_{\mathrm{Inf}_{\succ}} S_1 \vdash_{\mathrm{Inf}_{\succ}} \ldots S_i \vdash_{\mathrm{Inf}_{\succ}} \ldots,$$

where each S_i is a set of clauses, obtained by applying an expansion or a contraction rule to clauses in S_{i-1}. The *limit* of such a derivation is the set of *persistent clauses*: $S_\infty = \bigcup_{j\geq 0} \bigcap_{i \geq j} S_i$. If a derivation is finite and of length n, we may write $S_0 \vdash^n_{\mathrm{Inf}_{\succ}} S_n$.

The *superposition calculus*, or $\mathcal{SP}$, is a refutationally complete *rewrite-based inference system* for first-order logic with equality (see, e.g., [14]). It consists of expansion and contraction inference rules. Since it is based on a CSO on terms, extended to literals and clauses in a standard way, we write $\mathcal{SP}_{\succ}$ to specify the ordering. A *strategy*, denoted by $\mathfrak{S}$, is given by an inference system and a search plan that controls the application of the inference rules. A strategy with inference system $\mathcal{SP}_{\succ}$ is called an $\mathcal{SP}_{\succ}$-strategy.

3 The decomposition framework

When approaching SMT problems in the context of generic first-order reasoning, a problem has the form $\mathcal{T} \cup P$, where $\mathcal{T}$ is the presentation of the theory and P is a set of ground clauses, since the theory is not necessarily built into the inference system. We propose an approach that consists in "compiling" the theory away, so that a problem $\mathcal{T} \cup P$ is transformed into the satisfiability problem of another set of clauses, which does not mention $\mathcal{T}$. Such a problem transformation can be

viewed as a *reduction*, with the distinction that the problem is approached from a proof-theoretic point of view and *does not depend on the considered theory.* In order to investigate under what conditions this problem transformation is correct, we solve a more general problem:
Question: For a presentation $\mathcal{T}$ and sets of clauses S, S', A and A' such that $\mathcal{T} \cup S \equiv_s A$ and $\mathcal{T} \cup S' \equiv_s A'$, what conditions guarantee that $\mathcal{T} \cup S \cup S' \equiv_s A \cup A'$?

3.1 Supported strategies and $\mathcal{T}$-compatibility

In order to separate the clauses in $\mathcal{T}$ from the other clauses, we adopt a *supported strategy* (cf. Section 2.6 in [4]), with an inference system $\mathrm{Inf}_{\succ}$, that works on *pairs* $(\mathcal{T}, SOS)$ of sets of clauses, where SOS stands for set-of-support. We assume that all inference rules in $\mathrm{Inf}_{\succ}$ are either unary or binary, all expansion inferences have at least one premise from SOS, all clauses generated by expansion are added to SOS, and $\mathrm{Inf}_{\succ}$ is *refutationally complete for* $\mathcal{T}$. In what follows, when a clause D is generated by a *binary expansion rule* applied to C and C', we will need to distinguish between the first and the second premise of the applied inference rule, hence the following definition:

Definition 1. *A clause D is* generated from parents *C and C' if it is generated by a binary expansion rule applied to C and C', with C as first premise and C' as second premise.*

$A \equiv_s \mathcal{T} \cup S$ and $A' \equiv_s \mathcal{T} \cup S'$ do not imply that $A \cup A' \equiv_s \mathcal{T} \cup S \cup S'$, since equisatisfiability is not preserved by the union operation. A naïve solution would be to impose that A and $\mathcal{T} \cup S$ (resp. A' and $\mathcal{T} \cup S'$) are logically equivalent, but this would be too strong a requirement, satisfied by choosing $A = \mathcal{T} \cup S$ and $A' = \mathcal{T} \cup S'$, which would defeat the purpose. Thus, we relax the model-theoretic condition of logical equivalence by still requiring $\mathcal{T} \cup S \models A$ and $\mathcal{T} \cup S' \models A'$, while replacing $A \models \mathcal{T} \cup S$ and $A' \models \mathcal{T} \cup S'$ with weaker requirements. Then, we define proof-theoretic conditions that suffice to retain equisatisfiability.

Definition 2. *Let C be a clause and S and $\mathcal{T}$ be sets of clauses. The set of* $\mathcal{T}$-children *of C is defined by:*

$$\mathcal{G}(C, \mathcal{T}) = \{F \mid \exists Q \in \mathcal{T} : F \text{ is generated from parents } C \text{ and } Q\}.$$

The set $\mathcal{D}_i(C, \mathcal{T})$ of i-steps $\mathcal{T}$-descendants *of C is defined inductively as follows:*

$$\mathcal{D}_0(C, \mathcal{T}) = \{C\} \quad \text{and} \quad \mathcal{D}_{i+1}(C, \mathcal{T}) = \bigcup_{F \in \mathcal{D}_i(C, \mathcal{T})} \mathcal{G}(F, \mathcal{T}), \text{ for } i \geq 0.$$

The sets of $\mathcal{T}$-descendants *of C and* $\mathcal{T}$-descendants *of S are defined by:*

$$\mathcal{D}(C, \mathcal{T}) = \bigcup_{i \geq 0} \mathcal{D}_i(C, \mathcal{T}) \quad \text{and} \quad \mathcal{D}(S, \mathcal{T}) = \bigcup_{C \in S} \mathcal{D}(C, \mathcal{T}).$$

Instead of imposing that A and A' logically entail $\mathcal{T} \cup S$ and $\mathcal{T} \cup S'$ respectively, we only require that they entail the $\mathcal{T}$-descendants of S and S'. This property is termed *$\mathcal{T}$-compatibility*:

Definition 3. *Given a set of clauses A, a clause C (resp. a set of clauses S) is* $\mathcal{T}$-compatible with A *if $A \models \mathcal{D}(C, \mathcal{T})$ (resp. $A \models \mathcal{D}(S, \mathcal{T})$).*

The name $\mathcal{T}$-compatibility is meant to convey the intuition that this relation between sets is weaker than logical equivalence but stronger than equisatisfiability. Strictly speaking, the notions of $\mathcal{T}$-descendant, $\mathcal{T}$-compatibility, and all those that depend on them, should be parametric with respect to the inference system: for example, a set of clauses may be $\mathcal{T}$-compatible with a given set for one inference system but not for another one. Nevertheless, for the sake of readability, we let the dependence on the inference system remain implicit. We also use the notion of $\mathcal{T}$-compatibility to specify formally what it means for a set of clauses not to "interact" with $\mathcal{T}$. This property is captured by the notion of *$\mathcal{T}$-disconnection*:

Definition 4. *A set of clauses S is* $\mathcal{T}$-disconnected *if $\mathcal{D}(S, \mathcal{T}) = S$. If $S = \{C\}$, we may also write that C is $\mathcal{T}$-disconnected.*

Intuitively, if a set of clauses S is $\mathcal{T}$-compatible with itself, it means that it does not need to interact with $\mathcal{T}$. The notion of $\mathcal{T}$-disconnection entails this property:

Proposition 1. *If S is $\mathcal{T}$-disconnected, then S is $\mathcal{T}$-compatible with itself.*

3.2 Preserving $\mathcal{T}$-compatibility

We define a few restrictions guaranteeing that if a set of clauses S is $\mathcal{T}$-compatible with A and $(\mathcal{T}, S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}, S')$, then S' is also $\mathcal{T}$-compatible with A. As we introduce them by considering the possible $\mathrm{Inf}_\succ$-inferences, we start with contraction inferences.

Definition 5. *A set of clauses S is* $\mathcal{T}$-contraction-compatible *if for all sets A, for all contraction inferences $(\mathcal{T}, S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}, S')$, if S is $\mathcal{T}$-compatible with A then S' is also $\mathcal{T}$-compatible with A.*

$\mathcal{T}$-compatibility is clearly preserved by contraction rules that delete a clause in S, such as *subsumption* and *tautology deletion*. On the other hand, the behavior of rules such as *simplification* depends on $\mathcal{T}$ and the given set of clauses:

Example 1. Let $\mathcal{T} = \{f(a) \simeq b\}$, $S = \{f(c) \simeq d,\ c \simeq a\}$, and assume that $\mathcal{D}(S, \mathcal{T}) = S$, that is, S is $\mathcal{T}$-compatible with itself. Suppose that $(\mathcal{T}, S) \vdash_{\mathrm{Inf}_\succ}$ $(\mathcal{T}, S')$, where S' is obtained from S by the simplification of $f(c) \simeq d$ by $c \simeq a$, i.e., $S' = \{f(a) \simeq d,\ c \simeq a\}$. If $\mathrm{Inf}_\succ$ features a rule generating $d \simeq b$ from parents $f(a) \simeq d$ and $f(a) \simeq b$, then $b \simeq d \in \mathcal{D}(S', \mathcal{T})$. In this case, S' is not $\mathcal{T}$-compatible with S, since $S \not\models \{b \simeq d\}$.

We proceed with expansion inferences, distinguishing between unary and binary ones.

Unary expansion inferences. The property of *$\mathcal{T}$-neutrality* is sufficient to control unary inferences. Intuitively, this notion prevents clauses generated by unary inferences from interacting with clauses in $\mathcal{T}$.

Definition 6. *A clause C is $\mathcal{T}$-*neutral *if, for every clause D generated from C by a unary inference, D is $\mathcal{T}$-disconnected. A set of clauses is $\mathcal{T}$-*neutral *if all its clauses are.*

Proposition 2. *Let A be a set of clauses such that S is $\mathcal{T}$-compatible with A, and suppose that S is $\mathcal{T}$-neutral. If $(\mathcal{T}, S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}, S \cup \{D\})$, where D is generated from a clause in S by a unary inference, then $S \cup \{D\}$ is $\mathcal{T}$-compatible with A.*

Binary expansion inferences between a clause in $\mathcal{T}$ and one in S. With Definitions 1, 2 and 3, we defined $\mathcal{T}$-compatibility for binary expansion inferences involving a clause in S as first premise and a clause in $\mathcal{T}$ as second premise. We require these to be the only binary expansion inferences that can take place between clauses in S and clauses in $\mathcal{T}$. This leads to the notion of *$\mathcal{T}$-orientation*:

Definition 7. *A clause C is $\mathcal{T}$-*oriented *if no binary expansion inference applies with a clause in $\mathcal{T}$ as first premise and C as second premise. A set of clauses is* $\mathcal{T}$-oriented *if all its clauses are.*

Proposition 3. *Let A be a set of clauses such that S is $\mathcal{T}$-compatible with A and suppose S is $\mathcal{T}$-oriented. If $(\mathcal{T}, S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}, S \cup \{D\})$, where D is generated by a binary inference applied to a clause in S and one in $\mathcal{T}$, then $S \cup \{D\}$ is $\mathcal{T}$-compatible with A.*

Example 2. Let $\mathcal{T} = \{a \simeq b\}$, $S = \{f(b) \simeq c\}$, and let $\mathrm{Inf}_\succ$ be an inference system featuring a rule that generates $f(a) \simeq c$ from parents $a \simeq b$ and $f(b) \simeq c$. Then S is not $\mathcal{T}$-oriented, since this rule involves a clause in $\mathcal{T}$ as first premise and a clause in S as second premise. Suppose further that S is $\mathcal{T}$-compatible with itself. Since $(\mathcal{T}, S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}, S')$, with $S' = \{f(b) \simeq c, f(a) \simeq c\}$, and $S \not\models S'$, the set S' is not $\mathcal{T}$-compatible with S.

This example shows how, for inference systems such as $\mathcal{SP}$ where the binary expansion rules are *superposition* and *paramodulation*, the notion of first and second premise instantiates to the traditional notion of clauses paramodulated *from* and *into*, respectively. In such a context, C is $\mathcal{T}$-oriented if no clause of $\mathcal{T}$ paramodulates into C. This property is satisfied by all persistent clauses generated by $\mathcal{SP}$ from $\mathcal{T}$-satisfiability problems in the theories of in [3, 2, 6].

Binary expansion inferences within S. These are the inferences that require the most control. For this purpose, we define a notion of *associativity*, that lends itself to an easy symbolic representation: informally, let $C \rightarrow C'$ denote a clause generated from parents C and C', then C is $[C', D']$-*associative* if either $(C \rightarrow C') \rightarrow D' = C \rightarrow (C' \rightarrow D')$, or $(C \rightarrow C') \rightarrow D'$ is subsumed by $C' \rightarrow D'$. This property is relevant when D' is an axiom in $\mathcal{T}$. Intuitively, it means that if an inference between two clauses yields another inference with an axiom as second premise, then an inference with an axiom as second premise could have been done before hand. This disentangles inferences into axioms from the others, allowing us to do them first.

Definition 8. *Given the clauses C, C' and D', the clause C is $[C', D']$-* associative *if for every clause D generated from parents C and C', and for every clause E generated from parents D and D', there exists a clause E' generated from parents C' and D' such that either E can be generated from parents C and E', or E' subsumes E. The clause C is* weakly $\mathcal{T}$-associative for C' *if for every $Q \in \mathcal{T}$, C is $[C', Q]$-associative. C is* $\mathcal{T}$-associative for C' *if it is weakly $\mathcal{T}$-associative for every $E' \in \mathcal{D}(C', \mathcal{T})$. A set of clauses S is* weakly $\mathcal{T}$-associative *(resp.* $\mathcal{T}$-associative*) if for all $C, C' \in S$, C is weakly $\mathcal{T}$-associative for C' (resp. $\mathcal{T}$-associative for C').*

One can prove that if S is $\mathcal{T}$-associative, it is also $\mathcal{T}$-contraction-compatibile[3]. This requires to instantiate $\mathrm{Inf}_{\succ}$ to a concrete inference system (cf. Lemma 29 in [7] for this result for the system $\mathcal{U}_{\succ}$ of Sec. 5). Since $\mathcal{T}$-associativity is defined using subsumption, to ensure that $\mathcal{T}$-compatibility is preserved by binary inferences between clauses in S, we need subsumption to preserve $\mathcal{T}$-compatibility:

Definition 9. *A clause C is* $\mathcal{T}$-subsumption-preserving *if for all sets A that C is $\mathcal{T}$-compatible with and for all clauses C' subsumed by C, clause C' is also $\mathcal{T}$-compatible with A. A set of clauses S is* $\mathcal{T}$-subsumption-preserving *if all its clauses are $\mathcal{T}$-subsumption-preserving.*

$\mathcal{T}$-subsumption-preservation, together with $\mathcal{T}$-associativity, guarantees that if D is generated by a binary expansion inference applied to two clauses in S, then $S \cup \{D\}$ is $\mathcal{T}$-compatible with A. The statement of the following lemma is actually stronger and is also used to prove that $\mathcal{T}$-compatible sets can be combined, provided they are $\mathcal{T}$-associative:

Lemma 1. *Let C, C' and D be clauses such that D is generated from parents C and C', and let A (resp. A') be a set of clauses such that C is $\mathcal{T}$-compatible with A (resp. C' with A'). If $\mathcal{D}(C', \mathcal{T})$ is $\mathcal{T}$-subsumption-preserving and C is $\mathcal{T}$-associative for C', then D is $\mathcal{T}$-compatible with $A \cup A'$.*

3.3 $\mathcal{T}$-stability

In order to ensure that all relevant properties are preserved by all rules in $\mathrm{Inf}_{\succ}$, we introduce $\mathcal{T}$*-closure* and $\mathcal{T}$*-stability*:

Definition 10. *Given a presentation $\mathcal{T}$, a set of clauses S is* $\mathcal{T}$-closed under $\mathrm{Inf}_{\succ}$ *if all clauses generated by an inference applied to $(\mathcal{T}, S)$ are in S.*

[3] The set of clauses S of Example 1 satisfies neither property.

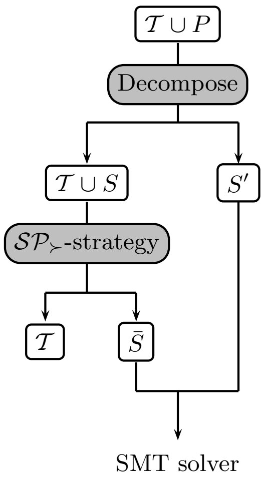

Fig. 1. $\mathcal{T}$-decision procedures based on $\mathcal{SP}$

Definition 11. *Given a presentation $\mathcal{T}$, a set of clauses $\mathcal{B}$ is* $\mathcal{T}$-stable for $\mathrm{Inf}_{\succ}$ *if every subset of $\mathcal{B}$ is $\mathcal{T}$-contraction-compatible and the set $\mathcal{B}$ is* (i) *$\mathcal{T}$-oriented,* (ii) *$\mathcal{T}$-neutral,* (iii) *weakly $\mathcal{T}$-associative,* (iv) *$\mathcal{T}$-subsumption-preserving and* (v) *$\mathcal{T}$-closed under* $\mathrm{Inf}_{\succ}$.

We are now in a position to state the main theorem:

Theorem 1. *Given a presentation $\mathcal{T}$, let $\mathcal{B}$ be a set of clauses that is $\mathcal{T}$-stable for* $\mathrm{Inf}_{\succ}$ *and consider sets of clauses $S, S' \subseteq \mathcal{B}$. If A and A' are sets of clauses such that* (i) $\mathcal{T} \cup S \models A$, (ii) $\mathcal{T} \cup S' \models A'$, (iii) *$S$ is $\mathcal{T}$-compatible with A and* (iv) *S' is $\mathcal{T}$-compatible with A', then $\mathcal{T} \cup S \cup S'$ and $A \cup A'$ are equisatisfiable.*

4 $\mathcal{T}$-decision procedures based on $\mathcal{SP}$

Let $\mathcal{T}$ be a presentation such that a fair $\mathcal{SP}_{\succ}$-strategy is a $\mathcal{T}$-satisfiability procedure. We outline how an $\mathcal{SP}$-based $\mathcal{T}$-decision procedure can be designed by applying the decomposition framework. The binary expansion rules of $\mathcal{SP}$ are superposition and paramodulation. We use paramodulation for both for simplicity. In a typical theory presentation (e.g., those considered in [3, 2, 6]), no paramodulation applies from a non-ground axiom into a strictly flat clause. This, together with the fact that every set of ground clauses can be flattened into an equisatisfiable set containing flat unit clauses and strictly flat clauses, is used to test the satisfiability of $\mathcal{T} \cup P$ as shown in Fig. 1. The set P is decomposed into $S \uplus S'$, where S only contains *flat unit clauses*. A fair $\mathcal{SP}_{\succ}$-strategy is applied to $\mathcal{T} \cup S$, to generate a finite limit set $\mathcal{T} \cup \bar{S}$. Then the satisfiability of $\bar{S} \cup S'$ is tested. To guarantee that this process is correct, we must prove that $\mathcal{T} \cup P$ and $\bar{S} \cup S'$ are equisatisfiable. Therefore we check whether the hypotheses of Theorem 1 are

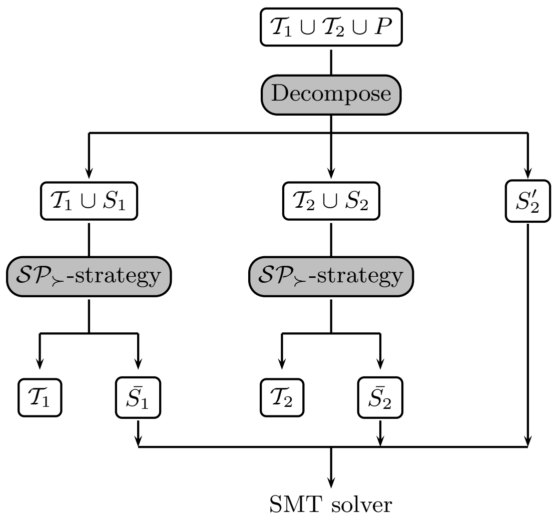

Fig. 2. $\mathcal{T}$-decision procedure in a combination of theories

satisfied. For this purpose, we define first an inference system $\mathrm{Inf}_\succ$, such that S is $\mathcal{T}$-compatible with $\bar{S}$ and S' is $\mathcal{T}$-disconnected. Then, we verify that there exists a set $\mathcal{B}_\mathcal{T}$ that includes S and S' and is $\mathcal{T}$-stable for $\mathrm{Inf}_\succ$. If these conditions are met, Theorem 1 applies with A instantiated to $\bar{S}$ and A' instantiated to S'. Indeed, we have $\mathcal{T} \cup S \models \bar{S}$, $S' \models S'$, S is $\mathcal{T}$-compatible with $\bar{S}$, and since S' is $\mathcal{T}$-disconnected, it is $\mathcal{T}$-compatible with itself by Proposition 1. It follows that $\mathcal{T} \cup P \equiv_\mathrm{s} \bar{S} \cup S'$.

If $\mathcal{T}$ is a combination of theories, that is, $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, the scheme of Fig. 1 can be used again, under the assumption that $\mathcal{T}_2$ is $\mathcal{T}_1$-disconnected. Indeed, $\mathcal{T}_2$ is then contained in S', and the sets $\mathcal{T}_1 \cup \mathcal{T}_2 \cup P$ and $\bar{S} \cup S'$ are equisatisfiable. This is especially useful when $\mathcal{T}_2$ is a presentation that $\mathcal{SP}$ cannot handle, such as the (infinite) axiomatization of linear arithmetic: the theory is *ignored by the prover* and dealt with by the specialized procedure of the SMT solver. On the other hand, if the inference system can handle both theories, the more specialized scheme of Fig. 2 applies:

The 2-theories scheme. Assume that a fair $\mathcal{SP}_\succ$-strategy is also a $\mathcal{T}_2$-satisfiability procedure and $\bar{S}$ is $\mathcal{T}_2$-disconnected. The set of clauses P is decomposed into S_1 and $\mathcal{T}_2 \cup S'_1$, in such a way that S_1 only contains flat unit $\mathcal{T}_1$-clauses. In turn, S'_1 is decomposed into S_2 and S'_2, in such a way that S_2 only contains flat unit $\mathcal{T}_2$-clauses. The $\mathcal{SP}_\succ$-strategy applied to $\mathcal{T}_1 \cup S_1$ generates the limit $\mathcal{T}_1 \cup \bar{S}_1$, and, by hypothesis, $\bar{S}_1$ is $\mathcal{T}_2$-disconnected. The $\mathcal{SP}_\succ$-strategy applied to $\mathcal{T}_2 \cup S_2$ generates the limit $\mathcal{T}_2 \cup \bar{S}_2$. By Theorem 1,

$$\begin{aligned}\mathcal{T}_1 \cup \mathcal{T}_2 \cup P &\equiv_\mathrm{s} (\mathcal{T}_1 \cup S_1) \ \cup\ (\mathcal{T}_2 \cup S'_1) \equiv_\mathrm{s} \bar{S}_1 \ \cup\ [(\mathcal{T}_2 \cup S_2) \ \cup\ S'_2] \\ &\equiv_\mathrm{s} (\mathcal{T}_2 \cup S_2) \ \cup\ (\bar{S}_1 \cup S'_2) \equiv_\mathrm{s} \bar{S}_1 \ \cup\ \bar{S}_2 \ \cup\ S'_2,\end{aligned}$$

so that the resulting procedure is correct. In the sequel, we shall see that the 2-theories scheme applies to the theories of records, integer offsets and arrays.

5 Some $\mathcal{T}$-stable sets

In this section we apply our design of $\mathcal{T}$-decision procedures to the theories of records, integer offsets and arrays. It was already proved in [3, 2] that a fair $\mathcal{SP}_{\succ}$-strategy is a $\mathcal{T}$-satisfiability procedure for these theories. We first define an inference system $\mathcal{U}_{\succ}$ that will be the adopted inference system when determining $\mathcal{T}$-stable sets. Then for each of the considered theories, we prove that the scheme described above can be applied to decide the satisfiability of $\mathcal{T} \cup P$, for all sets of ground clauses P. More specifically, we describe how P is decomposed into $S \uplus S'$, and then prove there exists a $\mathcal{T}$-stable set for $\mathcal{U}_{\succ}$ that contains S and S'.

5.1 The inference system $\mathcal{U}_{\succ}$

Superposition and paramodulation in $\mathcal{SP}$ are restricted in such a way that only maximal sides of maximal literals are paramodulated from and into. We define an $\mathcal{SP}$-based supported strategy that relaxes this restriction to some extent:

Definition 12. *Let $\mathcal{U}_{\succ}$ be the strategy that works on pairs of sets $(\mathcal{T}, S)$, contains all the contraction rules of $\mathcal{SP}_{\succ}$, and such that $(\mathcal{T}, S) \vdash_{\mathcal{U}_{\succ}} (\mathcal{T}, S \cup \{D\})$, if D is generated by:*

- *a unary inference rule of $\mathcal{SP}_{\succ}$ applied to a clause in S,*
- *a binary inference rule of $\mathcal{SP}_{\succ}$ applied from a clause in $\mathcal{T}$ into one in S,*
- *a binary inference rule of $\mathcal{SP}_{\succ}$ applied to two clauses in S, with the relaxation that if $u \simeq v \vee D$ paramodulates into $C = l[u'] \bowtie r \vee C'$, neither $l[u'] \bowtie r$ is required to be a maximal literal in C nor $l[u']$ is required to be maximal in this literal.*

Proposition 4. *Let $\mathcal{T}$ be a presentation. For all sets of flat literals S, if $S_\infty = \mathcal{T} \cup \bar{S}$ is the limit generated by a fair $\mathcal{SP}_{\succ}$-strategy from $\mathcal{T} \cup S$, then S is $\mathcal{T}$-compatible with $\bar{S}$ for $\mathcal{U}_{\succ}$. Furthermore, if no $\mathcal{SP}_{\succ}$-inference applies from a clause C into a clause in $\mathcal{T}$, then C is $\mathcal{T}$-disconnected for $\mathcal{U}_{\succ}$.*

5.2 The theory of records

The theory of records with n fields assumes a signature Σ that contains, for all i, $1 \leq i \leq n$, the function symbol rstore_i, which stores a value in the ith-field of a record, and $\mathrm{rselect}_i$, which extracts a value from the ith-field of a record. It is defined by the following (saturated) presentation, denoted by $\mathcal{R}$:

$$\forall x, v. \quad \mathrm{rselect}_i(\mathrm{rstore}_i(x, v)) \simeq v \quad \text{for all } 1 \leq i \leq n, \tag{1}$$

$$\forall x, v. \ \mathrm{rselect}_j(\mathrm{rstore}_i(x, v)) \simeq \mathrm{rselect}_j(x) \quad \text{for all } 1 \leq i \neq j \leq n. \tag{2}$$

The theory of records with extensionality is axiomatized by the (saturated) presentation $\mathcal{R}^e$, which consists of the previous axioms together with:

$$\forall x, y.\ (\textstyle\bigwedge_{i=1}^{n} \mathrm{rselect}_i(x) \simeq \mathrm{rselect}_i(y)) \rightarrow x \simeq y.$$

An $\mathcal{R}^e$-satisfiability problem can be reduced to an $\mathcal{R}$-satisfiability problem:

Lemma 2 (Lemma 1 of [2]). *Let $S = S_1 \uplus S_2$ be a set of ground flat literals, such that S_2 contains the literals of the form $l \not\simeq r$, where l and r are records. For all $L = l \not\simeq r \in S_2$ let C_L denote the clause $\bigvee_{i=1}^{n} \mathrm{rselect}_i(l) \not\simeq \mathrm{rselect}_i(r)$. Then $\mathcal{R}^e \cup S \ \equiv_{\mathrm{s}}\ \mathcal{R} \cup S_1 \cup \{C_L \mid L \in S_2\}$.*

Since an $\mathcal{R}^e$-decision problem can be reduced to an $\mathcal{R}^e$-satisfiability problem by reduction to disjunctive normal form, this reduction holds also for $\mathcal{R}^e$-decision problems. A given set of ground clauses P is decomposed into S and S' as follows: P is flattened; among the resulting clauses, the unit clauses of the form $\mathrm{rstore}_i(a, e) \simeq b$ go into S and all other clauses go into S'. Indeed, the decomposition scheme of Fig. 1 only requires that S is made of flat unit clauses, so that decomposition does not coincide with flattening. It is simple to check that S' is $\mathcal{R}$-disconnected for $\mathcal{U}_{\succ}$.

Definition 13. *Let $\mathcal{B}_{\mathcal{R}}$ denote the set of ground clauses consisting of:*

i) all strictly flat clauses,
ii) all clauses of the form $\mathrm{rstore}_i(a, e) \simeq b \vee B$ where $1 \leq i \leq n$ and B is ground and strictly flat,
iii) all clauses of the form $\mathrm{rselect}_i(a) \simeq e \vee B$ where $1 \leq i \leq n$ and B is ground and strictly flat,
iv) all clauses of the form $\mathrm{rselect}_i(a) \simeq \mathrm{rselect}_i(b) \vee B$, where $1 \leq i \leq n$ and B is ground and strictly flat.

Theorem 2. *$\mathcal{B}_{\mathcal{R}}$ is $\mathcal{R}$-stable for $\mathcal{U}_{\succ}$.*

Since $\mathcal{B}_{\mathcal{R}}$ contains $S \uplus S'$ and is $\mathcal{R}$-stable for $\mathcal{U}_{\succ}$, the scheme of Fig. 1 applies to $\mathcal{R}$. Since $\mathcal{B}_{\mathcal{R}}$ only contains ground clauses and is $\mathcal{T}$-closed under $\mathcal{U}_{\succ}$, not only S and S', but also $\bar{S}$ is ground. Thus, $\bar{S} \cup S'$ is ground, and its satisfiability can be decided by a decision procedure for the theory of Equality with Uninterpreted Function symbols (EUF).

5.3 The theory of integer offsets

The theory of integer offsets is a fragment of the theory of integers. Its signature Σ contains two unary function symbols s and p, that represent the successor and predecessor functions, respectively. This theory is presented by the following (infinite) set of axioms $\mathcal{I}$:

$$\begin{array}{l} \forall x.\ \mathsf{s}(\mathsf{p}(x)) \simeq x, \\ \forall x.\ \mathsf{p}(\mathsf{s}(x)) \simeq x, \\ \forall x.\ \ \mathsf{s}^i(x) \ \ \not\simeq x \text{ for } i > 0, \end{array}$$

where $\mathsf{s}^0(x) = x$ and $\mathsf{s}^{i+1}(x) = \mathsf{s}(\mathsf{s}^i(x))$ for $i \geq 0$. For the sake of convenience, we also define for all $n \in \mathbb{N}$

$$\begin{aligned} A_{\mathcal{I}} &= \{\mathsf{s}(\mathsf{p}(x)) \simeq x,\ \mathsf{p}(\mathsf{s}(x)) \simeq x\}, \\ Ac(n) &= \{\mathsf{s}^i(x) \not\simeq x \mid 0 < i \leq n\}, \\ Ac &= \textstyle\bigcup_{n \geq 0} Ac(n). \end{aligned}$$

It was proved in [5] that if S is a set of flat ground literals, then for all n greater or equal to the number of occurrences of the function symbols s and p in S, $A_{\mathcal{I}} \cup Ac \cup S \equiv_{\mathrm{s}} A_{\mathcal{I}} \cup Ac(n) \cup S$. We deduce a similar result for $\mathcal{I}$-decision problems:

Proposition 5. *Let P be an $\mathcal{I}$-decision problem containing n occurrences of the function symbols s and p. Then $A_{\mathcal{I}} \cup Ac \cup P \equiv_{\mathrm{s}} A_{\mathcal{I}} \cup Ac(n) \cup P$.*

Given $n \in \mathbb{N}$, let $A_{\mathrm{s}}(n)$ denote the set $\{\mathsf{s}^i(x) \not\simeq \mathsf{p}^j(x) \mid 1 \leq i + j \leq n\}$. The saturated limit of $A_{\mathcal{I}} \cup Ac(n)$ has the form $A_{\mathcal{I}} \cup A_{\mathrm{s}}(n)$. Let $\mathcal{I}_{\mathrm{s}}[n]$ denote this saturated set. A set of ground clauses P is decomposed as follows: P is flattened; all the resulting unit clauses go into S and all strictly flat clauses go into S'. Thus, in this case, decomposition coincides with flattening.

Definition 14. *Let $\mathcal{B}_{\mathcal{I}_{\mathrm{s}}[n]}$ denote the set consisting of:*

i) *all strictly flat clauses,*
ii) *all clauses of the form $\mathsf{p}(a) \simeq b \vee B$ where B is ground and strictly flat,*
iii) *all clauses of the form $\mathsf{s}(a) \simeq b \vee B$ where B is ground and strictly flat,*
iv) *all clauses of the form $\mathsf{s}^i(a) \not\simeq \mathsf{p}^j(b) \vee B$, where $0 \leq i + j \leq n - 1$ and B is ground and strictly flat.*

Theorem 3. *$\mathcal{B}_{\mathcal{I}_{\mathrm{s}}[n]}$ is $\mathcal{I}_{\mathrm{s}}[n]$-stable for $\mathcal{U}_{\succ}$.*

Since $\mathcal{B}_{\mathcal{I}_{\mathrm{s}}[n]}$ only contains ground clauses, by the same observation made for the theory of records, the satisfiability of $\bar{S} \cup S'$ can be decided by a decision procedure for EUF.

5.4 The theory of arrays

The theory of arrays is defined by the following (saturated) presentation $\mathcal{A}$:

$$\forall x, z, v.\ \mathrm{select}(\mathrm{store}(x, z, v), z) \simeq v, \tag{3}$$

$$\forall x, z, w, v.\ (z \simeq w \vee \mathrm{select}(\mathrm{store}(x, z, v), w) \simeq \mathrm{select}(x, w)). \tag{4}$$

The theory of arrays with extensionality $\mathcal{A}^{\mathrm{e}}$ is defined by axioms (3) and (4), along with the following extensionality axiom:

$$\forall x, y.\ (\forall z.\ \mathrm{select}(x, z) \simeq \mathrm{select}(y, z) \rightarrow x \simeq y). \tag{5}$$

It was proved in [3] that satisfiability of sets of ground literals in $\mathcal{A}^{\mathrm{e}}$ can be reduced to satisfiability of sets of ground literals in $\mathcal{A}$ by replacing every negative literal of the form $a \not\simeq a'$, where a and a' are arrays, by a literal $\mathrm{select}(a, sk) \not\simeq \mathrm{select}(a', sk)$, where sk is a fresh (Skolem) constant. Such a reduction also holds for sets of ground clauses. A set of ground clauses P is decomposed as follows: P is flattened; S contains all the unit clauses of the form $\mathrm{store}(a, i, e) \simeq a'$, and S' contains all the other clauses.

Definition 15. *Consider the set $\mathcal{B}_{\mathcal{A}}$ consisting of:*

i) *all strictly flat clauses,*
ii) *all clauses of the form* $\mathrm{store}(a, i, e) \simeq a' \vee B$ *where B is strictly flat,*
iii) *all clauses of the form* $\mathrm{select}(a, i) \simeq e \vee B$ *where B is strictly flat,*
iv) *all clauses of the form* $\mathrm{select}(a, x) \simeq \mathrm{select}(a', x) \vee x \simeq i_1 \vee \ldots \vee x \simeq i_n \vee B$, *where the i_j's are constants and B is strictly flat.*

Theorem 4. *$\mathcal{B}_{\mathcal{A}}$ is $\mathcal{A}$-stable for $\mathcal{U}_{\succ}$.*

Note that the set $\mathcal{B}_{\mathcal{A}}$ does not contain only ground clauses, and testing the satisfiability of the set $\bar{S} \cup S'$ requires to rely on the ability of SMT-solvers to handle quantifiers (cf. Section 5 in [11]). The non-ground clauses in this set have the form

$$\mathrm{select}(a, x) \simeq \mathrm{select}(a', x) \vee x \simeq i_1 \vee \ldots \vee x \simeq i_n \vee B,$$

where B is strictly flat and ground. Borrowing the notation of [17, Definition 2] for so-called *partial equations*, this clause can be rewritten as $a \approx_D a' \vee B$, where D denotes the set $\{i_1, \ldots, i_k\}$. Such a clause states that under the guard $\neg B$, the *uninterpreted* terms $\mathrm{select}(a, x)$ and $\mathrm{select}(a', x)$ are identical for all $x \notin D$. Since arrays can be regarded as representing functions, and a term $\mathrm{select}(a, x)$ has the same intuitive meaning of $apply(a, x)$, the *uninterpreted* functions a and a' agree for all values of x except those in D.

5.5 Combination of records, integer offsets and arrays

We conclude by showing how the 2-theories scheme applies to combine these theories. In [2], the termination of $\mathcal{SP}$ on $\mathcal{T}$-satisfiability problems $\mathcal{T} \cup S$, where S is a set of flat unit clauses, is proved by analyzing all possible kinds of clauses in the limit $\mathcal{T} \cup \bar{S}$, generated by a fair derivation from $\mathcal{T} \cup S$. Let $\mathcal{T} \cup S$ be a problem in one of these three theories, either records, or integer offsets or arrays, and let $\mathcal{T} \cup \bar{S}$ be the corresponding limit. Based on those analyses, no clause in $\bar{S}$ paramodulates into a clause of either one of the two other presentations. In other words, $\bar{S}$ is disconnected from either one of the two other presentations[4]. We deduce that:

Theorem 5. *The 2-theories scheme is correct for any combination of the theories of records, integer offsets and arrays.*

[4] Technically, this follows from the fact that the theories are variable-inactive and share no function symbols.

6 Discussion

In this paper we introduced a decomposition framework that consists in using a generic theorem prover to "compile a theory away" from a $\mathcal{T}$-decision problem, before invoking an SMT-solver on the resulting problem. After presenting a set of sufficient conditions collectively termed *$\mathcal{T}$-stability*, that guarantee the correctness of this scheme, we showed how it applies to the superposition calculus and the theories of *records*, *integer offsets* and *arrays*. This rewrite-based approach to $\mathcal{T}$-decision problems is more efficient than one based on feeding the entire formula to theorem provers, that are not designed to handle its Boolean part efficiently, and it is simpler than a lazy approach that would require a tight integration of a theorem prover and a SAT-solver. The compilation stage can be viewed also as a *reduction*: in essence, our results *reduce* the theories of records and integer offsets to that of equality with uninterpreted functions (EUF). Thus, concrete $\mathcal{T}$-decision procedures can be implemented simply by interfacing a theorem prover implementing $\mathcal{SP}$, such as E [16], with a decision procedure for EUF, such as DPLL(=) (e.g., [13]). Similarly, the theory of arrays is reduced, as expected, to a theory of *partial functions*, with axioms stating that some uninterpreted functions are equal everywhere except on a given domain.

We also showed that this framework can be used to deal with combinations of theories under the *$\mathcal{T}$-disconnection* condition: if the theorem prover can handle all the theories in the problem, they can all be compiled away; if it cannot handle some of them, as it would be the case, for instance, with linear arithmetic or the theory of bitvectors (e.g., [9]), these are simply passed on to the SMT solver. In this scheme, the theorem prover plays the role of a procedure that reduces the SMT problems in a *uniform* manner. After such a *sifting* process, the SMT-solver only needs to solve "simpler" problems that are equisatisfiable to the original ones.

We are currently investigating how to adapt the techniques of [8] to devise a generic theory reasoner for a theory of *partial functions*, that may be generated by $\mathcal{SP}$ from a problem involving the theory of arrays. After integrating such a theory reasoner into an SMT-solver and interfacing the resulting tool with a theorem prover implementing $\mathcal{SP}$, we intend to run experiments to evaluate the efficiency of the system thus obtained.

For future work, we plan to investigate which other theories or inference systems satisfy the requirements of the decomposition framework. Indeed, the framework was presented in such a way to be as abstract as possible, so that it does not depend on a particular inference system. Thus, it is likely that more theories can be captured, with $\mathcal{SP}$ or other inference systems. Another direction for future research is to explore how to integrate automated model building methods into this framework, with the goal of designing SMT-tools capable of constructing both proofs and counterexamples, a highly desirable feature in applications. To this end, we intend to study how *hybrid* model building techniques (e.g., [15]), that try simultaneously to prove unsatisfiability or compute a model of a formula, could be applied within our approach.

References

1. 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. 5th FroCoS*, volume 3717 of *LNAI*, pages 65–80. Springer, 2005.
2. A. Armando, M. P. Bonacina, S. Ranise, and S. Schulz. New results on rewrite-based satisfiability procedures. *ACM Transactions on Computational Logic*, 2007. Accepted pending revision, February 2007, full version of [1], available at `http://profs.sci.univr.it/~bonacina/rewsat.html`.
3. A. Armando, S. Ranise, and M. Rusinowitch. A rewriting approach to satisfiability procedures. *Inf. Comput.*, 183(2):140–164, 2003.
4. 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, August 1999.
5. M. P. Bonacina and M. Echenim. On variable-inactivity and polynomial T-satisfiability procedures. Submitted.
6. M. P. Bonacina and M. Echenim. Rewrite-based satisfiability procedures for recursive data structures. In B. Cook and R. Sebastiani, editors, *Proc. 4th PDPAR Workshop, FLoC 2006*, ENTCS. Elsevier, 2007. To appear.
7. M. P. Bonacina and M. Echenim. Theory decision by decomposition. Technical Report RR 50/2007, Dipartimento di Informatica, Università degli Studi di Verona, 2007. Available at `http://profs.sci.univr.it/~echenim`.
8. A. R. Bradley, Z. Manna, and H. B. Sipma. What's decidable about arrays? In E. A. Emerson and K. S. Namjoshi, editors, *Proc. VMCAI-7*, volume 3855 of *LNCS*, pages 427–442. Springer, 2006.
9. D. Cyrluk, O. Möller, and H. Rueß. An efficient decision procedure for a theory of fixed-sized bitvectors with composition and extraction. Technical Report UIB96-08, Fakultät für Informatik, Universität Ulm, Ulm, Germany, 1996.
10. N. Dershowitz and D. A. Plaisted. Rewriting. In J. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 535–610. Elsevier Science Publishers, 2001.
11. D. L. Detlefs, G. Nelson, and J. B. Saxe. Simplify: a theorem prover for program checking. *J. ACM*, 52(3):365–473, 2005.
12. G. Nelson and D. C. Oppen. Simplification by cooperating decision procedures. *ACM TOPLAS*, 1(2):245–257, 1979.
13. 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, Nov. 2006.
14. R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In J. A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, pages 371–443. Elsevier and MIT Press, 2001.
15. N. Peltier. A calculus combining resolution and enumeration for building finite models. *Journal of Symbolic Computation*, 36(1-2):49–77, 2003.
16. S. Schulz. E – a brainiac theorem prover. *J. of AI Communications*, 15(2–3):111–126, 2002.
17. A. Stump, C. W. Barrett, D. L. Dill, and J. R. Levitt. A decision procedure for an extensional theory of arrays. In *LICS*, pages 29–37, 2001.
18. L. Zhang and S. Malik. The quest for efficient boolean satisfiability solvers. In A. Voronkov, editor, *Proc. CADE-18*, volume 2392 of *LNCS*, pages 295–313. Springer, 2002.