Journal of Symbolic Computation 45 (2010) 229–260

Contents lists available at ScienceDirect

Journal of Symbolic Computation

journal homepage: www.elsevier.com/locate/jsc

Theory decision by decomposition

Maria Paola Bonacina [a,1], Mnacho Echenim [b]

[a] *Dipartimento di Informatica, Università degli Studi di Verona, Strada Le Grazie 15, I-37134 Verona, Italy*

[b] *Laboratoire d'Informatique de Grenoble, Institut National Polytechnique de Grenoble, 46, Avenue Felix Viallet, F-38031 Grenoble, France*

ARTICLE INFO

Article history:
Received 2 April 2008
Accepted 31 October 2008
Available online 18 June 2009

Keywords:
Satisfiability modulo theories: Decision procedures, combination of theories
Automated theorem proving: Rewriting, superposition, paramodulation

ABSTRACT

The topic of this article is decision procedures for satisfiability modulo theories (SMT) of arbitrary quantifier-free formulæ. We propose an approach that *decomposes* the formula in such a way that its *definitional part*, including the theory, can be *compiled* by a rewrite-based first-order theorem prover, and the residual problem can be decided by an SMT-solver, based on the Davis–Putnam–Logemann–Loveland procedure. The resulting *decision by stages* mechanism may unite the complementary strengths of first-order provers and SMT-solvers. We demonstrate its practicality by giving decision procedures for the theories of *records*, *integer offsets* and *arrays*, with or without extensionality, and for combinations including such theories.

1. Introduction

Decision procedures are at the heart of formal verification tools, which invoke them to decide the validity of logical formulæ in decidable fragments of relevant theories. Software or hardware verification problems require us to decide validity *modulo* a theory $\mathcal{T}$, that is often a combination of theories, such as *linear arithmetic* and theories of *data structures*, on top of the theory of *equality*.[2] The problem of validity modulo $\mathcal{T}$ is called the $\mathcal{T}$*-decision problem*, or, dually, *SMT problem*, where SMT stands for satisfiability modulo theories. Its decision procedures are called $\mathcal{T}$*-decision procedures* and the reasoners implementing them *SMT-solvers*.

E-mail addresses: mariapaola.bonacina@univr.it (M.P. Bonacina), mnacho.echenim@imag.fr (M. Echenim).
URLs: http://profs.sci.univr.it/~bonacina/ (M.P. Bonacina), http://equipes-lig.imag.fr/capp/members.php (M. Echenim).
1 Fax: +39 045 802 7068.

2 Also known as *EUF* for *equality with uninterpreted function symbols*, where "uninterpreted" means "free," as opposed to "interpreted" or "definite" symbols, whose interpretation is restricted to the models of the given theory.

0747-7171/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsc.2008.10.008

In foreseeable application scenarios, an SMT-solver acts as the *back-end reasoner* for another system, such as an interactive proof assistant or program analyzer, or a model-checker in the context of counterexample-guided abstraction refinement (Henzinger et al., 2002). Since each verification session may generate many and very large formulæ, the SMT-solver, and therefore its $\mathcal{T}$-decision procedures, must be *efficient* and *scalable*. On the other hand, they should be *expressive*, to handle the required combinations of theories, capable to generate *justifications* for their answers (e.g., proofs, unsatisfiable cores, counter-models), as well as *sound* and *complete* to avoid misleading results. The challenge of meeting all these requirements and the importance of the underlying applications have motivated a significant body of research, that we summarize shortly.

1.1. State of the art: DPLL-based SMT-solvers

Due to the large boolean structure of formulæ, the core of most state-of-the-art SMT-solvers is a *SAT-solver* implementing the *Davis–Putnam–Logemann–Loveland procedure* for propositional satisfiability (SAT), originated in Davis and Putnam (1960) and Davis et al. (1962) and developed by many authors.[3]

The *eager approach* reduces the $\mathcal{T}$-decision problem to a SAT-problem and applies the SAT-solver, taking advantage of its extreme efficiency. Combination of theories is not an issue, since all get reduced to propositional logic. Two drawbacks are the loss of problem structure and the space complexity of the reduction, which relates the size of the resulting formula to that of the original one. Even a quadratic reduction, which may sound efficient in theory, may be problematic in practice, since the size of relevant formulæ is of the order of megabytes. Thus, much research in this direction concentrates on making the reduction as space-efficient as possible. References include (Jackson and Vaziri, 2000; Bryant and Velev, 2001; Bryant et al., 2002; Seshia et al., 2003; Meir and Strichman, 2005).

The *lazy approach* integrates the SAT-solver with a *theory solver*, or $\mathcal{T}$-solver, in such a way that while the SAT-solver searches for a model of the formula, the $\mathcal{T}$-solver ensures that the propositional model, represented by a conjunction of literals, is also a $\mathcal{T}$-model. A key issue is the interplay of *propositional reasoning* and *theory reasoning*. At one extreme of the spectrum, the SAT-solver generates a propositional model before calling the $\mathcal{T}$-solver to check it, hence the "laziness" of the approach. The well-known downside is the work wasted by the SAT-solver pursuing a candidate model that is not a $\mathcal{T}$-model. At the other extreme, the SAT-solver invokes the $\mathcal{T}$-solver at every truth assignment. The symmetric downside is the repetitious work done by the $\mathcal{T}$-solver. Many authors investigated finding the best trade-off between propositional and theory reasoning, including Zhang (2000), Barrett et al. (2002), de Moura et al. (2002), Armando et al. (2004), Bozzano et al. (2005) and Lahiri and Musuvathi (2005). This line of research has led to the design of tight integration schemes, where the $\mathcal{T}$-solver works *incrementally*, in order to avoid repetitions, produces *minimal conflict sets* or *unsatisfiable cores*, to justify rejecting a candidate model and guide the search of the SAT-solver, and generates lemmas entailed by the formula in the theory. A systematic theoretical treatment was given in Nieuwenhuis et al. (2006). Another general presentation appeared in Sebastiani (2007).

A crucial issue is that $\mathcal{T}$ is usually a combination of theories $\mathcal{T}_1 \ldots \mathcal{T}_n$. Most SMT-solvers resort to the method of Nelson and Oppen (1979) to combine decision procedures, or one of its extensions,[4] so that the $\mathcal{T}$-solver is a Nelson–Oppen combination of n $\mathcal{T}_k$-solvers for $1 \leq k \leq n$. The Nelson–Oppen method lets each decision procedure propagate equalities between variables[5] to the others. This is sufficient if equality is the only shared symbol, and each $\mathcal{T}_k$ is *convex*, that is, whenever $\mathcal{T}_k \models H \rightarrow \bigvee_{i=1}^{m} u_i \simeq v_i$, where H is a conjunction of atoms, then $\mathcal{T}_k \models H \rightarrow u_j \simeq v_j$, for some j, $1 \leq j \leq m$. If a theory is not convex, its decision procedure needs to propagate disjunctions of equalities. This generalization is sufficient, if each theory $\mathcal{T}_k$ is *stably infinite*, that is, any $\mathcal{T}_k$-satisfiable quantifier-free formula admits a

[3] See Zhang and Malik (2002) for a survey.

[4] See Ghilardi et al. (2005) for a survey.

[5] They are constants, logically, or variables that are implicitly existentially quantified, can be replaced by (Skolem) constants, and are treated like constants by the algorithm, but it is traditional to call them variables.

model with a domain of infinite cardinality. Propagation of disjunctions is implemented through case analysis and backtracking. Since it would be inefficient to let the $\mathcal{T}$-solver and the SAT-solver perform two distinct backtracking searches, and the SAT-solver is generally faster, the idea is to entrust the SAT-solver also with the case analysis for the $\mathcal{T}$-solver. In *delayed theory combination*, the combination of theories happens in the SAT-solver, which has all possible equalities between variables generated upfront as proxy boolean variables, and invokes directly the n $\mathcal{T}_k$-solvers (Bozzano et al., 2006). If this encumbers the SAT-solver too much, one may resort to *splitting on demand*, that lets the $\mathcal{T}_k$-solvers generate equalities and pass their proxies to the SAT-solver as lemmas (Barrett et al., 2006), as experienced first in the Yices solver (Dutertre and de Moura, 2006). A general framework for Nelson–Oppen combination in the context of DPLL was given recently in Krstić and Goel (2007). *Model-based combination* assumes that each $\mathcal{T}_k$-solver maintains a candidate model and lets it propagate equalities between variables, that are true in the candidate model, regardless of whether they are entailed: such assertions will be undone upon backtracking if they generate conflicts (de Moura and Bjørner, 2008b).

In most cases, the decidable fragment is the *quantifier-free fragment*, so that the $\mathcal{T}$-solver does not need to reason about quantifiers. However, problems with quantified variables do arise in applications, and $\mathcal{T}$-solvers approach them by instantiating the variables based on heuristics (Detlefs et al., 2005; Ge et al., 2007), enhanced with indexing techniques to accelerate instance generation (de Moura and Bjørner, 2007).

1.2. State of the art: First-order theorem provers as SMT-solvers

If the need to reason in propositional logic, and specifically with huge clauses, suggested DPLL-based solvers, the need to reason with equality suggested first-order theorem provers, where equality reasoning is based on *rewriting*, as done empirically for instance in Arkoudas et al. (2004) and Bouillaguet et al. (2007). The first-order presentation of the theory is part of the input to the prover and a combination of theories is given by the union of their presentations.

Theory reasoning can be seen as reasoning in a *hierarchy* of successive *extensions* of theories. One approach is to view functions in the extension as *partial* with respect to the base theory: then, an inference system for first-order logic with equality (FOL$+=$) can be modified to reason about partial functions (Ganzinger et al., 2006), or be integrated with decision procedures for special theories such as linear arithmetic (Waldmann and Prevosto, 2006). If the extension is *local*, it is sufficient to reason with finitely many ground instances of its axioms, so that reasoning in the extended theory is reduced to reasoning in the combination of the base theory and EUF (Sofronie-Stokkermans, 2005). An implementation of local reasoning by integrating instance generation, as in instance-based theorem proving, with an SMT-solver was proposed in Jacobs (2008).

A refutationally complete theorem prover is guaranteed to terminate only if the input is unsatisfiable. In order to have *$\mathcal{T}$-decision procedures*, it is necessary to prove *termination*, regardless of the satisfiability of the input. Several such results were obtained in Armando et al. (2003), Armando et al. (2009) and Bonacina and Echenim (2007b) for *$\mathcal{T}$-satisfiability problems*, that is, $\mathcal{T}$-decision problems where the formula is a set of ground unit clauses. It was proved that a standard, superposition-based inference system, named $\mathcal{SP}$, paired with a fair search plan, is guaranteed to terminate on any $\mathcal{T}$-satisfiability problem in several theories of data structures, including *records*, *integer offsets* and *arrays*, and therefore it represents a $\mathcal{T}$-satisfiability procedure. The experiments reported in Armando et al. (2009) showed that a theorem prover would perform well, compared with state-of-the-art SMT-solvers.

Combination of theories was addressed in Armando et al. (2009) by identifying a sufficient condition, termed *variable-inactivity*, for *modularity of termination*: if $\mathcal{SP}$ terminates on $\mathcal{T}_k$-satisfiability problems and each $\mathcal{T}_k$ is variable-inactive, then $\mathcal{SP}$ also terminates on $\mathcal{T}$-satisfiability problems for $\mathcal{T} = \bigcup_{k=1}^{n} \mathcal{T}_k$. All theories in Armando et al. (2003), Armando et al. (2009) and Bonacina and Echenim (2007b) are variable-inactive and variable-inactivity implies *stable-infiniteness* (Bonacina et al., 2006). Further results on the rewrite-based approach were obtained by using a *meta-saturation procedure* to determine complexity bounds (Lynch and Tran, 2007) and sufficient conditions for stable-infiniteness and other properties (Kirchner et al., 2006).

1.3. Problem statement, technical motivation and contributions

In this article we study the problem of extending the rewrite-based approach from $\mathcal{T}$-satisfiability problems to $\mathcal{T}$-decision problems given by sets of ground clauses. We achieve this goal by an approach that unites the strengths of DPLL-based SMT-solvers and rewrite-based provers, and therefore contributes to advance the long-term goal of combining *expressivity*, *soundness* and *completeness*, with *efficiency* and *scalability*, as well as the capacity to generate justifications, in $\mathcal{T}$-decision procedures.

First-order provers offer *expressivity*, since their language is full first-order logic with equality, *soundness* and *completeness* of the inference system, the native capability to *generate proofs*, and a basis for *theory-independent model building*, starting from the finite saturated set produced by a terminating derivation from a satisfiable input.[6] They excel at reasoning with non-ground unit equalities, by matching and rewriting, and with non-ground clauses, by using unification to instantiate variables. On the other hand, they were not designed to focus on propositional efficiency in non-Horn theories[7] and certainly not to break the huge disjunctions of verification problems. Nor were they conceived to reason in theories, such as *linear arithmetic* or *bitvectors*, that require computing and solving rather than deducing. Dually, DPLL-based SMT-solvers offer *efficiency* and *scalability*; they can generate proofs[8] and sometimes models, at least for some theories (de Moura and Bjørner, 2008b). SMT-solvers are strong at propositional reasoning and embedding special theories (e.g., (Bruttomesso, 2008) for bitvectors and (Dutertre and de Moura, 2006) for linear arithmetic), but they were not born to reason about quantifiers and non-ground equalities. Since generic provers and SMT-solvers appear to complement each other, we propose an approach to let them work together.

We begin by observing that an SMT-problem can be *decomposed* into:

- A *definitional part*, that interacts with the theory, and
- An *operational part* that contains the boolean structure of the formula.

For instance, the definitional part includes the presentation of the theory and unit equalities that "define" functions, whereas the operational part includes non-unit clauses, hence the disjunctions. Then the idea is to solve an SMT-problem by extracting first the needed information from the definitional part, and then using it to decide the satisfiability of the operational part. The first task is performed by a first-order theorem prover, which *compiles* the definitional part, performing as much theory reasoning as possible. The second task is performed by an SMT-solver, which decides satisfiability of the residual problem. An SMT-problem is thus *solved by stages*, by *pipelining* prover and solver, in such a way that each one does what it is best at: the prover reasons about equality and universally quantified variables from the axioms, while the solver reasons about propositional structure (especially disjunction) and ground equality.

The realization is far from trivial, because we need to ensure that the scheme is correct and the prover generates something that the solver can receive. We develop a framework of sufficient conditions, collectively termed $\mathcal{T}$*-stability*, to ensure that satisfiability is preserved through theory compilation. This framework is fully abstract, because it does not even assume a specific inference system. Then, we show how it can be instantiated to define $\mathcal{T}$*-decision procedures by stages* for the theories of *records with or without extensionality*, *integer offsets* and *arrays with or without extensionality*, using the inference system $\mathcal{SP}$. Combination of theories is included throughout: not only do decomposition and decision by stages apply also to a combination of theories, but they allow one to *postpone* reasoning about a theory, that is, compile one first and then deal with another.

Theory compilation can be viewed also as *problem reduction*. However, since it is a reduction achieved by generic inferences, it is proof-theoretic in nature and therefore *theory-independent*, unlike model-theoretic reductions. In perspective, one would hope that this "division of labour" between

6 See Caferra et al. (2004) for an introduction to automated model building.

7 See for instance the analyses already in Plaisted and Zhu (1997).

8 For instance, DPLL-based SMT-solvers can generate proofs by a translation of their steps into resolution-style inferences. (Personal communication of Leonardo de Moura to the first author, May 2008.)

first-order prover and DPLL-based solver may benefit experimenters and developers, allowing them to use existing tools and making the design of SMT-solvers simpler and less ingenious.

1.4. Further comparison with related work

We began investigating the issue of extending the rewrite-based approach from $\mathcal{T}$-satisfiability problems to $\mathcal{T}$-decision problems in Bonacina and Echenim (2007a). In that paper, we showed that if theory $\mathcal{T}$ is *subterm-inactive*, then $\mathcal{SP}$ is guaranteed to terminate also on $\mathcal{T}$-decision problems. Subterm-inactivity implies variable-inactivity, and several theories of data structures are subterm-inactive. However, other theories of interest, such as those of *lists* or *records*, are variable-inactive, but not subterm-inactive, an the conditions for subterm-inactivity are very complicated. Thus, we sought a simpler and more general solution, by proving in Bonacina and Echenim (2008) that variable-inactivity suffices: if $\mathcal{T}$ is variable-inactive and such that an $\mathcal{SP}$-strategy is a $\mathcal{T}$-satisfiability procedure, then the $\mathcal{SP}$-strategy is also a $\mathcal{T}$-decision procedure. This approach is different from the one presented here, because in Bonacina and Echenim (2008) the $\mathcal{SP}$-strategy is applied to the whole problem, and there is no combination of first-order prover and SMT-solver. Thus, the method of Bonacina and Echenim (2008) can be considered a "pure" extension of the rewrite-based approach to $\mathcal{T}$-decision problems, in the sense that all the reasoning is done by a first-order rewrite-based prover. The limitation of such an approach is the aforementioned weakness of first-order provers in dealing with huge propositional disjunctions and theories such as linear arithmetic or bitvectors. It was precisely to address this weakness that we embarked in studying how to combine first-order rewrite-based reasoners with SMT-solvers.

An early, mostly empirical combination of a rewrite-based prover, applied as $\mathcal{T}$-satisfiability procedure, with a SAT-solver, based on ordered binary decision diagrams, appeared in Déharbe and Ranise (2003) and Ranise and Déharbe (2003). More recently, de Moura and Bjørner (2008a) proposed an approach, where the $\mathcal{SP}$-engine is one of the inner satellite solvers of the DPLL-based core engine. A fundamental difference between all such methods and ours is that embedding the $\mathcal{SP}$-engine as a satellite solver requires a tight integration, where the $\mathcal{SP}$-engine is invoked multiple times, incurring in the issue of avoiding repetitious work, already illustrated in Section 1.1. In our approach, first-order prover and SMT-solver can be taken almost "off the shelf," with minimal integration effort, and each reasoner is applied *only once*.

A technical similarity between the method in de Moura and Bjørner (2008a) and ours is the common intuition that ground, non-unit clauses should be left to the SMT-solver. However, another major difference is that the method of de Moura and Bjørner (2008a) yields semi-decision procedures: termination is obtained only by imposing a bound on inference depth and forbidding the generation of clauses that require more inferences.[9] Our approach yields decision procedures, because first-order prover and SMT-solver cooperate in such a way that refutational completeness and termination are preserved, without restrictions on inferences.

Outline of the paper. After a summary of basic definitions in Section 2, Section 3 presents the *decomposition principle* and the abstract framework for $\mathcal{T}$*-stability*. In Section 4 the abstract principle is realized in the *decision by stages* scheme using $\mathcal{SP}$. Section 5 applies decision by stages to *records*, *integer offsets* and *arrays*. Since the latter theory would require us to feed to the SMT-solver a non-ground problem, Section 6 shows how this can be avoided by giving a decision procedure for *theories of partial functional equations*, that reduces the problem to ground equality. In Section 7 the treatment of *combination of theories* of Sections 3 and 4 is exemplified with combinations including the above theories.

2. Preliminaries

Given a first-order signature Σ, we assume the standard definitions of Σ-terms, Σ-atoms, Σ-literals, Σ-clauses and Σ-sentences. As usual, clauses are variable-disjoint. For notation, $\simeq$ is

[9] Actually, the bound on inference depth is needed also for refutational completeness, if the $\mathcal{SP}$-engine is barred from seeing ground, non-unit clauses. (Personal communication of Leonardo de Moura to the first author, September 2008.)

unordered equality, $\bowtie$ is either $\simeq$ or $\not\simeq$, while $=$ is identity. Lower-case Latin letters l, r, u, v, t denote generic terms, while we typically reserve w, x, y, z for variables and a, b, c, d, e, i for constants; other lower-case letters in the signature denote function symbols. Lower-case Greek letters are used for substitutions, with postfix notation. Upper-case Latin letters C, D, E, F, Q denote clauses, or multisets of literals interpreted as disjunctions; L is used for literals, A, S, P, G for sets of clauses, and $\mathcal{B}$ for sets of clauses with special properties. More notation may be introduced as needed.

Definition 1. A theory is defined by a signature Σ and a set $\mathcal{T}$ of Σ-sentences, called its *presentation* or *axiomatization*.

For simplicity, we omit Σ whenever possible, using the presentation $\mathcal{T}$ to identify the theory, and knowing that its signature Σ is that of $\mathcal{T}$.

Definition 2. A *$\mathcal{T}$-satisfiability problem* is the problem of deciding whether a set of ground unit clauses is satisfiable in $\mathcal{T}$, or has a $\mathcal{T}$-model. A *$\mathcal{T}$-decision problem*, or *SMT problem*, is the problem of deciding whether a ground formula is satisfiable in $\mathcal{T}$.

Without loss of generality, we assume that the ground formula of an SMT problem is a set of clauses P, and sentences in $\mathcal{T}$ are reduced to clausal form, so that an SMT problem $\mathcal{T} \cup P$ is a set of clauses. For sets of clauses S and S', we write $S \equiv S'$ if they are *logically equivalent* (every model of S is a model of S' and vice versa) and $S \equiv_s S'$ if they are *equisatisfiable* (S has a model if and only if S' has a model).

2.1. Inference systems

Standard inference systems for first-order logic with equality employ well-founded orderings on terms, literals and clauses, that assume a *precedence*, or a partial order on the symbols of the signature. A *simplification ordering* $\succ$ is *stable*, *monotonic* and contains the *subterm ordering*: if $l \succ r$, then $t[l]\sigma \succ t[r]\sigma$ for any context t and substitution σ, and if r is a proper subterm of t then $t \succ r$. Simplification orderings are well-founded. A *complete simplification ordering*, or CSO, is also total on ground terms. Definitions and results about orderings can be found, for instance, in Dershowitz and Plaisted (2001). We say that a CSO is *good*, if $t \succ c$, whenever t is a compound term and c a constant. Throughout this article we consider only good orderings.

An *inference system* Inf consists of a set of *inference rules*, distinguished into *expansion rules* that generate new clauses and *contraction rules* that replace or delete clauses. An inference rule is *unary* if it has one premise, *binary* if it has two premises, and so on. If the inference system is based on a CSO, we write $\text{Inf}_\succ$ for Inf equipped with $\succ$. A well-founded ordering provides the basis for a notion of *redundancy*: a ground clause C is redundant in S if $\{D_1, \ldots D_k\} \models C$ and $D_i \prec C$ for all i, $1 \leq i \leq k$, for ground instances $\{D_1, \ldots D_k\}$ of clauses in S; a clause is redundant if all its ground instances are. An inference is redundant if it uses or generates a redundant clause. A set of clauses is *saturated* if all expansion inferences in the set are redundant.

A *derivation* is a sequence $S_0 \vdash_{\text{Inf}_\succ} S_1 \vdash_{\text{Inf}_\succ} \ldots S_i \vdash_{\text{Inf}_\succ} \ldots$, where each S_i is a set of clauses, derived by applying an inference rule to clauses in S_{i-1}. The *limit* of 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, for some $n \geq 0$, we write $S_0 \vdash^n_{\text{Inf}_\succ} S_n$. An inference system is *refutationally complete* if there exist derivations $S_0 \vdash^n_{\text{Inf}_\succ} S_n$ with $\Box \in S_n$, whenever S_0 is unsatisfiable. A derivation is *fair* if all expansion inferences become redundant eventually, and a fair derivation generates a *saturated* limit.

A *strategy* $\mathfrak{S}$ is given by an inference system and a *search plan*, that controls the application of the inference rules, and determines the unique derivation $S_0 \vdash_{\mathfrak{S}} S_1 \vdash_{\mathfrak{S}} \ldots S_i \vdash_{\mathfrak{S}} \ldots$ generated from a given S_0. A search plan is *fair* if it only generates fair derivations, and a strategy with a fair search plan is also called fair. A strategy $\mathfrak{S}$ is *complete*, if it has a refutationally complete inference system and a fair search plan, so that $S_0 \vdash^n_{\mathfrak{S}} S_n$ with $\Box \in S_n$, whenever S_0 is unsatisfiable. A strategy with inference system $\text{Inf}_\succ$ is an $\text{Inf}_\succ$-strategy.

Depending on the strategy, the *state* of a derivation may be a tuple, rather than a single set S_i. Supported strategies allow one to separate the axioms of $\mathcal{T}$ from the other clauses. A *supported strategy* works on *pairs* of sets of clauses $(\mathcal{T}; S)$, where S is the *set-of-support*, and applies inferences in such a

Superposition	$\dfrac{C \vee l[u'] \simeq r \quad D \vee u \simeq t}{(C \vee D \vee l[t] \simeq r)\sigma}$	$(i), (ii), (iii), (iv)$
Paramodulation	$\dfrac{C \vee l[u'] \not\simeq r \quad D \vee u \simeq t}{(C \vee D \vee l[t] \not\simeq r)\sigma}$	$(i), (ii), (iii), (iv)$
Reflection	$\dfrac{C \vee u' \not\simeq u}{C\sigma}$	(v)
Equational Factoring	$\dfrac{C \vee u \simeq t \vee u' \simeq t'}{(C \vee t \not\simeq t' \vee u \simeq t')\sigma}$	$(i), (vi)$

where the notation $l[u']$ means that u' appears as a subterm in l, σ is the most general unifier (mgu) of u and u', u' is not a variable in *Superposition* and *Paramodulation*, and the following abbreviations hold:

(i): $u\sigma \not\preceq t\sigma$;

(ii): $\forall L \in D : (u \simeq t)\sigma \not\preceq L\sigma$;

(iii): $l[u']\sigma \not\preceq r\sigma$;

(iv): $\forall L \in C : (l[u'] \bowtie r)\sigma \not\preceq L\sigma$;

(v): $\forall L \in C : (u' \not\simeq u)\sigma \not\prec L\sigma$;

(vi): $\forall L \in \{u' \simeq t'\} \cup C : (u \simeq t)\sigma \not\prec L\sigma$.

In standard terminology, $D \vee u \simeq t$ *paramodulates into* $C \vee l[u'] \not\simeq r$, which is the clause *paramodulated into*, while $D \vee u \simeq t$ is the clause *paramodulated from*. We adopt this terminology also for superposition.

Fig. 1. Expansion inference rules of $\mathcal{SP}$: in expansion rules, what is below the inference line is added to the clause set that contains what is above the inference line.

way that all expansion inferences have at least one premise in the set-of-support and all clauses thus generated are added to the set-of-support. An inference system is *refutationally complete for supported strategies*, if there exist derivations $(\mathcal{T}; S_0) \vdash^n_{\mathrm{Inf}_\succ} (\mathcal{T}; S_n)$ with $\Box \in S_n$, whenever $\mathcal{T} \cup S_0$ is unsatisfiable and $\mathcal{T}$ is consistent. A survey of strategies can be found in Bonacina (1999) (cf. Section 2.6 for supported strategies).

We assume a supported $\mathrm{Inf}_\succ$-strategy $\mathfrak{S}$, such that $\mathrm{Inf}_\succ$ is refutationally complete for supported strategies and $\mathfrak{S}$ is fair, so that $(\mathcal{T}; S_0) \vdash^n_{\mathfrak{S}} (\mathcal{T}; S_n)$ with $\Box \in S_n$, whenever $\mathcal{T} \cup S_0$ is unsatisfiable and $\mathcal{T}$ is consistent. Furthermore, all inference rules in $\mathrm{Inf}_\succ$ are either unary or binary.

In the second part of the paper, $\mathrm{Inf}_\succ$ will be replaced by a concrete inference system, the *superposition calculus* $\mathcal{SP}$, whose expansion and contraction rules are listed in Figs. 1 and 2, respectively. It is well known that $\mathcal{SP}$ is refutationally complete (e.g., (Nieuwenhuis and Rubio, 2001)) and refutationally complete for supported strategies whenever $\mathcal{T}$ is saturated.

2.2. Flattening

$\mathcal{SP}$ assumes that equality is the only predicate, or non-equational atoms (e.g., $P, P(\bar{t})$) are translated into equations (e.g., $p \simeq \mathit{true}$, $p(\bar{t}) \simeq \mathit{true}$), introducing disjoint sorts for ordinary terms and boolean terms. Since flattening will be used in combination with $\mathcal{SP}$, this subsection assumes that equality is the only predicate. For a term t, the *depth* of t is $\mathrm{depth}(t) = 0$, if t is a constant or a variable, and $\mathrm{depth}(f(t_1, \ldots, t_n)) = 1 + \max\{\mathrm{depth}(t_i) \mid 1 \leq i \leq n\}$, otherwise. For literals, we define $\mathrm{depth}(l \bowtie r) = \mathrm{depth}(l) + \mathrm{depth}(r)$.

Strict Subsumption	$\dfrac{C \quad D}{C}$	$D \succcurlyeq C$
Simplification	$\dfrac{C[u] \quad l \simeq r}{C[r\sigma] \quad l \simeq r}$	$u = l\sigma,\ l\sigma \succ r\sigma,\ C[u] \succ (l \simeq r)\sigma$
Deletion	$\dfrac{C \vee t \simeq t}{}$	

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

Fig. 2. Contraction inference rules of $\mathcal{SP}$: in contraction rules, what is above the double inference line is removed from the clause set and what is below the double inference line is added to the clause set.

Definition 3. A literal is *flat* if its depth is at most 1 and *strictly flat* if its depth is 0. A clause is *flat* if all its positive literals are flat and all its negative literals are strictly flat; a clause is *strictly flat* if all its literals are.

Goodness of the ordering implies the following:

Proposition 4. *If a ground clause C contains a maximal literal whose maximal term is a constant, then C is strictly flat.*

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

- Σ' is obtained by adding a finite number of new constant symbols to Σ,
- Every non-unit clause in S_F is strictly flat,
- Every unit clause in S_F is flat, and
- For all presentations $\mathcal{T}$: $\mathcal{T} \cup S \equiv_s \mathcal{T} \cup S_F$.

For example, flattening $S = \{f(a) \not\simeq f(b) \vee f(a) \not\simeq f(c)\}$ gives $S_F = \{f(a) \simeq a',\ f(b) \simeq b',\ f(c) \simeq c',\ a' \not\simeq b' \vee a' \not\simeq c'\}$, where a', b' and c' are new constants.[10]

3. Theory compilation

The *decomposition principle* to solve SMT problems can be stated as follows:

(1) Decompose the SMT problem into a *definitional part* and an *operational* part,
(2) Extract the needed information from the definitional part, and
(3) Use this information to test the satisfiability of the operational part.

In other words, in order to decide an SMT problem $\mathcal{T} \cup S \cup S'$, where S and S' represent the definitional and operational parts, respectively, we test the satisfiability of $\bar{S} \cup S'$, where $\bar{S}$ represents information extracted from S and $\mathcal{T}$. The idea is that $\bar{S}$ does not include $\mathcal{T}$, so that the theory has been *compiled away*, so to speak.

Correctness of this approach requires that $\mathcal{T} \cup S \cup S' \equiv_s \bar{S} \cup S'$. In order to establish equisatisfiability, we proceed in two phases, covered in this one and Section 4, respectively. In this section, we address a more general question:

Given presentation $\mathcal{T}$ and sets of clauses S, S', A and A' such that $\mathcal{T} \cup S \equiv_s A$ and $\mathcal{T} \cup S' \equiv_s A'$, how do we guarantee that $\mathcal{T} \cup S \cup S' \equiv_s A \cup A'$?

10 Many such transformations appear in the literature on equational reasoning and decision procedures, the oldest we are aware of is in Brand (1975).

The question is not trivial, because equisatisfiability is not preserved by taking unions. We provide an answer by introducing the condition of $\mathcal{T}$-*congruity*[11] with respect to the generic inference system $\mathrm{Inf}_{\succ}$ of $\mathfrak{S}$. $\mathcal{T}$-congruity ensures equisatisfiability, provided $\mathcal{T}$-congruity is preserved by inferences, requirement that leads us in turn to $\mathcal{T}$-*stability*. In Section 4, we will show how to generate $\mathcal{T}$-congruous sets and obtain *decision procedures by stages*, by instantiating the abstract framework of this section for the concrete inference system $\mathcal{SP}$.

3.1. $\mathcal{T}$-congruity

We start by observing that answering our question by imposing $A \equiv \mathcal{T} \cup S$ and $A' \equiv \mathcal{T} \cup S'$ would be too strong a requirement, satisfied also by the trivial choice of taking $A = \mathcal{T} \cup S$ and $A' = \mathcal{T} \cup S'$. Thus, we relax the condition of logical equivalence by 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 something weaker: namely, that A entails the set of $\mathcal{T}$-*descendants of* S and A' entails the set of $\mathcal{T}$-*descendants of* S'. In order to define $\mathcal{T}$-descendants, we need the following preliminary:

Definition 5. A clause D is *generated from parents* C and C' if it is generated by a binary expansion rule applied to C as first premise and C' as second premise. C and C' may be called *first* and *second parent*, respectively.

To keep the treatment in this section fully generic, we do not assume specific inference rules except in examples. For the sake of the examples, it is simple to instantiate Definition 5 for common expansion rules:

Example 6. For paramodulation and superposition, the first parent is the *clause paramodulated from* and the second parent is the *clause paramodulated into*.

We can now define $\mathcal{T}$-descendants:

Definition 7. The set of $\mathcal{T}$-*children of a clause* C is defined by:

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

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

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

Then, the sets of $\mathcal{T}$-*descendants of* C and $\mathcal{T}$-*descendants of* S, where S is a set of clauses, are defined by:

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

Note that $\mathcal{T} \cup S \models \mathcal{D}(S, \mathcal{T})$ *regardless of the inference system*, provided it is sound.

Example 8. Let $\mathcal{T} = \{\mathsf{p}(\mathsf{s}(x)) \simeq x;\ \mathsf{s}(\mathsf{p}(x)) \simeq x;\ \mathsf{s}(x) \not\simeq x;\ \mathsf{s}(\mathsf{s}(x)) \not\simeq x\}$, and suppose $\mathrm{Inf}_{\succ}$ features paramodulation. Then a clause $C = \mathsf{p}(a) \simeq b$ paramodulates only into $\mathsf{s}(\mathsf{p}(x)) \simeq x$, generating $F = \mathsf{s}(b) \simeq a$. Thus, the set of $\mathcal{T}$-children of C is $\mathcal{G}(C, \mathcal{T}) = \{\mathsf{s}(b) \simeq a\}$. In turn, F paramodulates into three of the axioms of $\mathcal{T}$, hence the set of $\mathcal{T}$-children of F is $\{\mathsf{p}(a) \simeq b;\ a \not\simeq b;\ \mathsf{s}(a) \not\simeq b\}$. The set of $\mathcal{T}$-descendants of C is $\mathcal{D}(C, \mathcal{T}) = \{\mathsf{s}(b) \simeq a;\ \mathsf{p}(a) \simeq b;\ a \not\simeq b;\ \mathsf{s}(a) \not\simeq b\}$.

We can then define the desired property of entailing $\mathcal{T}$-descendants, with the intuition that it be weaker than logical equivalence, but stronger than equisatisfiability:

Definition 9. A set of clauses A is $\mathcal{T}$-*congruous* with a clause C if $A \models \mathcal{D}(C, \mathcal{T})$; it is $\mathcal{T}$-*congruous* with a set S if $A \models \mathcal{D}(S, \mathcal{T})$.

[11] This property was named $\mathcal{T}$-*compatibility* in Bonacina and Echenim (2007c) and was renamed in this article because $\mathcal{T}$-*compatibility* already has other meanings in the literature.

All the notions in this section, beginning with $\mathcal{T}$-descendants and $\mathcal{T}$-congruity, are parametric with respect to $\mathrm{Inf}_\succ$: for example, A may be $\mathcal{T}$-congruous with S for an inference system, but not for another one. For the sake of generality and readability, we let this dependence remain implicit.

Example 10. Assume that $\mathcal{T} = \{h(g(x), g(x)) \simeq a\}$, $C = g(a) \simeq b$ and $\mathrm{Inf}_\succ$ features paramodulation. The sets of $\mathcal{T}$-children and $\mathcal{T}$-descendants of C are, respectively, $\mathcal{G}(C, \mathcal{T}) = \{h(b, g(a)) \simeq a,\ h(g(a), b) \simeq a\}$ and $\mathcal{D}(C, \mathcal{T}) = \{g(a) \simeq b,\ h(b, g(a)) \simeq a,\ h(g(a), b) \simeq a,\ h(b, b) \simeq a\}$. Let $A = \{g(a) \simeq b,\ h(b, b) \simeq a\}$. Since $A \models \mathcal{D}(C, \mathcal{T})$, A is $\mathcal{T}$-congruous with C.

Proposition 11. *Let S, S', A and A' be sets of clauses such that A is $\mathcal{T}$-congruous with S and A' with S'. Then the following properties hold:*

(1) *$A \cup A'$ is $\mathcal{T}$-congruous with $S \cup S'$;*
(2) *If $(\mathcal{T}; S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}; S \cup \{D\})$, where D is generated from first parent in S and second parent in $\mathcal{T}$, then A is $\mathcal{T}$-congruous with $S \cup \{D\}$.*

Proof. For item (1), since $A \cup A' \models \mathcal{D}(S, \mathcal{T})$ and $A \cup A' \models \mathcal{D}(S', \mathcal{T})$, it follows that $A \cup A' \models \mathcal{D}(S \cup S', \mathcal{T})$. Item (2) is obvious: since $D \in \mathcal{D}(S, \mathcal{T})$, we have $\mathcal{D}(S \cup \{D\}, \mathcal{T}) = \mathcal{D}(S, \mathcal{T})$. □

The property of *$\mathcal{T}$-disconnection* captures the notion that a set of clauses does not interact with $\mathcal{T}$:

Definition 12. A set of clauses S is *$\mathcal{T}$-disconnected* if no binary inference applies to a premise in S and one in $\mathcal{T}$. If $S = \{C\}$, we may also write that C is $\mathcal{T}$-disconnected.

If S is $\mathcal{T}$-disconnected, $\mathcal{D}(S, \mathcal{T}) = S$ and $S \models \mathcal{D}(S, \mathcal{T})$ holds trivially, so that:

Proposition 13. *If S is $\mathcal{T}$-disconnected then it is $\mathcal{T}$-congruous with itself.*

3.2. Preservation of $\mathcal{T}$-congruity

$\mathcal{T}$-congruity will be sufficient to ensure that $\mathcal{T} \cup S \cup S' \equiv_s A \cup A'$, provided all inference rules of $\mathrm{Inf}_\succ$ preserves it. This section is devoted to identify sufficient conditions to guarantee that if A is $\mathcal{T}$-congruous with S and $(\mathcal{T}; S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}; S')$, then A is also $\mathcal{T}$-congruous with S'. We consider, in the order:

- Contraction inferences,
- Unary expansion inferences,
- Binary expansion inferences between a clause in $\mathcal{T}$ and a clause in S,
- Binary expansion inferences within S.

3.2.1. Contraction inferences

Definition 14. A set of clauses S is *$\mathcal{T}$-contraction-safe*, if for all contraction inferences $(\mathcal{T}; S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}; S')$ and sets A $\mathcal{T}$-congruous with S, A is also $\mathcal{T}$-congruous with S'.

Since $A \models S \cup \{C\}$ implies $A \models S$, $\mathcal{T}$-contraction-safety is trivial relative to contraction rules that delete clauses, such as subsumption and tautology deletion. In the presence of contraction rules that replace clauses, on the other hand, it is not trivial:

Example 15. Let $\mathcal{T} = \{f(a) \simeq b\}$, $S = \{f(c) \simeq d,\ c \simeq a\}$, and assume that S is $\mathcal{T}$-disconnected (e.g., assume that $c \succ a$, so that paramodulation of $c \simeq a$ into $f(a) \simeq b$ is prevented by the ordering), hence $\mathcal{T}$-congruous with itself by Proposition 13. Suppose that $(\mathcal{T}; S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}; S')$, where S' is derived from S by simplifying $f(c) \simeq d$ by $c \simeq a$, that is, $S' = \{f(a) \simeq d,\ c \simeq a\}$. If $d \simeq b$ is generated from $f(a) \simeq d$ and $f(a) \simeq b$, then $d \simeq b$ is in $\mathcal{D}(S', \mathcal{T})$, and since $S \not\models \{d \simeq b\}$, S is not $\mathcal{T}$-congruous with S'.

Actually, this example is somewhat extreme: a typical presentation will contain some non-ground axioms, so that S is not $\mathcal{T}$-disconnected and such a situation does not occur. Since $\mathcal{T}$-contraction-safety pertains to sets of clauses, it might restrict the class of admissible sets. However, we shall see (cf. Lemma 41 in Section 4.2) that when $\mathrm{Inf}_\succ$ is replaced by $\mathcal{SP}$, $\mathcal{T}$-contraction-safety will turn out to be harmless, and there will be no restriction on simplification in practice.

3.2.2. Unary inferences

If clauses generated by unary inferences do not interact with $\mathcal{T}$, then $\mathcal{T}$-congruity is preserved:

Definition 16. 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 17. *If A is $\mathcal{T}$-congruous with S, S is $\mathcal{T}$-neutral, and $(\mathcal{T}; S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}; S \cup \{D\})$, where D is generated from a clause in S by a unary inference, then A is $\mathcal{T}$-congruous with $S \cup \{D\}$.*

Proof. By soundness of $\mathrm{Inf}_\succ$, $S \models D$. Since $S \subseteq \mathcal{D}(S, \mathcal{T})$, it follows that $\mathcal{D}(S, \mathcal{T}) \models D$. By hypothesis, $A \models \mathcal{D}(S, \mathcal{T})$ and therefore $A \models \mathcal{D}(S, \mathcal{T}) \cup \{D\}$. Since S is $\mathcal{T}$-neutral, D is $\mathcal{T}$-disconnected, $\mathcal{D}(D, \mathcal{T}) = \{D\}$, and $\mathcal{D}(S \cup \{D\}, \mathcal{T}) = \mathcal{D}(S, \mathcal{T}) \cup \{D\}$. Thus, $A \models \mathcal{D}(S \cup \{D\}, \mathcal{T})$. □

3.2.3. Binary inferences between a clause in $\mathcal{T}$ and a clause in S

With Definitions 5, 7 and 9, we defined $\mathcal{T}$-congruity for binary expansion inferences with first parent in S and second parent in $\mathcal{T}$. We require these to be the only binary expansion inferences between S and $\mathcal{T}$. This leads to the notion of *$\mathcal{T}$-orientation*:

Definition 18. A clause C is *$\mathcal{T}$-oriented* if no binary expansion inference applies to a clause in $\mathcal{T}$ as first parent and C as second parent. A set of clauses is *$\mathcal{T}$-oriented* if all its clauses are.

Proposition 19. *If A is $\mathcal{T}$-congruous with S, S is $\mathcal{T}$-oriented, and $(\mathcal{T}; S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}; S \cup \{D\})$, where D is generated by a binary inference applied to a clause in S and one in $\mathcal{T}$, then A is $\mathcal{T}$-congruous with $S \cup \{D\}$.*

Proof. Immediate consequence of Definition 18 and Proposition 11(2). □

The following example shows that $\mathcal{T}$-congruity may not be preserved without $\mathcal{T}$-orientation:

Example 20. Let $\mathcal{T} = \{a \simeq b\}$, $S = \{f(b) \simeq c\}$, and say $\mathrm{Inf}_\succ$ generates $f(a) \simeq c$ from first parent $a \simeq b$ and second parent $f(b) \simeq c$, so that $(\mathcal{T}; S) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}; S')$, with $S' = \{f(b) \simeq c; f(a) \simeq c\}$. Then S is not $\mathcal{T}$-oriented. Suppose that S is $\mathcal{T}$-disconnected and therefore $\mathcal{T}$-congruous with itself by Proposition 13. Since $S \not\models S'$, S is not $\mathcal{T}$-congruous with S'.

Although $\mathcal{T}$-orientation may sound restrictive, we shall see that it is a by-product of flattening for all presentations in Armando et al. (2003), Armando et al. (2009) and Bonacina and Echenim (2007b):

Example 21. Let $\mathcal{T} = \{\mathsf{s}(\mathsf{p}(x)) \simeq x\}$ and $S = \{\mathsf{s}(\mathsf{p}(a)) \simeq b\}$. If $\mathrm{Inf}_\succ$ features paramodulation, S is not $\mathcal{T}$-oriented, because $\mathsf{s}(\mathsf{p}(x)) \simeq x$ paramodulates into $\mathsf{s}(\mathsf{p}(a)) \simeq b$ to generate $a \simeq b$. However, the set $S' = \{\mathsf{p}(a) \simeq a'; \mathsf{s}(a') \simeq b\}$, obtained by flattening S, is $\mathcal{T}$-oriented, because $\mathsf{s}(\mathsf{p}(x)) \simeq x$ does not paramodulate into any of its clauses.

3.2.4. Binary inferences within S

Binary inferences between clauses in the set-of-support require the most control. For this purpose, we define a notion of *associativity* of clauses. The name is suggested by the following notation: let $C \to C'$ represent a clause generated from parents C and C'; then C is $[C', D']$-*associative* if either $(C \to C') \to D' = C \to (C' \to D')$, or $(C \to C') \to D'$ is subsumed by $C' \to D'$. This property is relevant when D' is an axiom in $\mathcal{T}$. Intuitively, it means that if an inference between two clauses leads to another inference with an axiom as second premise, then an inference with an axiom as second premise could have been done beforehand. This disentangles the inferences into axioms from the others, allowing us to do them first. In the next definition, D is $(C \to C')$, E is $(C \to C') \to D'$ and E' is $(C' \to D')$:

Definition 22. 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

(i) either E can be generated from parents C and E',
(ii) or $E' \trianglelefteq E$.

C is *weakly $\mathcal{T}$-associative for C'* if for all $Q \in \mathcal{T}$, C is $[C', Q]$-associative; C is *$\mathcal{T}$-associative for C'* if for all $E' \in \mathcal{D}(C', \mathcal{T})$ and $Q \in \mathcal{T}$, C is $[E', Q]$-associative. A set of clauses S is *weakly $\mathcal{T}$-associative* if for all $C, C' \in S$, C is weakly $\mathcal{T}$-associative for C'; S is *$\mathcal{T}$-associative* if for all $C, C' \in S$, C is $\mathcal{T}$-associative for C'.

Thus, C is $\mathcal{T}$-associative for C', if it is weakly $\mathcal{T}$-associative for all $E \in \mathcal{D}(C', \mathcal{T})$. If C is $\mathcal{T}$-associative for C' and D is generated from parents C and C', all its $\mathcal{T}$-children inherit the conditions of Definition 22. The following lemma proves that this is the case for *all $\mathcal{T}$-descendants* of D:

Lemma 23. *If C is $\mathcal{T}$-associative for C', and D is generated from parents C and C', then for all $n \geq 0$ and $E_n \in \mathcal{D}_n(D, \mathcal{T})$:*

(1) *Either there exists a clause $E'_n \in \mathcal{D}_n(C', \mathcal{T})$ such that E_n is generated from parents C and E'_n;*
(2) *Or for some $k \in \{0, \ldots, n\}$, there exist clauses $E_k \in \mathcal{D}_k(D, \mathcal{T})$ and $E'_k \in \mathcal{D}_k(C', \mathcal{T})$, such that $E_n \in \mathcal{D}(E_k, \mathcal{T})$ and $E'_k \trianglelefteq E_k$.*

Proof. The proof is by induction on n.
Base case: if $n = 0$ then $\mathcal{D}_0(D, \mathcal{T}) = \{D\}$ and $\mathcal{D}_0(C', \mathcal{T}) = \{C'\}$, so that $E_n = D$, $E'_n = C'$ and Claim (1) holds by hypothesis.
Induction hypothesis: assume that either (1) or (2) holds for $n \geq 0$.
Induction step: let F be a clause in $\mathcal{D}_{n+1}(D, \mathcal{T})$. Then, there exist an $E_n \in \mathcal{D}_n(D, \mathcal{T})$ and a $Q \in \mathcal{T}$, such that F is generated from parents E_n and Q. By induction hypothesis, either (1) or (2) is true for E_n. If Claim (2) holds for E_n, then it also holds for F for the same E_k and E'_k.
Assume that Claim (1) holds for E_n: there exists an $E'_n \in \mathcal{D}_n(C', \mathcal{T})$ such that E_n is generated from parents C and E'_n. Since by hypothesis C is $\mathcal{T}$-associative for C', C is $[E'_n, Q]$-associative. We can therefore apply Definition 22: since E_n is generated from parents C and E'_n, and F is generated from parents E_n and Q, there exists a clause F' generated from parents E'_n and Q, such that either Condition (i) or (ii) of Definition 22 applies to F'. Furthermore, by definition, $F' \in \mathcal{D}_{n+1}(C', \mathcal{T})$. If Condition (i) applies, that is, F can be generated from parents C and F', then Claim (1) holds for F. If (ii) applies, that is, $F' \trianglelefteq F$, then Claim (2) holds for F with $k = n + 1$. □

Since $\mathcal{T}$-associativity involves subsumption, in order to ensure that $\mathcal{T}$-congruity is preserved by binary inferences between clauses in S under the hypothesis of $\mathcal{T}$-associativity, we also need that subsumption preserves $\mathcal{T}$-congruity:

Definition 24. A clause C is *$\mathcal{T}$-subsumption-preserving* if for all sets of clauses A $\mathcal{T}$-congruous with C and for all C' such that $C \trianglelefteq C'$, A is $\mathcal{T}$-congruous with C'. A set of clauses S is *$\mathcal{T}$-subsumption-preserving* if all its clauses are.

$\mathcal{T}$-subsumption-preservation, together with $\mathcal{T}$-associativity, guarantees that if A is $\mathcal{T}$-congruous with S, and D is generated by expansion from two clauses in S, then A is also $\mathcal{T}$-congruous with $S \cup \{D\}$. The following lemma establishes a stronger statement, which also allows one to combine $\mathcal{T}$-congruous sets:

Lemma 25. *Assume that D is generated from parents C and C', A is $\mathcal{T}$-congruous with C and A' is $\mathcal{T}$-congruous with C'. If $\mathcal{D}(C', \mathcal{T})$ is $\mathcal{T}$-subsumption-preserving and C is $\mathcal{T}$-associative for C', then $A \cup A'$ is $\mathcal{T}$-congruous with D.*

Proof. We need to show that $A \cup A' \models \mathcal{D}(D, \mathcal{T})$. Assume that $E_n \in \mathcal{D}_n(D, \mathcal{T})$ for some $n \geq 0$. Since C is $\mathcal{T}$-associative for C', either Statement (1) or (2) of Lemma 23 holds for E_n. If Statement (1) holds, $\{C, E'_n\} \models E_n$, and since $E'_n \in \mathcal{D}_n(C', \mathcal{T})$, we have $A' \models E'_n$. Since $A \models C$ holds by hypothesis, $A \cup A' \models E_n$. If Statement (2) holds, by Definition 7, $\mathcal{D}(E'_k, \mathcal{T}) \subseteq \mathcal{D}(C', \mathcal{T})$. Since A' is $\mathcal{T}$-congruous with C', it follows that A' is $\mathcal{T}$-congruous with E'_k as well. By subsumption-preservation, A' is also $\mathcal{T}$-congruous with E_k, hence $A' \models E_n$ and therefore $A \cup A' \models E_n$. □

3.3. $\mathcal{T}$-stability

The collection of conditions built hitherto ensures that each inference preserves $\mathcal{T}$-congruity. In order to generalize preservation from single inferences to derivations, we introduce a notion of $\mathcal{T}$-*closure*:

Definition 26. S is $\mathcal{T}$-*closed* if all clauses inferred from $(\mathcal{T}; S)$ are in S.

Under $\mathcal{T}$-closure, weak $\mathcal{T}$-associativity and $\mathcal{T}$-associativity are equivalent:

Proposition 27. *If a set of clauses S is $\mathcal{T}$-closed, then S is weakly $\mathcal{T}$-associative if and only if it is $\mathcal{T}$-associative.*

Proof. If S is $\mathcal{T}$-associative then it is trivially weakly $\mathcal{T}$-associative. If S is weakly $\mathcal{T}$-associative, for all $C, C' \in S$ and $E \in \mathcal{D}(C', \mathcal{T})$, we have $E \in S$, because S is $\mathcal{T}$-closed. Since S is weakly $\mathcal{T}$-associative, C is weakly $\mathcal{T}$-associative for E. Therefore, C is $\mathcal{T}$-associative for C'. □

The definition of $\mathcal{T}$-*stability* recapitulates all needed conditions for the main theorem:

Definition 28. A set of clauses $\mathcal{B}$ is $\mathcal{T}$-*stable* if it is

- $\mathcal{T}$-neutral,
- $\mathcal{T}$-oriented,
- $\mathcal{T}$-associative,
- $\mathcal{T}$-subsumption-preserving,
- $\mathcal{T}$-closed and such that
- All its subsets are $\mathcal{T}$-contraction-safe.

Theorem 29. *Given a $\mathcal{T}$-stable set $\mathcal{B}$ and two subsets $S, S' \subseteq \mathcal{B}$, if A and A' are sets of clauses such that $\mathcal{T} \cup S \models A$, $\mathcal{T} \cup S' \models A'$, A is $\mathcal{T}$-congruous with S and A' is $\mathcal{T}$-congruous with S', then*

$$\mathcal{T} \cup S \cup S' \equiv_s A \cup A'.$$

Proof. Since $\mathcal{T} \cup S \cup S' \models A \cup A'$, if $A \cup A'$ is unsatisfiable, so is $\mathcal{T} \cup S \cup S'$.
For the converse, let $S_0 = S \cup S'$. We prove by induction on the length of the derivation that if $(\mathcal{T}; S_0) \vdash^i_{\mathrm{Inf}_\succ} (\mathcal{T}; S_i)$ then $A \cup A'$ is $\mathcal{T}$-congruous with S_i.
Base case: for $i = 0$ the claim follows from Proposition 11(1).
Induction hypothesis: assume that the claim holds for $i - 1$, where $i \geq 1$.
Induction step: consider an inference $(\mathcal{T}; S_{i-1}) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}; S_i)$. If it is a contraction, we observe that $S_{i-1} \subseteq \mathcal{B}$, because $\mathcal{B}$ is $\mathcal{T}$-closed by the $\mathcal{T}$-stability hypothesis. Then, again by $\mathcal{T}$-stability, S_{i-1} is $\mathcal{T}$-contraction-safe, hence S_i is $\mathcal{T}$-congruous with $A \cup A'$. If the inference is an expansion, then $S_i = S_{i-1} \cup \{D\}$. Again, because $\mathcal{B}$ is $\mathcal{T}$-closed, $S_i \subseteq \mathcal{B}$. We distinguish three cases:

- If D is generated from $C \in S_{i-1}$ by a unary inference, then, by $\mathcal{T}$-stability, C is $\mathcal{T}$-neutral, and by Proposition 17, $A \cup A'$ is $\mathcal{T}$-congruous with S_i.
- If D is generated by a binary expansion inference from $C \in S_{i-1}$ and $Q \in \mathcal{T}$, then, since S_{i-1} is $\mathcal{T}$-oriented by $\mathcal{T}$-stability, C must be the first parent and Q the second one. By Proposition 11(2), $A \cup A'$ is $\mathcal{T}$-congruous with S_i.
- If D is generated from $C \in S_{i-1}$ and $C' \in S_{i-1}$, then, by hypothesis, C is $\mathcal{T}$-associative for C', hence $A \cup A'$ is $\mathcal{T}$-congruous with D by Lemma 25. Since $A \cup A'$ is $\mathcal{T}$-congruous with both S_{i-1} and D, it follows from Proposition 11(1) that $A \cup A'$ is $\mathcal{T}$-congruous with S_i as well.

If $\mathcal{T} \cup S \cup S'$ is unsatisfiable, by the refutational completeness of $\mathrm{Inf}_\succ$, there exists an n such that $\Box \in S_n$. Since $A \cup A'$ is $\mathcal{T}$-congruous with S_n, it follows that $A \cup A'$ is also unsatisfiable. □

Theorem 29 answers the question posed at the beginning of this section: if S and S' are subsets of a $\mathcal{T}$-stable set, A is $\mathcal{T}$-congruous with S and A' is $\mathcal{T}$-congruous with S', then deciding the satisfiability of $A \cup A'$ is equivalent to deciding that of $\mathcal{T} \cup S \cup S'$.

We conclude with a consequence of Theorem 29 that will be applied in Section 7. Since a set of clauses G can be read as their conjunction, let $\neg G$ denote a set of clauses that is logically equivalent to the negation of G:

Corollary 30. *Given a $\mathcal{T}$-stable set $\mathcal{B}$ and two subsets $S, G \subseteq \mathcal{B}$, if $\mathcal{T} \cup S \models A$, A is $\mathcal{T}$-congruous with S and G is $\mathcal{T}$-congruous with itself, then $\mathcal{T} \cup S \models \neg G$ if and only if $A \models \neg G$.*

Proof. Since $\mathcal{T} \cup G \models G$ trivially holds, all hypotheses of Theorem 29 are satisfied. By Theorem 29, $\mathcal{T} \cup S \cup G \equiv_s A \cup G$, whence the result follows. □

3.4. Combination of theories

We complete the framework by considering the case where $\mathcal{T}$ is a combination of theories. Throughout this section, $\mathcal{T}_1$ and $\mathcal{T}_2$ are the presentations of two theories, and $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$. The central issue is whether it is sufficient to require $\mathcal{T}_1$-congruity and $\mathcal{T}_2$-congruity to guarantee equisatisfiability with respect to $\mathcal{T}$. The next lemma states sufficient conditions for $\mathcal{T}$-descendants to be $\mathcal{T}_1$-descendants (or, symmetrically, $\mathcal{T}_2$-descendants), so that $\mathcal{T}$-congruity reduces to $\mathcal{T}_1$-congruity (or, symmetrically, $\mathcal{T}_2$-congruity):

Lemma 31. *If $\mathcal{B}$ is $\mathcal{T}_1$-stable and $\mathcal{T}_2$-disconnected, then for all $S \subseteq \mathcal{B}$, $\mathcal{D}(S, \mathcal{T}) = \mathcal{D}(S, \mathcal{T}_1)$.*

Proof. Let $S \subseteq \mathcal{B}$. Clearly, $\mathcal{D}(S, \mathcal{T}_1) \subseteq \mathcal{D}(S, \mathcal{T})$. For the other direction, we prove by induction on n that $\mathcal{D}_n(S, \mathcal{T}) \subseteq \mathcal{D}_n(S, \mathcal{T}_1)$. For $n = 0$, the result is obvious, since $\mathcal{D}_0(S, \mathcal{T}) = \mathcal{D}_0(S, \mathcal{T}_1) = S$. Suppose the result is true for $n - 1$, and let $E \in \mathcal{D}_n(S, \mathcal{T})$. By definition, there exists a $D \in \mathcal{D}_{n-1}(S, \mathcal{T})$ and a $Q \in \mathcal{T}$ such that E is generated from parents D and Q. By the induction hypothesis, $D \in \mathcal{D}_{n-1}(S, \mathcal{T}_1)$, and since $\mathcal{B}$ is $\mathcal{T}_1$-closed, $D \in \mathcal{B}$. Since by hypothesis, $\mathcal{B}$ is $\mathcal{T}_2$-disconnected, Q must be in $\mathcal{T}_1$ for E to be generated from parents D and Q. Thus, $E \in \mathcal{D}_n(S, \mathcal{T}_1)$, which proves the result. □

Under the hypotheses of Lemma 31, if A is $\mathcal{T}_1$-congruous with S, then it is also $\mathcal{T}$-congruous with S. This is the main property we use in the proof of the following:

Theorem 32. *Given two sets of clauses $\mathcal{B}_1$ and $\mathcal{B}_2$ such that, for $\{i, j\} = \{1, 2\}$, $i \neq j$, $\mathcal{B}_i$ is $\mathcal{T}_i$-stable and $\mathcal{T}_j$-disconnected, assume there exists a $\mathcal{T}$-stable set $\mathcal{B}$, such that $\mathcal{B}_1 \cup \mathcal{B}_2 \subseteq \mathcal{B}$. Then for all sets S_1, S_2, A_1 and A_2 such that for $i \in \{1, 2\}$, $S_i \subseteq \mathcal{B}_i$, $\mathcal{T}_i \cup S_i \models A_i$ and A_i is $\mathcal{T}_i$-congruous with S_i,*

$$\mathcal{T}_1 \cup \mathcal{T}_2 \cup S_1 \cup S_2 \equiv_s A_1 \cup A_2.$$

Proof. By the hypotheses, $S_1, S_2 \subseteq \mathcal{B}$. For $i \in \{1, 2\}$, since $\mathcal{T}_i \cup S_i \models A_i$, we also have $\mathcal{T} \cup S_i \models A_i$. By Lemma 31, $\mathcal{D}(S_i, \mathcal{T}) = \mathcal{D}(S_i, \mathcal{T}_i)$, so that A_i is $\mathcal{T}$-congruous with S_i. We can therefore apply Theorem 29 to conclude that $\mathcal{T} \cup S_1 \cup S_2 \equiv_s A_1 \cup A_2$. □

Theorem 32 relies on the existence of a $\mathcal{T}$-stable set $\mathcal{B}$ encompassing both the $\mathcal{T}_1$-stable set $\mathcal{B}_1$ and the $\mathcal{T}_2$-stable set $\mathcal{B}_2$. We develop sufficient conditions to guarantee that it is sufficient to take the union $\mathcal{B}_1 \cup \mathcal{B}_2$ to have such a $\mathcal{T}$-stable set.

Lemma 33. *If $\mathcal{B}$ is $\mathcal{T}_1$-stable and $\mathcal{T}_2$-disconnected, then $\mathcal{B}$ is $\mathcal{T}$-associative.*

Proof. For $C, C' \in \mathcal{B}$ and $E' \in \mathcal{D}(C', \mathcal{T})$, we need to show that for all $Q \in \mathcal{T}$, C is $[E', Q]$-associative. By Lemma 31, $E' \in \mathcal{D}(C', \mathcal{T}_1)$, and therefore, $E' \in \mathcal{B}$, since $\mathcal{B}$ is $\mathcal{T}_1$-closed. If $Q \in \mathcal{T}_1$, then C is $[E', Q]$-associative, since $\mathcal{B}$ is $\mathcal{T}_1$-associative. Now suppose that $Q \in \mathcal{T}_2$, and let D be a clause generated from parents C and E'. Since $\mathcal{B}$ is $\mathcal{T}_1$-closed, $D \in \mathcal{B}$, and since $\mathcal{B}$ is $\mathcal{T}_2$-disconnected, there is no binary inference with D and Q as premises. Therefore, C is trivially $[E', Q]$-associative. □

Lemma 34. *Assume that $\mathcal{B}_1$ and $\mathcal{B}_2$ are sets of clauses with the following properties: for $\{i, j\} = \{1, 2\}$, $i \neq j$, $\mathcal{B}_i$ is $\mathcal{T}_i$-stable and $\mathcal{T}_j$-disconnected, and whenever a binary expansion inference applies to $C \in \mathcal{B}_i$ as first parent and $C' \in \mathcal{B}_j$ as second parent, $C \in \mathcal{B}_j$. Then $\mathcal{B}_1 \cup \mathcal{B}_2$ is $\mathcal{T}$-closed.*

Proof. Let $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$ and consider an inference $(\mathcal{T}; \mathcal{B}) \vdash_{\mathrm{Inf}_\succ} (\mathcal{T}; \mathcal{B}')$, where $\mathcal{B}' = \mathcal{B} \cup \{D\}$. If D is generated by a unary inference applied to a clause C, we assume without loss of generality that $C \in \mathcal{B}_1$. Since $\mathcal{B}_1$ is $\mathcal{T}_1$-stable, it is $\mathcal{T}_1$-closed, and $D \in \mathcal{B}_1 \subseteq \mathcal{B}$. If D is generated by a binary inference from parents $C \in \mathcal{B}$ and $Q \in \mathcal{T}$, we may assume that $C \in \mathcal{B}_1$. Since $\mathcal{B}_1$ is $\mathcal{T}_2$-disconnected, Q must be in $\mathcal{T}_1$, so that, by $\mathcal{T}_1$-stability of $\mathcal{B}_1$, it must be the case that $D \in \mathcal{B}_1 \subseteq \mathcal{B}$. Suppose D is generated by a binary inference from parents $C, C' \in \mathcal{B}$. If both C and C' are in $\mathcal{B}_1$, D also must be in $\mathcal{B}_1$, because $\mathcal{B}_1$ is $\mathcal{T}_1$-closed. If $C \in \mathcal{B}_1$ and $C' \in \mathcal{B}_2$, then by hypothesis $C \in \mathcal{B}_2$ and we have the result. The cases where both C and C' are in $\mathcal{B}_2$ or $C \in \mathcal{B}_2$ and $C' \in \mathcal{B}_1$ are symmetric. □

The following theorem summarizes the conditions for $\mathcal{T}$-stability of $\mathcal{B}_1 \cup \mathcal{B}_2$:

Theorem 35. *Given sets of clauses $\mathcal{B}_1$ and $\mathcal{B}_2$ such that, for $\{i, j\} = \{1, 2\}$, $i \neq j$:*

- *$\mathcal{B}_i$ is $\mathcal{T}_i$-stable and $\mathcal{T}_j$-disconnected;*
- *If there is a binary inference with $C \in \mathcal{B}_i$ as first parent and $C' \in \mathcal{B}_j$ as second parent, then $C \in \mathcal{B}_j$;*
- *Every subset of $\mathcal{B}_1 \cup \mathcal{B}_2$ is $\mathcal{T}$-contraction-safe;*
- *$\mathcal{B}_1 \cup \mathcal{B}_2$ is $\mathcal{T}$-subsumption-preserving;*

then $\mathcal{B}_1 \cup \mathcal{B}_2$ is $\mathcal{T}$-stable.

Proof. Let $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$. By hypothesis, every subset of $\mathcal{B}_1 \cup \mathcal{B}_2$ is $\mathcal{T}$-contraction-safe and $\mathcal{B}$ is $\mathcal{T}$-subsumption-preserving, and by Lemma 34 $\mathcal{B}$ is $\mathcal{T}$-closed. We are left to prove that $\mathcal{B}$ is $\mathcal{T}$-neutral, $\mathcal{T}$-oriented and $\mathcal{T}$-associative:

$\mathcal{T}$-neutrality: let D be a clause generated by a unary inference applied to a clause C; we may assume without loss of generality that $C \in \mathcal{B}_1$. As in the proof of Lemma 34, $D \in \mathcal{B}_1$. Since $\mathcal{B}_1$ is $\mathcal{T}_1$-neutral, D is $\mathcal{T}_1$-disconnected, and since $\mathcal{B}_1$ is $\mathcal{T}_2$-disconnected, D is $\mathcal{T}$-disconnected.

$\mathcal{T}$-orientation: let D be a clause generated by a binary inference applied to clauses $C \in \mathcal{B}$ and $Q \in \mathcal{T}$; we may assume that $C \in \mathcal{B}_1$. Since $\mathcal{B}_1$ is $\mathcal{T}_2$-disconnected, Q must be in $\mathcal{T}_1$, and since $\mathcal{B}_1$ is $\mathcal{T}_1$-oriented, D is generated from parents C and Q, thus, $\mathcal{B}$ is also $\mathcal{T}$-oriented.

$\mathcal{T}$-associativity: by Lemma 33, both $\mathcal{B}_1$ and $\mathcal{B}_2$ are $\mathcal{T}$-associative. Let $C, C' \in \mathcal{B}$, and suppose that $C \in \mathcal{B}_2$ and $C' \in \mathcal{B}_1$. Take $E \in \mathcal{D}(C', \mathcal{T})$. Since $\mathcal{B}_1$ is $\mathcal{T}_1$-stable and $\mathcal{T}_2$-disconnected, by Lemma 31, $\mathcal{D}(C', \mathcal{T}) = \mathcal{D}(C', \mathcal{T}_1)$, hence $E \in \mathcal{B}_1$, since $\mathcal{B}_1$ is $\mathcal{T}_1$-closed. If no clause can be generated from C and E, there is nothing to prove. If D is generated from $C \in \mathcal{B}_2$ and $E \in \mathcal{B}_1$, then, by hypothesis, C must be in $\mathcal{B}_1$. Thus, $C, C' \in \mathcal{B}_1$, and since $\mathcal{B}_1$ is $\mathcal{T}$-associative, C is $\mathcal{T}$-associative for C'. □

Theorems 32 and 35 give a set of conditions ensuring that $\mathcal{T}_1$-congruity and $\mathcal{T}_2$-congruity can be combined. These conditions are fully abstract, and do not rest on assumptions such as the two theories having disjoint signatures. As we shall see in Sections 4.3 and 7, disjointness of signatures will be one of the ingredients to satisfy the abstract conditions for specific inference systems and theories.

4. Decision procedures by stages

The framework developed in the previous section, and chiefly the notions of $\mathcal{T}$-congruity and $\mathcal{T}$-stability, allow us to refine the *decomposition principle*, stated at the beginning of Section 3, as follows:

(1) Given SMT problem $\mathcal{T} \cup P$, decompose P into a definitional part S and an operational part S', in such a way that $\mathcal{T} \cup P \equiv_s \mathcal{T} \cup S \cup S'$;
(2) Generate sets A and A' that are $\mathcal{T}$-congruous with S and S', respectively;
(3) Test the satisfiability of $A \cup A'$ to decide that of $\mathcal{T} \cup S \cup S'$.

A few remarks are in order:

- The distinction between definitional and operational part depends on the theory: we shall see specific instances, that involve flattening, in Section 5.
- This method is correct if there exists a $\mathcal{T}$-stable set containing S and S'. This $\mathcal{T}$-stable set should be *as large as possible*, to include, ideally, S and S' for all input problems. In Section 5, we will construct $\mathcal{T}$-stable sets that contain *all sets of flat clauses*, so that this method is correct for all *ground formulæ* in the theories at hand.
- The operational part S' is not meant to interact with $\mathcal{T}$, and this will be achieved by showing that it is $\mathcal{T}$-disconnected. Since a $\mathcal{T}$-disconnected set is $\mathcal{T}$-congruous with itself (cf. Proposition 13), the rôle of A' will be played by S' itself.
- Since A' is S' itself, the crucial issue is to generate A. To this end, we replace the generic $\mathrm{Inf}_\succ$ with a variant of $\mathcal{SP}_\succ$, named $\mathcal{U}_\succ$, and show that if S_∞ is the limit of a fair $\mathcal{SP}_\succ$-derivation from $\mathcal{T} \cup S$, it is possible to extract from S_∞ a subset $\bar{S}$, that is $\mathcal{T}$-congruous with S with respect to $\mathcal{U}_\succ$. If S_∞ is finite, so is $\bar{S}$. Thus, A will be $\bar{S}$, generated by $\mathcal{SP}_\succ$ by compiling the theory.

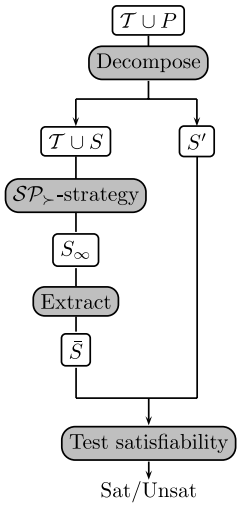

Fig. 3. $\mathcal{T}$-decision procedures by stages based on $\mathcal{SP}$.

Fig. 3 summarizes the resulting implementation of the decomposition principle, that we call *decision by stages*: after P is decomposed into S and S', a fair $\mathcal{SP}_\succ$-strategy applied to $\mathcal{T} \cup S$ generates S_∞; next, $\bar{S}$ is extracted from S_∞ and $\bar{S} \cup S'$ is tested for satisfiability.

4.1. The inference system $\mathcal{U}_\succ$

We define $\mathcal{U}_\succ$ in such a way that it can perform all $\mathcal{SP}_\succ$-inferences:

Definition 36. Let $\mathcal{U}_\succ$ denote the inference system that features all contraction rules of $\mathcal{SP}_\succ$, and such that for all expansion inferences $(\mathcal{T}; S) \vdash_{\mathcal{U}_\succ} (\mathcal{T}; S \cup \{D\})$:

- Either D is generated by a unary $\mathcal{SP}_\succ$-inference with premise in S,
- Or D is generated by a binary $\mathcal{SP}_\succ$-inference with a premise in $\mathcal{T}$ and one in S,
- Or D is generated by a binary $\mathcal{SP}_\succ$-inference with both premises in S, where Conditions (iii) and (iv) of Fig. 1 are not required to hold.[12]

$\mathcal{U}_\succ$ is refutationally complete for supported strategies, provided $\mathcal{T}$ is saturated, because $\mathcal{SP}_\succ$ is. For readability, we write *superposition/paramodulation*, when the inference satisfies all ordering constraints of Fig. 1, and *unordered superposition/paramodulation*, when the inference, applied to two clauses in S, satisfies only the weaker ordering constraints. The reason for relaxing Conditions (iii) and (iv) of Fig. 1 is to guarantee that a larger class of sets be $\mathcal{T}$-associative with respect to $\mathcal{U}_\succ$:

Example 37. Let $C = a \simeq b \vee a \simeq c$, $C' = \mathsf{p}(a) \simeq c$ and $Q = \mathsf{s}(\mathsf{p}(x)) \simeq x$. Assuming a precedence where $\mathsf{s} \succ \mathsf{p} \succ a \succ b \succ c$, we show that C is $[C', Q]$-associative. Clause $D = \mathsf{p}(b) \simeq c \vee a \simeq c$ is generated by superposing C into C', $E = \mathsf{s}(c) \simeq b \vee a \simeq c$ is generated by superposing D into Q, and $E' = \mathsf{s}(c) \simeq a$ is generated by superposing C' into Q. Since a is not maximal in E', E cannot be generated by superposing C into E', but it can be generated by *unordered superposition*. Thus, C is $[C', Q]$-associative with respect to $\mathcal{U}_\succ$.

The next proposition establishes the key relation between $\mathcal{U}_\succ$ and $\mathcal{SP}_\succ$, namely that a fair $\mathcal{SP}_\succ$-derivation yields a set that is $\mathcal{T}$-congruous with S for $\mathcal{U}_\succ$:

Proposition 38. (1) *For all sets of clauses S, if $S_\infty = \mathcal{T} \uplus \bar{S}$ is the limit of a fair $\mathcal{SP}_\succ$-derivation from $\mathcal{T} \cup S$, then $\bar{S}$ is $\mathcal{T}$-congruous with S for $\mathcal{U}_\succ$.*

[12] That is: if $u \simeq v \vee D$ paramodulates into $C = l[u'] \bowtie r \vee C'$, then $l[u'] \bowtie r$ is not necessarily a maximal literal in C, and $l[u']$ is not necessarily maximal in this literal.

(2) *A set of clauses $\mathcal{B}$ is $\mathcal{T}$-disconnected for $\mathcal{U}_\succ$ if and only if no $\mathcal{SP}_\succ$-inference applies to a clause in $\mathcal{B}$ and one in $\mathcal{T}$.*

Proof. By Definition 36, $\mathcal{U}_\succ$-inferences are the same as $\mathcal{SP}_\succ$-inferences, whenever a premise from $\mathcal{T}$ is involved. Statement (2) follows immediately from this observation. For the same reason, the set of $\mathcal{T}$-descendants of S with respect to $\mathcal{U}_\succ$ is equal to the set of $\mathcal{T}$-descendants of S with respect to $\mathcal{SP}_\succ$. Let $\mathcal{D}(S, \mathcal{T})$ be this set of descendants. By soundness of $\mathcal{SP}_\succ$ and fairness of the derivation, $S_\infty \models \mathcal{D}(S, \mathcal{T})$, and $S \models \mathcal{D}(S, \mathcal{T})$, since $\mathcal{D}(S, \mathcal{T})$ does not include $\mathcal{T}$ itself, whence Statement (1) follows. □

Proposition 38 justifies using $\mathcal{U}_\succ$, to establish $\mathcal{T}$-congruity and $\mathcal{T}$-stability, and using $\mathcal{SP}_\succ$ in practice, so that the additional inferences allowed by $\mathcal{U}_\succ$ bear no consequences on performances.

4.2. $\mathcal{T}$-stability with respect to $\mathcal{U}_\succ$

Once the generic $\mathrm{Inf}_\succ$ is instantiated to the specific inference system $\mathcal{U}_\succ$, $\mathcal{T}$-stability is conquered by proving that every clause is $\mathcal{T}$-subsumption-preserving (cf. Definition 24) for $\mathcal{U}_\succ$ and that $\mathcal{T}$-associativity implies $\mathcal{T}$-contraction-safety, essentially because simplification can be simulated by superposition.

Lemma 39. *If $C \trianglelefteq C'$, then for all $D' \in \mathcal{D}(C', \mathcal{T})$, there is a $D \in \mathcal{D}(C, \mathcal{T})$ such that $D \trianglelefteq D'$.*

Proof. We prove the result on the $\mathcal{D}_i(C', \mathcal{T})$'s by induction on i.
Base case: if $i = 0$, the claim is trivially true, since $\mathcal{D}_0(C', \mathcal{T}) = \{C'\}$ and $C \in \mathcal{D}(C, \mathcal{T})$.
Induction hypothesis: assume that the claim holds for all $E' \in \mathcal{D}_{i-1}(C', \mathcal{T})$.
Induction step: let $D' \in \mathcal{D}_i(C', \mathcal{T})$. By Definition 7 instantiated for $\mathcal{U}_\succ$, there exists an $E' \in \mathcal{D}_{i-1}(C', \mathcal{T})$ and a $Q \in \mathcal{T}$ such that $E' = u \simeq v \vee F'$, $Q = l[u'] \bowtie r \vee Q'$ and $D' = (l[v] \bowtie r \vee F' \vee Q')\sigma$, where $u\sigma = u'\sigma$. By induction hypothesis there is an $E \in \mathcal{D}(C, \mathcal{T})$ such that $E \trianglelefteq E'$, and there are two cases:

- If $E\mu \subseteq F'$ (as multisets) for some substitution μ, then $E\mu\sigma \subseteq D'$ and $E \trianglelefteq D'$.
- Otherwise, $E = u'' \simeq v'' \vee F$, $(u'' \simeq v'')\mu = u \simeq v$ and $F\mu \subseteq F'$ (as multisets) for some substitution μ. Literal $u'' \simeq v''$ is maximal in E, because $u \simeq v$ is maximal in E'. Furthermore $u''\mu\sigma = u'\sigma$. Superposition of E into Q generates $D = (l[v''] \bowtie r \vee F \vee Q')\mu\sigma = (l[v] \bowtie r \vee F\mu \vee Q')\sigma$, which is in $\mathcal{D}(C, \mathcal{T})$ and subsumes D'.

Thus, the claim is established. □

Theorem 40. *Every clause is $\mathcal{T}$-subsumption-preserving for $\mathcal{U}_\succ$.*

Proof. Consider clauses C and C' such that $C \trianglelefteq C'$, and a set of clauses A that is $\mathcal{T}$-congruous with C. By Lemma 39 every element of $\mathcal{D}(C', \mathcal{T})$ is subsumed by an element of $\mathcal{D}(C, \mathcal{T})$, whence $\mathcal{D}(C, \mathcal{T}) \models \mathcal{D}(C', \mathcal{T})$. Since $A \models \mathcal{D}(C, \mathcal{T})$, it follows that $A \models \mathcal{D}(C', \mathcal{T})$ and A is also $\mathcal{T}$-congruous with C'. □

Lemma 41. *For inference system $\mathcal{U}_\succ$, if a set of clauses S is $\mathcal{T}$-associative, then it is also $\mathcal{T}$-contraction-safe.*

Proof. Let A be a set of clauses that is $\mathcal{T}$-congruous with S. As already noted, if $(\mathcal{T}; S) \vdash_{\mathcal{U}_\succ} (\mathcal{T}; S')$ by a subsumption or deletion inference, A is also $\mathcal{T}$-congruous with S'. Suppose the inference is a simplification of $C' \in S$ by a unit clause $C \in S$ and let D be the generated clause, so that $S' = (S \setminus \{C'\}) \cup \{D\}$. Clause D can also be generated by superposing C into C'. By Theorem 40, $\mathcal{D}(C', \mathcal{T})$ is $\mathcal{T}$-subsumption-preserving, and, by hypothesis, C is $\mathcal{T}$-associative for C'. We can therefore apply Lemma 25, which proves that A is $\mathcal{T}$-congruous with D, and deduce that A is also $\mathcal{T}$-congruous with S'. □

In summary, we have the following characterization of $\mathcal{T}$-stability for $\mathcal{U}_\succ$:

Theorem 42. *If a set of clauses $\mathcal{B}$ is*

- *$\mathcal{T}$-neutral,*
- *$\mathcal{T}$-oriented,*

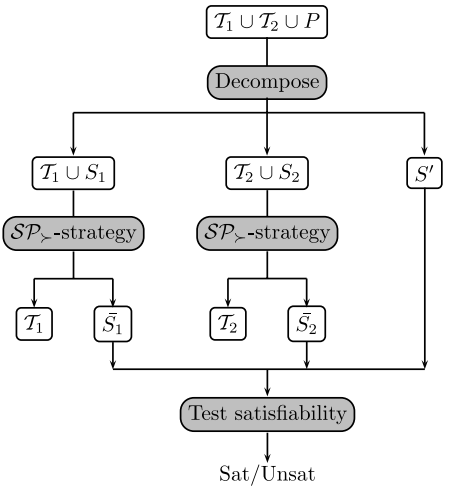

Fig. 4. $\mathcal{T}$-decision procedure by stages for a combination of theories.

- *Weakly $\mathcal{T}$-associative and*
- *$\mathcal{T}$-closed*

with respect to $\mathcal{U}_\succ$, then it is $\mathcal{T}$-stable for $\mathcal{U}_\succ$.

Proof. Following Definition 28, we need to show that $\mathcal{B}$ is $\mathcal{T}$-associative, $\mathcal{T}$-subsumption-preserving and such that all its subsets are $\mathcal{T}$-contraction-safe. Since $\mathcal{B}$ is $\mathcal{T}$-closed and weakly $\mathcal{T}$-associative, it is $\mathcal{T}$-associative by Proposition 27. By Theorem 40, every clause in $\mathcal{B}$ is $\mathcal{T}$-subsumption-preserving, and therefore $\mathcal{B}$ itself is. Because $\mathcal{B}$ is $\mathcal{T}$-associative, its subsets are also, and therefore by Lemma 41, they are $\mathcal{T}$-contraction-safe. □

4.3. *Combination of theories: The* two-theories scheme

If $\mathcal{T}_1$ and $\mathcal{T}_2$ are the presentations of two theories and $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$, the decomposition principle yields the *two-theories scheme* of Fig. 4. Given an SMT problem $\mathcal{T} \cup P$, P is decomposed into its definitional part S and its operational part S' as before. Furthermore, S is decomposed into a set S_1, that contains the clauses that interact with $\mathcal{T}_1$, and a set S_2, that contains the clauses that interact with $\mathcal{T}_2$. A fair $\mathcal{SP}_\succ$-strategy applied to $\mathcal{T}_1 \cup S_1$ and $\mathcal{T}_2 \cup S_2$ generates limits $\mathcal{T}_1 \cup \bar{S}_1$ and $\mathcal{T}_2 \cup \bar{S}_2$, respectively. Then, $\bar{S}_1 \cup \bar{S}_2 \cup S'$ is tested for satisfiability. Theorem 32 guarantees that this scheme is correct, provided we determine a $\mathcal{T}$-stable set large enough to contain S_1, S_2 and S'. While Theorem 35 provides a general way to determine such a set, the next theorem shows that fewer hypotheses are sufficient when the inference system is $\mathcal{U}_\succ$:

Theorem 43. *Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be sets of clauses such that, for $\{i, j\} = \{1, 2\}$, $i \neq j$,*

(1) *$\mathcal{B}_i$ is $\mathcal{T}_i$-stable and $\mathcal{T}_j$-disconnected;*
(2) *Whenever there is a binary $\mathcal{U}_\succ$-inference with $C \in \mathcal{B}_i$ as first parent and $C' \in \mathcal{B}_j$ as second parent, then $C \in \mathcal{B}_j$.*

Then $\mathcal{B}_i \cup \mathcal{B}_j$ is $(\mathcal{T}_1 \cup \mathcal{T}_2)$-stable.

Proof. Let $\mathcal{B} = \mathcal{B}_1 \cup \mathcal{B}_2$: we show that $\mathcal{B}$ is $\mathcal{T}$-subsumption-preserving and all its subsets are $\mathcal{T}$-contraction-safe; then Theorem 35 will prove that $\mathcal{B}$ is $\mathcal{T}$-stable. $\mathcal{T}$-subsumption-preservation follows from Theorem 40. For $\mathcal{T}$-contraction-safety, if we can prove that $\mathcal{B}$ is $\mathcal{T}$-associative, then every subset of $\mathcal{B}$ will also be $\mathcal{T}$-associative, and by Lemma 41, $\mathcal{T}$-contraction-safe. By Lemma 34, $\mathcal{B}$ is $\mathcal{T}$-closed: if we prove that $\mathcal{B}$ is weakly $\mathcal{T}$-associative, by Proposition 27, $\mathcal{B}$ will also be $\mathcal{T}$-associative. Let $C, C' \in \mathcal{B}$, and, without loss of generality, suppose that $C \in \mathcal{B}_1$ and $C' \in \mathcal{B}_2$ (the other case is symmetric). If no

clause can be generated from C and C', then there is nothing to prove. Otherwise, by Hypothesis (2), C must be in $\mathcal{B}_2$, and both C and C' are in $\mathcal{B}_2$. Since $\mathcal{B}_2$ is $\mathcal{T}_2$-stable, it is $\mathcal{T}_2$-associative, and by Lemma 33, it is $\mathcal{T}$-associative. Therefore, C is $\mathcal{T}$-associative for C', and we have the result. □

Although requirement (2) may seem strong, we shall see in Section 7 (cf. Lemma 69) that this is not the case, if the theories do not share function symbols and $\succ$ is good.

5. Decision procedures by stages for specific theories

In order to apply the scheme of decision by stages, we begin by replacing the insofar abstract notion of decomposition by a concrete one, inspired by two observations from previous work: for the theories of data structures in Armando et al. (2003), Armando et al. (2009) and Bonacina and Echenim (2007b),

(1) $\mathcal{SP}_\succ$ terminates on $\mathcal{T}$-satisfiability problems $\mathcal{T} \cup S$, where S is a set of ground flat unit clauses;
(2) No $\mathcal{SP}_\succ$-inference applies to a ground strictly flat clause and a non-ground axiom in $\mathcal{T}$.

Then, decomposition is defined as follows: given an SMT problem $\mathcal{T} \cup P$, P is flattened and the resulting set is decomposed into S and S', in such a way that S contains *only flat unit clauses*, and S' contains *all the strictly flat non-unit clauses*. This decomposition matches the abstract notion of decomposition of the previous sections: S will only contain clauses of the form $f(a) \simeq b$, that contribute to *defining* function symbols of the theory (*definitional* part); while S' will include all the strictly flat clauses (*operational* part). The latter contains the boolean structure of the formula, represents the relations between individuals, is $\mathcal{T}$-disconnected with respect to $\mathcal{SP}_\succ$, and, therefore, with respect to $\mathcal{U}_\succ$, by Statement (2) of Proposition 38. Observation (1) says that $\mathcal{SP}$ generates a finite limit S_∞ from $\mathcal{T} \cup S$, while Observation (2) befits the scheme of Fig. 3 where there is no need to apply $\mathcal{SP}$ to S'. For each theory considered in this section, we determine a $\mathcal{T}$-stable set, with respect to $\mathcal{U}_\succ$, that is large enough to contain all flat unit clauses and strictly flat clauses. This guarantees that decision by stages solves *all SMT problems* in these theories.

5.1. The theory of records

The *theory of records* with n fields assumes a signature Σ that features, for all $i \in \{1, \ldots, n\}$, the function symbols rstore_i, that represents storing a value in the ith-field of a record, and rselect_i, that represents extracting a value from the ith-field of a record. It is presented by the following (saturated) presentation, named $\mathcal{R}$:

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

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

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

$$\forall x, y. \quad \left(\bigwedge_{i=1}^{n} \text{rselect}_i(x) \simeq \text{rselect}_i(y) \right) \rightarrow x \simeq y.$$

An $\mathcal{R}^e$-satisfiability problem can be reduced to an $\mathcal{R}$-decision problem (cf. Lemma 1 in Armando et al. (2009)):

Lemma 44 (*Armando et al., 2009*). *Let $S = S_1 \uplus S_2$ be a set of ground flat unit clauses, such that S_2 contains the clauses 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 $\bigvee_{i=1}^{n} \text{rselect}_i(l) \not\simeq \text{rselect}_i(r)$. Then $\mathcal{R}^e \cup S \equiv_s \mathcal{R} \cup S_1 \cup \{C_L \mid L \in S_2\}$.*

Since an $\mathcal{R}^e$-decision problem can be reduced to an $\mathcal{R}^e$-satisfiability problem by reduction to disjunctive normal form, it follows that an $\mathcal{R}^e$-decision problem can be reduced to an $\mathcal{R}$-decision problem. Thus, we assume without loss of generality that we have to decide the satisfiability of a set of ground clauses P modulo $\mathcal{R}$. It is sufficient to put in S the unit clauses that represent a "store" operation on a record:

Decomposition. Set P is decomposed into S and S' as follows: P is flattened; among the resulting clauses, the unit clauses of the form $\text{rstore}_i(a, e) \simeq b$ go in S and all other clauses in S'.

Proposition 45. *S' is $\mathcal{R}$-disconnected.*

Proof. Clauses in S' are strictly flat or have the form $\text{rselect}_i(a) \bowtie e$. By inspection, there are no $\mathcal{SP}_\succ$-inferences between these clauses and those in $\mathcal{R}$. □

By Lemma 2 of Armando et al. (2009), the saturation of $\mathcal{R} \cup S$ by $\mathcal{SP}_\succ$ (first stage in Fig. 3) produces a finite limit $\mathcal{R} \cup \bar{S}$, that is, the axioms go unaltered through the saturation. In order to establish that $\mathcal{R} \cup P$ and $\bar{S} \cup S'$ are equisatisfiable, we need to identify a set $\mathcal{B}_\mathcal{R}$ that contains all flat unit and strictly flat clauses, so that it contains S and S' for all problems P, and is $\mathcal{R}$-stable.

Definition 46. Let $\mathcal{B}_\mathcal{R}$ denote the set of all ground clauses that are

(i) strictly flat, or have one of the following forms:
(ii) $\text{rstore}_i(a, e) \simeq b \vee B$,
(iii) $\text{rselect}_i(a) \simeq e \vee B$ or
(iv) $\text{rselect}_i(a) \simeq \text{rselect}_i(b) \vee B$,

where $1 \leq i \leq n$ and B is a possibly empty, strictly flat clause.

According to Theorem 42, we begin by proving neutrality, orientation and closure:

Lemma 47. *$\mathcal{B}_\mathcal{R}$ is $\mathcal{R}$-neutral, $\mathcal{R}$-oriented and $\mathcal{R}$-closed.*

Proof. Since every clause in $\mathcal{B}_\mathcal{R}$ is ground and contains at most one non-strictly-flat literal, by goodness of the ordering $\succ$, if such a literal exists, then it must be maximal in the clause. Assume that a $\mathcal{U}_\succ$-inference applied to $(\mathcal{R}; \mathcal{B}_\mathcal{R})$ generates a clause D; we distinguish the following cases:

Unary inference: if D is generated by a unary inference applied to a $C \in \mathcal{B}_\mathcal{R}$, it is either a reflection or an equational factoring step. If it is a reflection, C must be of the form $C' \vee u \not\simeq u'$, where $u \not\simeq u'$ is maximal in C. Such an inference is possible only if C is a clause in category (i) of Definition 46. Thus, D is a strictly flat clause, which is in $\mathcal{B}_\mathcal{R}$. If the inference is by equational factoring, C must be of the form $C' \vee u \simeq t \vee u' \simeq t'$. For the rule to apply, u and u' must unify; since they are both ground, they must be equal and can only be constants, because C contains at most one literal that is not strictly flat. By Proposition 4, it follows that C is strictly flat. Thus, D is also strictly flat and is in $\mathcal{B}_\mathcal{R}$. Since there is paramodulation into variables, there is no inference between a strictly flat clause and an axiom in $\mathcal{R}$, which means that D is $\mathcal{R}$-disconnected. Hence, $\mathcal{B}_\mathcal{R}$ is $\mathcal{R}$-neutral.

Binary inference between $\mathcal{B}_\mathcal{R}$ and $\mathcal{R}$: Suppose D is generated by a binary inference applied to $C \in \mathcal{B}_\mathcal{R}$ and $C' \in \mathcal{R}$. It is simple to check that C must have been the first parent and C' the second one, so that $\mathcal{B}_\mathcal{R}$ is $\mathcal{R}$-oriented. A superposition of a clause in category (ii) of Definition 46 into a clause of the form $\text{rselect}_i(\text{rstore}_i(x, v)) \simeq v$ or $\text{rselect}_j(\text{rstore}_i(x, v)) \simeq \text{rselect}_j(x)$ generates clauses in categories (iii) or (iv) of Definition 46, and there are no other possible inferences. Hence, $D \in \mathcal{B}_\mathcal{R}$.

Binary inference within $\mathcal{B}_\mathcal{R}$: Suppose D is generated by a binary inference applied to $C, C' \in \mathcal{B}_\mathcal{R}$. Unordered superpositions within categories (i), (ii) and (iii) of Definition 46 generate clauses in category (i); unordered superpositions within clauses in category (iv) generate clauses in the same category. An unordered superposition from a clause in (i) and a clause in any category generates a clause in the same category. An unordered superposition between a clause in (iii) and a clause in (iv) generates a clause in (iii), and there are no other possible inferences.

This concludes the proof. □

Lemma 48. *$\mathcal{B}_\mathcal{R}$ is weakly $\mathcal{R}$-associative.*

Proof. Let C and C' be two clauses in $\mathcal{B}_\mathcal{R}$: we need to prove that C is weakly $\mathcal{R}$-associative for C', or that for all $Q \in \mathcal{R}$, C is $[C', Q]$-associative (cf. Definition 22). Let D be a clause generated by an unordered superposition of C into C': since $\mathcal{B}_\mathcal{R}$ is $\mathcal{R}$-closed, $D \in \mathcal{B}_\mathcal{R}$. We examine the inferences between D and axioms in $\mathcal{R}$. Since the only clauses in $\mathcal{B}_\mathcal{R}$ that interact with the axioms of $\mathcal{R}$ are those in category (ii) of Definition 46, we assume that D belongs to this category. In turn, since more clauses

	Clauses in $\mathcal{B}_{\mathcal{R}}$	Superpositions into $\mathcal{R}$
0.	$C' = \mathrm{rstore}_i(a, e) \simeq b \vee B$	$E'_1 = \mathrm{rselect}_i(b) \simeq e \vee B$
		$E'_2 = \mathrm{rselect}_j(b) \simeq \mathrm{rselect}_j(a) \vee B$
1.	$C = a \simeq a' \vee B'$	$E_1 = \mathrm{rselect}_i(b) \simeq e \vee B \vee B'$
	$D = \mathrm{rstore}_i(a', e) \simeq b \vee B \vee B'$	$E_2 = \mathrm{rselect}_j(b) \simeq \mathrm{rselect}_j(a') \vee B \vee B'$
2.	$C = e \simeq e' \vee B'$	$E_1 = \mathrm{rselect}_i(b) \simeq e' \vee B \vee B'$
	$D = \mathrm{rstore}_i(a, e') \simeq b \vee B \vee B'$	$E_2 = \mathrm{rselect}_j(b) \simeq \mathrm{rselect}_j(a) \vee B \vee B'$
3.	$C = b \simeq b' \vee B'$	$E_1 = \mathrm{rselect}_i(b') \simeq e \vee B \vee B'$
	$D = \mathrm{rstore}_i(a, e) \simeq b' \vee B \vee B'$	$E_2 = \mathrm{rselect}_j(b') \simeq \mathrm{rselect}_j(a) \vee B \vee B'$
4.	$C = d \simeq d' \vee B'$	$E_1 = \mathrm{rselect}_i(b) \simeq e \vee B''$
	$D = \mathrm{rstore}_i(a, e) \simeq b \vee B''$	$E_2 = \mathrm{rselect}_j(b) \simeq \mathrm{rselect}_j(a) \vee B''$

Fig. 5. Case analysis for the theory of records.

of category (ii) can be generated only by an unordered inference of a clause in category (i) into a clause in category (ii), it follows that C is in (i) and C' is in (ii), that is, C is strictly flat and C' has the form $\mathrm{rstore}_i(a, e) \simeq b \vee B$, where B is ground and strictly flat. On the other hand, we need to examine the inferences between C' and axioms in $\mathcal{R}$, and show that the clauses they generate "cover", in the sense of Definition 22, those generated by D and axioms in $\mathcal{R}$. Fig. 5 displays all possible inferences. Row 0 has C' on the left, and the clauses E'_1 and E'_2 generated by superposing C' into axioms of $\mathcal{R}$ on the right. Rows 1, 2, 3 and 4 correspond to the four possible unordered superpositions of C into C': each row displays C and the resulting D on the left, and the clauses generated by superposing D into axioms of $\mathcal{R}$ on the right. We consider the cases corresponding to Rows 1 to 4:

(1) $E'_1 \trianglelefteq E_1$ and E_2 is generated by the unordered superposition of C into E'_2.
(2) E_1 is generated by the unordered superposition of C into E'_1 and $E'_2 \trianglelefteq E_2$.
(3) E_1 and E_2 are generated by the unordered superposition of C into E'_1 and E'_2, respectively.
(4) Same as the previous case.

This proves that C is weakly $\mathcal{R}$-associative for C'; hence, $\mathcal{B}_{\mathcal{R}}$ is weakly $\mathcal{R}$-associative. □

Since all the requirements of Theorem 42 are met, we conclude:

Theorem 49. *$\mathcal{B}_{\mathcal{R}}$ is $\mathcal{R}$-stable.*

Since $\mathcal{B}_{\mathcal{R}}$ contains $S \uplus S'$ and is $\mathcal{R}$-stable, Theorem 29 and the procedure of decision by stages as in Fig. 3 apply to the theory of records. Furthermore, since $\mathcal{B}_{\mathcal{R}}$ is ground, $S \cup S'$ is also ground, and its satisfiability can be decided by a decision procedure for EUF.

5.2. The theory of integer offsets

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

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

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

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

where Ac comes from "acyclicity axioms." A crucial step to handle this theory is to justify using a finite number of such axioms (cf. Theorem 5.7 of Bonacina and Echenim (2008)):

Lemma 50 (*Bonacina and Echenim (2008)*). *If S is a set of flat ground unit clauses, then $A_{\mathcal{I}} \cup Ac \cup S \equiv_{\mathsf{s}} A_{\mathcal{I}} \cup Ac(n) \cup S$, for all n greater or equal to the number of occurrences of* s *and* p *in S.*

A similar result holds for $\mathcal{I}$-decision problems:

Proposition 51. *Let P be an $\mathcal{I}$-decision problem containing n occurrences of the function symbols* s *and* p. *Then $A_{\mathcal{I}} \cup Ac \cup P \equiv_{\mathsf{s}} A_{\mathcal{I}} \cup Ac(n) \cup P$.*

Proof. It is clear that if $A_{\mathcal{I}} \cup Ac \cup P$ is satisfiable, then so is $A_{\mathcal{I}} \cup Ac(n) \cup P$. Conversely, suppose that $A_{\mathcal{I}} \cup Ac \cup P$ is unsatisfiable, and let $S_1 \vee \cdots \vee S_k$ be a disjunctive normal form of P, where each S_i is a set of flat ground unit clauses. For all i, $1 \leq i \leq k$, $A_{\mathcal{I}} \cup Ac \cup S_i$ is unsatisfiable. Since n is greater than the number of occurrences of s and p in S_i, $A_{\mathcal{I}} \cup Ac(n) \cup S_i$ is unsatisfiable by Lemma 50. This proves that also $A_{\mathcal{I}} \cup Ac(n) \cup P$ is unsatisfiable. □

Thus, problem $\mathcal{I} \cup P$ is reduced to $A_{\mathcal{I}} \cup Ac(n) \cup P$. In order to apply decision by stages as in Fig. 3, we begin by defining decomposition:

Decomposition. Set P is decomposed into S and S' as follows: P is flattened; among the resulting clauses, all the unit clauses are added to S and all the strictly flat clauses are added to S'.

For integer offsets decomposition coincides with flattening, and it follows that

Proposition 52. *S' is $A_{\mathcal{I}} \cup Ac(n)$-disconnected.*

Following Fig. 3, $A_{\mathcal{I}} \cup Ac(n) \cup S$ is fed to a fair $\mathcal{SP}_{\succ}$-strategy. Unlike the theory of records, the axioms do not remain unaltered through the saturation. By Lemma 5.8 of Bonacina and Echenim (2008), the finite saturated limit generated by $\mathcal{SP}_{\succ}$ from an input of the form $A_{\mathcal{I}} \cup Ac(n) \cup S$ has the form $\mathcal{I}_{\mathsf{s}}[n] \cup \bar{S}$, where $\mathcal{I}_{\mathsf{s}}[n] = A_{\mathcal{I}} \cup A_{\mathsf{s}}(n)$ and $A_{\mathsf{s}}(n) = \{\mathsf{s}^i(x) \not\simeq \mathsf{p}^j(x) \mid 1 \leq i + j \leq n\}$. In other words, $\mathcal{I}_{\mathsf{s}}[n]$ is the saturated limit of $A_{\mathcal{I}} \cup Ac(n)$. Therefore, the presentation of reference for stability is $\mathcal{I}_{\mathsf{s}}[n]$: we need to identify a set $\mathcal{B}_{\mathcal{I}_{\mathsf{s}}[n]}$, that contains all strictly flat and flat unit clauses, so that it contains S and S' for all problems P, and is $\mathcal{I}_{\mathsf{s}}[n]$-stable.

Definition 53. Let $\mathcal{B}_{\mathcal{I}_{\mathsf{s}}[n]}$ denote the set of all ground clauses that are

(i) strictly flat, or have one of the following forms:
(ii) $\mathsf{p}(a) \simeq b \vee B$,
(iii) $\mathsf{s}(a) \simeq b \vee B$,
(iv) $\mathsf{s}^i(a) \not\simeq \mathsf{p}^j(b) \vee B$,

where $0 \leq i + j \leq n - 1$ and B is a possibly empty, strictly flat clause.[13]

Lemma 54. *$\mathcal{B}_{\mathcal{I}_{\mathsf{s}}[n]}$ is $\mathcal{I}_{\mathsf{s}}[n]$-neutral, $\mathcal{I}_{\mathsf{s}}[n]$-oriented and $\mathcal{I}_{\mathsf{s}}[n]$-closed.*

Proof. For neutrality, a unary inference can apply only to a clause in category (i) of Definition 53, and it generates a clause in the same category. Since the clauses in (i) are $\mathcal{I}_{\mathsf{s}}[n]$-disconnected, $\mathcal{B}_{\mathcal{I}_{\mathsf{s}}[n]}$ is $\mathcal{I}_{\mathsf{s}}[n]$-neutral. For orientation, a simple inspection shows that there are no $\mathcal{U}_{\succ}$-inferences from $\mathcal{I}_{\mathsf{s}}[n]$ clauses into $\mathcal{B}_{\mathcal{I}_{\mathsf{s}}[n]}$ clauses, so that $\mathcal{B}_{\mathcal{I}_{\mathsf{s}}[n]}$ is $\mathcal{I}_{\mathsf{s}}[n]$-oriented. For closure, the superposition or paramodulation of a clause in category (ii) into a clause in $\mathcal{I}_{\mathsf{s}}[n]$ generates clauses in categories (iii) or (iv); and the superposition or paramodulation of a clause in category (iii) into a clause in $\mathcal{I}_{\mathsf{s}}[n]$

[13] As we shall see in the following lemma, it is sufficient to take $0 \leq i + j \leq n - 1$, even if $A_{\mathsf{s}}(n)$ has $1 \leq i + j \leq n$, because paramodulations into these clauses reduce the exponent.

	Clauses in $\mathcal{B}_{\mathcal{I}_s[n]}$	Superpositions into $\mathcal{I}_s[n]$
0.	$C' = \mathsf{p}(a) \simeq b \vee B$	$E'_1 = \mathsf{s}(b) \simeq a \vee B$
		$E'_2 = \mathsf{s}^i(a) \not\simeq \mathsf{p}^j(b) \vee B$
1.	$C = a \simeq a' \vee B'$	$E_1 = \mathsf{s}(b) \simeq a' \vee B \vee B'$
	$D = \mathsf{p}(a') \simeq b \vee B \vee B'$	$E_2 = \mathsf{s}^i(a') \not\simeq \mathsf{p}^j(b) \vee B \vee B'$
2.	$C = b \simeq b' \vee B'$	$E_1 = \mathsf{s}(b') \simeq a \vee B \vee B'$
	$D = \mathsf{p}(a) \simeq b' \vee B \vee B'$	$E_2 = \mathsf{s}^i(a) \not\simeq \mathsf{p}^j(b') \vee B \vee B'$
3.	$C = c \simeq c' \vee B'$	$E_1 = \mathsf{s}(b) \simeq a \vee B''$
	$D = \mathsf{p}(a) \simeq b \vee B''$	$E_2 = \mathsf{s}^i(a) \not\simeq \mathsf{p}^j(b) \vee B''$

Fig. 6. Case analysis for the theory of integer offsets.

generates clauses in categories (ii) or (iv). Unordered superpositions within categories (i), (ii) and (iii), respectively, generate clauses in category (i). An unordered superposition or paramodulation from a clause in (i) into a clause in any category generates a clause in the same category. An unordered paramodulation from a clause in (ii) or (iii) into one in (iv) generates a clause in (iv) and there are no other possible inferences. □

Lemma 55. *$\mathcal{B}_{\mathcal{I}_s[n]}$ is weakly $\mathcal{I}_s[n]$-associative.*

Proof. Let C and C' be two clauses in $\mathcal{B}_{\mathcal{I}_s[n]}$: we prove that for all $Q \in \mathcal{I}_s[n]$, C is $[C', Q]$-associative (cf. Definition 22). Let D be a clause generated by an unordered superposition of C into C': since $\mathcal{B}_{\mathcal{I}_s[n]}$ is $\mathcal{I}_s[n]$-closed, $D \in \mathcal{B}_{\mathcal{I}_s[n]}$. We need to consider the inferences between D and clauses in $\mathcal{I}_s[n]$. For D to superpose or paramodulate into a clause of $\mathcal{I}_s[n]$, D must belong to categories (ii) or (iii) of Definition 53. In turn, the only unordered superpositions that generate more clauses in these categories are unordered superpositions of strictly flat clauses into clauses in (ii) or (iii). Thus, C is in (i) and C' is in either (ii) or (iii). We consider the case where C' is in (ii), that is, it has the form $\mathsf{p}(a) \simeq b \vee B$, where B is ground and strictly flat (the case where C' is in (iii) is similar). Its superposition into $\mathsf{s}(\mathsf{p}(x)) \simeq x$ generates $E'_1 = \mathsf{s}(b) \simeq a \vee B$, and its superposition into $\mathsf{s}^i(x) \not\simeq \mathsf{p}^{j+1}(x)$, where $1 \leq i + (j+1) \leq n$, generates $E'_2 = \mathsf{s}^i(a) \not\simeq \mathsf{p}^j(b) \vee B$. Row 0 in Fig. 6 displays C' on the left, and E'_1 and E'_2 on the right. We need to show that E'_1 and E'_2 "cover", in the sense of Definition 22, the clauses generated by D and axioms in $\mathcal{I}_s[n]$. Rows 1, 2 and 3 in Fig. 6 display all possible inferences. Each row shows on the left the form of C and the D resulting from the unordered superposition of C into C', and on the right the clauses E_1 and E_2, generated by superposing D into $\mathsf{s}(\mathsf{p}(x)) \simeq x$ and $\mathsf{s}^i(x) \not\simeq \mathsf{p}^{j+1}(x)$, respectively. For all three cases corresponding to Rows 1, 2 and 3, E_1 and E_2 are generated by unordered superpositions of C into E'_1 and E'_2, respectively. □

Thus, by Theorem 42, we have:

Theorem 56. *$\mathcal{B}_{\mathcal{I}_s[n]}$ is $\mathcal{I}_s[n]$-stable.*

Similar to the theory of records, since $\mathcal{B}_{\mathcal{I}_s[n]}$ is ground, the satisfiability of $\bar{S} \cup S'$ can be decided by a decision procedure for EUF.

5.3. The theory of arrays

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

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

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

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

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

$\mathcal{A}^{\mathrm{e}}$-satisfiability problems can be reduced to $\mathcal{A}$-satisfiability problems, by replacing every 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 (Armando et al., 2003). Since this reduction also holds for sets of ground clauses, we consider only $\mathcal{A}$-decision problems. Let P be such a problem. Similar to the theory of records, Decomposition puts in S and therefore submits to the $\mathcal{SP}_{\succ}$-strategy the unit clauses where "store" occurs:

Decomposition. Set P is decomposed into S and S' as follows: P is flattened; among the resulting clauses, all the unit clauses of the form $\mathrm{store}(a, i, e) \simeq a'$ go in S and all other clauses in S'.

Proposition 57. *S' is $\mathcal{A}$-disconnected.*

Proof. Since clauses in S' are strictly flat or have the form $\mathrm{select}(a, i) \simeq e$, no $\mathcal{SP}_{\succ}$-inferences apply to a clause in S' and one in $\mathcal{A}$. □

By Lemma 14 of Armando et al. (2009), the saturation of $\mathcal{A} \cup S$ by $\mathcal{SP}_{\succ}$ (first stage in Fig. 3) produces a finite limit $\mathcal{A} \cup \bar{S}$. In order to establish that $\mathcal{A} \cup P$ and $\bar{S} \cup S'$ are equisatisfiable, we need to identify a set $\mathcal{B}_{\mathcal{A}}$ that contains S and S' for all problems P, and is $\mathcal{A}$-stable:

Definition 58. Let $\mathcal{B}_{\mathcal{A}}$ be the set of all clauses that are

(i) ground and strictly flat, or have one of the following forms:
(ii) $\mathrm{store}(a, i, e) \simeq a' \vee B$,
(iii) $\mathrm{select}(a, i) \simeq e \vee B$, or
(iv) $\mathrm{select}(a, x) \simeq \mathrm{select}(a', x) \vee x \simeq i_1 \vee \cdots \vee x \simeq i_n \vee B$,

where $n \geq 0$ and B is a possibly empty, ground, strictly flat clause.

Unlike $\mathcal{B}_{\mathcal{R}}$ (cf. Definition 46) and $\mathcal{B}_{\mathcal{I}_{\mathrm{s}}[n]}$ (cf. Definition 53), $\mathcal{B}_{\mathcal{A}}$ is not ground, because of clauses in category (iv).

Lemma 59. *$\mathcal{B}_{\mathcal{A}}$ is $\mathcal{A}$-neutral, $\mathcal{A}$-oriented and $\mathcal{A}$-closed.*

Proof. It is plain to see that $\mathcal{B}_{\mathcal{A}}$ is $\mathcal{A}$-oriented (cf. Definition 18), since it does not contain occurrences of the select-over-store pattern. For neutrality (cf. Definition 16), the only unary inferences apply to clauses in category (i) of Definition 58, and they generate clauses in the same category. Since these clauses are $\mathcal{A}$-disconnected, $\mathcal{B}_{\mathcal{A}}$ is $\mathcal{A}$-neutral. For closure, the only superpositions that can take place between a clause in $\mathcal{B}_{\mathcal{A}}$ and an axiom in $\mathcal{A}$ are superpositions of a clause in category (ii) into one of category (3) or (4), and they generate clauses in categories (iii) and (iv). Unordered superpositions within categories (i), (ii) and (iii), respectively, all generate clauses in (i), while an unordered superposition within category (iv) generates a clause in the same category. An unordered superposition from a clause in (i) and a clause in any category generates a clause in the same category. An unordered superposition between a clause in (iii) and one in (iv) generates a clause in (iii), and there are no other possible inferences. □

Lemma 60. *$\mathcal{B}_{\mathcal{A}}$ is weakly $\mathcal{A}$-associative.*

Proof. Let C and C' be two clauses in $\mathcal{B}_{\mathcal{A}}$: we prove that for all $Q \in \mathcal{A}$, C is $[C', Q]$-associative (cf. Definition 22). Let D be a clause generated by an unordered superposition of C into C': since $\mathcal{B}_{\mathcal{A}}$ is $\mathcal{A}$-closed, $D \in \mathcal{B}_{\mathcal{A}}$. We need to cover the inferences between D and axioms in $\mathcal{A}$. If D can superpose into an axiom, it means that D belongs to category (ii) of Definition 58. Since the only inferences that generate clauses in (ii) are unordered superpositions of strictly flat clauses into clauses of (ii), it follows that C is strictly flat and C' is in category (ii). C' is shown on the left in Row 0 of Fig. 7. The possible shapes of C are listed on the left in Rows 1, 2, 3, 4 and 5 of Fig. 7: each row shows C and the D generated by unordered superposition of C into C'. In Row 5, d is a constant that appears in B and B'' is the union

	Clauses in $\mathcal{B}_{\mathcal{A}}$	Superpositions into $\mathcal{A}$
0.	$C' = \text{store}(a, i, e) \simeq b \vee B$	$E'_1 = \text{select}(b, i) \simeq e \vee B$
		$E'_2 = \text{select}(b, w) \simeq \text{select}(a, w) \vee w \simeq i \vee B$
1.	$C = a \simeq a' \vee B'$	$E_1 = \text{select}(b, i) \simeq e \vee B \vee B'$
	$D = \text{store}(a', i, e) \simeq b \vee B \vee B'$	$E_2 = \text{select}(b, w) \simeq \text{select}(a', w) \vee w \simeq i \vee B \vee B'$
2.	$C = i \simeq i' \vee B'$	$E_1 = \text{select}(b, i') \simeq e \vee B \vee B'$
	$D = \text{store}(a, i', e) \simeq b \vee B \vee B'$	$E_2 = \text{select}(b, w) \simeq \text{select}(a, w) \vee w \simeq i' \vee B \vee B'$
3.	$C = e \simeq e' \vee B'$	$E_1 = \text{select}(b, i) \simeq e' \vee B \vee B'$
	$D = \text{store}(a, i, e') \simeq b \vee B \vee B'$	$E_2 = \text{select}(b, w) \simeq \text{select}(a, w) \vee w \simeq i \vee B \vee B'$
4.	$C = b \simeq b' \vee B'$	$E_1 = \text{select}(b', i) \simeq e \vee B \vee B'$
	$D = \text{store}(a, i, e) \simeq b' \vee B \vee B'$	$E_2 = \text{select}(b', w) \simeq \text{select}(a, w) \vee w \simeq i \vee B \vee B'$
5.	$C = d \simeq d' \vee B'$	$E_1 = \text{select}(b, i) \simeq e \vee B''$
	$D = \text{store}(a, i, e) \simeq b \vee B''$	$E_2 = \text{select}(b, w) \simeq \text{select}(a, w) \vee w \simeq i \vee B''$

Fig. 7. Case analysis for the theory of arrays.

of B' and the result of the unordered superposition of $d \simeq d'$ into B. The superposition of C' into the axioms in $\mathcal{A}$ generates E'_1 and E'_2, displayed on the right in Row 0 of Fig. 7. We need to show that E'_1 and E'_2 "cover", in the sense of Definition 22, the clauses generated by D and axioms of $\mathcal{A}$. Rows 1, 2, 3, 4 and 5 in Fig. 7 show on the right clauses E_1 and E_2, generated by superposing D into the axioms in $\mathcal{A}$:

- **Row 1**: $E'_1 \trianglelefteq E_1$ and E_2 is generated by unordered superposition of C into E'_2.
- **Rows 2, 4 and 5**: E_1 and E_2 are generated by unordered superposition of C into E'_1 and E'_2, respectively.
- **Row 3**: E_1 is generated by unordered superposition of C into E'_1 and $E'_2 \trianglelefteq E_2$.

Thus, C is weakly $\mathcal{A}$-associative for C'; hence, $\mathcal{B}_{\mathcal{A}}$ is weakly $\mathcal{A}$-associative. □

By Theorem 42, it follows that:

Theorem 61. *$\mathcal{B}_{\mathcal{A}}$ is $\mathcal{A}$-stable.*

Since $\mathcal{B}_{\mathcal{A}}$ is not ground, neither is $\bar{S}$, and deciding the satisfiability of $\bar{S} \cup S'$, in the scheme of Fig. 3, requires more than a decision procedure for EUF. The non-ground clauses are those in category (iv) of Definition 58. We observe that these clauses belong to the *array property fragment* defined in Section 11.1 of Bradley and Manna (2007). Thus, it is sufficient to apply a decision procedure for that fragment.[14]

In summary, in order to solve by stages SMT problems in the theories of *records*, *integer offsets* and *arrays*, it suffices to pipeline an $\mathcal{SP}$-based theorem prover with an SMT-solver capable of handling EUF, for records and integer offsets, or the array property fragment for arrays. The prover and the solver can be taken *off the shelf* and combined *off-line*: the prover is invoked *only once* to compile the theory, and there is no complex integration issue to deal with.

[14] The procedure given in Section 11.1 of Bradley and Manna (2007) and based on Bradley (2007) is for arrays with infinite domain, as noted at http://theory.stanford.edu/~arbrad/pivc/book.html, viewed on March 20, 2009. Another source for extensions of the theory of arrays is Ghilardi et al. (2007).

6. A decision procedure for theories of partial functional equations

In this section, we present a decision procedure for *theories of partial functional equations*, that yields an alternative decision procedure[15] for problems including clauses of the form of category (iv) of Definition 58. This alternative approach reduces such problems to ground problems in EUF, so that it is not necessary to have an SMT-solver that features a decision procedure for the array property fragment, and an SMT-solver capable of handling EUF suffices also for the theory of arrays.

The notion of *partial equations* was defined in Stump et al. (2001) as follows:

$$a \simeq_\Delta a' \Leftrightarrow \left[\left(\bigwedge_{i\in\Delta} x \not\simeq i\right) \rightarrow \text{select}(a, x) \simeq \text{select}(a', x)\right] \tag{6}$$

where Δ is a set of indices, and the partial equation states that arrays a and a' are equal everywhere, except possibly on indices in Δ. Clauses in category (iv) of Definition 58 are given by the disjunction of a partial equation and ground strictly flat literals.

We generalize partial equations to *partial functional equations*, by replacing indices with ground terms and arrays with functions, according to the well-known notion that (monodimensional) arrays can be seen as representation of (unary) functions:

$$f \simeq_\Delta f' \Leftrightarrow \left[\left(\bigwedge_{s\in\Delta} x \not\simeq s\right) \rightarrow f(x) \simeq f'(x)\right] \tag{7}$$

where Δ is a set of ground terms, and the partial equation states that functions f and f' are equal everywhere, except possibly on arguments in Δ.

Therefore, the pattern of category (iv) of Definition 58 can be replaced by the more general pattern

$$\left(\bigvee_{s\in\Delta} x \simeq s\right) \vee f(x) \simeq f'(x) \vee B, \tag{8}$$

where f and f' are unary functions, Δ is a (possibly empty) set of ground terms and B is a ground clause. In this section we define a decision procedure for a theory presented by a set $\mathcal{PE}$ of axioms of form (8), setting aside the motivation originated with the theory of arrays, to which we shall return later. Adopting definition (7), presentation $\mathcal{PE}$ can be written as $\{f_i \simeq_{\Delta_i} f'_i \vee B_i\}_{i=1}^m = \{\mathrm{P}_i(x)\}_{i=1}^m$, where each axiom $\mathrm{P}_i(x)$ contains exactly one distinct variable x. Let $\mathrm{P}_i(s)$ represent its instance with x replaced by term s. Given a set Θ of ground terms, the *set of Θ-instances* of $\mathcal{PE}$ is

$$\mathcal{GI}_{[\Theta]} = \bigcup_{i=1}^{m} \bigcup_{s\in\Theta} \{\mathrm{P}_i(s)\}.$$

Example 62. If $\Delta_1 = \{a, g(b)\}$ and $B_1 = a \simeq c \vee c \not\simeq d$, then $\mathrm{P}_1(x)$ is the clause $x \simeq a \vee x \simeq g(b) \vee f_1(x) \simeq f'_1(x) \vee a \simeq c \vee c \not\simeq d$. If $\mathcal{PE} = \{\mathrm{P}_1(x)\}$ and $\Theta = \{f(a), b\}$, then $\mathcal{GI}_{[\Theta]}$ contains $f(a) \simeq a \vee f(a) \simeq g(b) \vee f_1(f(a)) \simeq f'_1(f(a)) \vee a \simeq c \vee c \not\simeq d$ and $b \simeq a \vee b \simeq g(b) \vee f_1(b) \simeq f'_1(b) \vee a \simeq c \vee c \not\simeq d$.

Given a set of ground clauses S in the signature of $\mathcal{PE}$, the idea of the decision procedure is to determine a set X such that $\mathcal{PE} \cup S \equiv_s \mathcal{GI}_{[X]} \cup S$, and decide the satisfiability of $\mathcal{GI}_{[X]} \cup S$ by a decision procedure for EUF.

Let $\Sigma_u \subseteq \Sigma$ be the set of unary function symbols $\bigcup_{i=1}^m \{f_i, f'_i\}$ that appear in non-ground terms in $\mathcal{PE}$. Given term t, let $GSub(t)$ denote the set of its ground subterms. This notation extends naturally to literals, clauses and sets of clauses. For a set S of ground clauses, let Γ_S be the set of ground subterms

[15] Although we were inspired to some extent by the *instantiation scheme* technique of Bradley et al. (2006), later renamed *quantifier instantiation* technique in Chapter 11 of Bradley and Manna (2007), we obtained this decision procedure independently of Bradley and Manna (2007).

that appear in $\mathcal{PE} \cup S$ as arguments of an $f \in \Sigma_u$:

$$\Gamma_S = \{t \mid f \in \Sigma_u \wedge f(t) \in GSub(\mathcal{PE} \cup S)\}.$$

By construction, $\Gamma_S \subseteq GSub(\mathcal{PE} \cup S)$.

Example 63. Let $\mathcal{PE} = \{\mathrm{P}_1(x)\}$, where $\mathrm{P}_1(x)$ is as in Example 62, and $S = \{f_1(b) \simeq a\}$. Then $GSub(\mathcal{PE} \cup S) = \{a, b, c, d, g(b), f_1(b)\}$ and $\Gamma_S = \{b\}$.

The next Proposition shows that Γ_S satisfies a closure property if axioms $\mathrm{P}_i(x)$ are instantiated with its terms:

Proposition 64. *If $r \in \Gamma_S$, $f(t) \in GSub(\mathrm{P}_i(r))$ for some i, $1 \leq i \leq m$, and $f \in \Sigma_u$, then $t \in \Gamma_S$.*

Proof. Since $f(t) \in GSub(\mathrm{P}_i(r))$, either $f(t)$ already appears among the ground subterms of $\mathrm{P}_i(x)$, that is, $f(t) \in GSub(\mathrm{P}_i(x))$, or not. If it does, since $\mathrm{P}_i(x) \in \mathcal{PE}$, $GSub(\mathrm{P}_i(x)) \subseteq GSub(\mathcal{PE} \cup S)$ and $f \in \Sigma_u$, it follows that $t \in \Gamma_S$. If not, either $f(t) \in GSub(r)$ or $f(t) \in \{f_i(r), {f'}_i(r)\}$. If $f(t) \in GSub(r)$, then $f(t) \in GSub(\mathcal{PE} \cup S)$, since $r \in \Gamma_S$ and $\Gamma_S \subseteq GSub(\mathcal{PE} \cup S)$, so that $t \in \Gamma_S$. Otherwise, t is r and the result holds by hypothesis. □

The central result is to show that Γ_S is precisely the set we are looking for, that is:

Theorem 65. $\mathcal{PE} \cup S \equiv_s \mathcal{GI}_{[\Gamma_S]} \cup S$.

Proof. If $\mathcal{PE} \cup S$ is satisfiable, then so is $\mathcal{GI}_{[\Gamma_S]} \cup S$. For the other direction, assume that $M = (D, I)$ is a model of $\mathcal{GI}_{[\Gamma_S]} \cup S$, with domain D and interpretation function I, and let d be any element of D. For $f \in \Sigma$, f^I is the interpretation of f according to I, and $I(t)$ is the interpretation of a term t according to I. We construct a model N of $\mathcal{PE} \cup S$ starting from M. Let $N = (D, J)$ be the interpretation defined as follows:

$$\forall f \in \Sigma \setminus \Sigma_u, \quad f^J = f^I,$$
$$\forall f \in \Sigma_u, \forall v \in D, \quad f^J(v) = \begin{cases} f^I(v) & \text{if } \exists t \in \Gamma_S \text{ such that } I(t) = v, \\ d & \text{otherwise.} \end{cases}$$

In other words, N interprets functions in Σ_u like M on the elements of the domain reached by terms in Γ_S, and as constant functions elsewhere. To prove that $N \models \mathcal{PE} \cup S$, we begin by showing that I and J interpret terms appearing in $\mathcal{GI}_{[\Gamma_S]}$ and S in the same way, that is, for all $t \in GSub(\mathcal{GI}_{[\Gamma_S]} \cup S)$, $I(t) = J(t)$. The reasoning is by structural induction. If t is a constant, $I(t) = J(t)$ by construction of J. Let $t = f(s_1, \ldots, s_n)$. The induction hypothesis is that for all i, $1 \leq i \leq n$, $I(s_i) = J(s_i)$. There are two cases:

(1) If $f \in \Sigma \setminus \Sigma_u$, then $J(t) = f^J(J(s_1), \ldots, J(s_n)) = f^I(J(s_1), \ldots, J(s_n)) = f^I(I(s_1), \ldots, I(s_n)) = I(t)$, by construction of N and induction hypothesis.
(2) If $f \in \Sigma_u$, then f is unary. If $f(s_1) \in GSub(S)$, then $s_1 \in \Gamma_S$ by definition of Γ_S. Otherwise, there exists an i, $1 \leq i \leq m$, and an $r \in \Gamma_S$, such that $f(s_1) \in GSub(\mathrm{P}_i(r))$. By Proposition 64, s_1 is also in Γ_S. Thus, in both cases, $J(t) = f^J(J(s_1)) = f^J(I(s_1)) = f^I(I(s_1)) = I(t)$ by induction hypothesis and construction of N.

Since I and J interpret the terms appearing in S in the same way and $M \models S$, it follows that $N \models S$. We are left to show that for all i, $1 \leq i \leq m$, $N \models \forall x.\, \mathrm{P}_i(x)$. We need to show that for all $v \in D$, if $N\{x \leftarrow v\}$ denotes the interpretation that maps every occurrence of x to v and is identical to N otherwise, then $N\{x \leftarrow v\} \models \mathrm{P}_i(x)$. Suppose first that there is a $t \in \Gamma_S$ such that $I(t) = v$. Since $M \models \mathcal{GI}_{[\Gamma_S]} \cup S$, it follows that $M \models \mathrm{P}_i(t)$. Since $GSub(\mathrm{P}_i(t)) \subseteq GSub(\mathcal{GI}_{[\Gamma_S]})$, $\mathrm{P}_i(t)$ is ground, and I and J interpret the terms appearing in $\mathcal{GI}_{[\Gamma_S]}$ in the same way, $N \models \mathrm{P}_i(t)$. Since x is the only variable appearing in $\mathrm{P}_i(x)$, we conclude that $N\{x \leftarrow v\} \models \mathrm{P}_i(x)$. Otherwise, suppose that there is no $t \in \Gamma_S$ such that $I(t) = v$. Then by construction of J, $f_i^J(v) = d = {f'}_i^J(v)$, so that $N\{x \leftarrow v\} \models \mathrm{P}_i(x)$, which completes the proof. □

Thus, an SMT problem $\mathcal{PE} \cup S$ can be reduced to an EUF problem, by replacing each non-ground clause in $\mathcal{PE} \cup S$ with the finite set of its instances, where variables are substituted with ground terms that appear as arguments of an $f \in \Sigma_u$ in $\mathcal{PE} \cup S$.

We conclude this section by showing how this reduction works for a set $\bar{S} \subseteq \mathcal{B}_{\mathcal{A}}$ generated by applying decision by stages to a problem in the theory of arrays. First, one introduces a new unary function symbol f_a for all constant symbols $a \in \Sigma$ that denote an array, and replaces all terms $\text{select}(a, t)$ in $\bar{S}$ by $f_a(t)$. Thus, $\bar{S} \subseteq \mathcal{B}_{\mathcal{A}}$ is transformed into a set $\mathcal{PE} \cup G$, where G is a set of ground clauses. Then, one can determine the set of ground terms Γ_G, instantiate the axioms in $\mathcal{PE}$ accordingly, and reduce the problem to an EUF problem according to Theorem 65.

Example 66. Assume that we have to decide the satisfiability modulo $\mathcal{A}$ of

$$P = \{\text{store}(b, i, d) \simeq a,\ \text{select}(b, i') \simeq e,\ \text{select}(c, i') \simeq e',\ a \simeq c,\ i \not\simeq i',\ e \not\simeq e'\}.$$

Note that P is unsatisfiable. First, since we are reasoning in the theory without extensionality, the equality between arrays $a \simeq c$ is replaced by the partial equation (with empty set of indices) $\text{select}(a, x) \simeq \text{select}(c, x)$. Then, the set is decomposed into

$$S = \{\text{store}(b, i, d) \simeq a\}$$
$$S' = \{\text{select}(b, i') \simeq e,\ \text{select}(c, i') \simeq e',\ \text{select}(a, x) \simeq \text{select}(c, x),\ i \not\simeq i',\ e \not\simeq e'\}.$$

S' is not ground, but this is irrelevant, given its passive role at this stage. A fair $\mathcal{SP}_{\succ}$-strategy applied to $\mathcal{A} \cup S$ generates the limit $S_\infty = \mathcal{A} \cup \bar{S}$, where

$$\bar{S} = \{\text{store}(b, i, d) \simeq a,\ \text{select}(a, i) \simeq d,\ x \simeq i \vee \text{select}(a, x) \simeq \text{select}(b, x)\},$$

and, by Theorem 29, $\mathcal{A} \cup S \cup S' \equiv_{\text{s}} \bar{S} \cup S'$. Since the symbol store does not occur in S', the clause $\text{store}(b, i, d) \simeq a$ can be discarded from $\bar{S} \cup S'$. By introducing unary function symbols for each array, $\bar{S} \cup S'$ is transformed into a problem $\mathcal{PE} \cup G$, where

$$\mathcal{PE} = \{f_a(x) \simeq f_c(x),\ x \simeq i \vee f_a(x) \simeq f_b(x)\}$$
$$G = \{f_a(i) \simeq d,\ f_b(i') \simeq e,\ f_c(i') \simeq e',\ i \not\simeq i',\ e \not\simeq e'\}.$$

Then $\Sigma_u = \{f_a, f_b, f_c\}$, and $\Gamma_G = \{i, i'\}$, since i and i' are the only ground terms occurring as arguments of functions in Σ_u. By instantiating the clauses in $\mathcal{PE}$ with the terms in Γ_G and removing tautologies, one generates the ground set

$$\{f_a(i) \simeq f_c(i),\ f_a(i') \simeq f_c(i'),\ i \simeq i' \vee f_a(i') \simeq f_b(i')\} \cup$$
$$\{f_a(i) \simeq d,\ f_b(i') \simeq e,\ f_c(i') \simeq e',\ i \not\simeq i',\ e \not\simeq e'\},$$

which can be shown unsatisfiable by a decision procedure for EUF.

As suggested in the example, the set of ground terms needed to instantiate the non-ground clauses merely consists of all constants denoting indices and appearing as arguments of "select". Indeed, when the reduction for problems in theories of partial functional equations is applied in the context of the theory of arrays, the unary functions in Σ_u are those introduced to replace "select" terms, the ground terms occurring as their arguments denote indices, and since the problem was flattened, they are all constants. Thus, our decision procedure can use *fewer indices*, and therefore generate *fewer instances*, than that of Section 11.1 of Bradley and Manna (2007):

Example 67. Let $\mathcal{PE}$ be given by the axiom $(\bigwedge_{j=1}^{n} x \not\simeq i_j) \rightarrow \text{select}(a, x) \simeq \text{select}(a', x)$, and consider $S = \{\text{select}(a, i) \simeq e,\ \text{select}(a', i) \not\simeq e\} \cup \{i \not\simeq i_j \mid j = 1, \ldots, n\}$, such that $\mathcal{PE} \cup S$ is unsatisfiable. According to our procedure, the set Γ_S is the singleton $\{i\}$. On the other hand, the procedure of Section 11.1 of Bradley and Manna (2007) uses $\{\lambda, i_1, \ldots, i_n, i\}$, because it picks all indices appearing in the antecedents of partial equations, that is, $i_1, \ldots, i_n$, all indices appearing as arguments of select( ), hence i, and the special constant λ.

The purposes of the two procedures are different: the goal of the one in Section 11.1 of Bradley and Manna (2007) is to decide the entire array property fragment. We focused on the non-ground clauses in $\bar{S} \cup S'$, because the decomposition approach is to do as much work as possible by generic $\mathcal{SP}_{\succ}$-inferences, and, if necessary, post-process the output of the $\mathcal{SP}_{\succ}$-strategy to submit to the SMT-solver as little theory knowledge as possible.

7. Combination of specific theories

We conclude by addressing $\mathcal{T}$-decision problems that combine records, integer offsets, arrays and possibly other theories, by showing that the two-theories scheme of Section 4.3 applies to any two theories $\mathcal{T}_1$ and $\mathcal{T}_2$ among $\mathcal{R}$, $\mathcal{I}$ and $\mathcal{A}$. Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be the corresponding $\mathcal{T}_i$-stable sets, for $i, j \in \{1, 2\}$, as determined in Section 5, and $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$. By Theorems 32 and 35, it suffices to prove that $\mathcal{B}_1 \cup \mathcal{B}_2$ is $\mathcal{T}$-stable. In turn, by Theorem 43, all we need is that for $i, j \in \{1, 2\}$, $\mathcal{B}_i$ is $\mathcal{T}_j$-disconnected (cf. Definition 12), and whenever there is a binary $\mathcal{U}_\succ$-inference with $C \in \mathcal{B}_i$ as first parent and $C' \in \mathcal{B}_j$ as second parent, then $C \in \mathcal{B}_j$. We observe that $\mathcal{B}_\mathcal{R}$ (cf. Definition 46) and $\mathcal{B}_\mathcal{A}$ (cf. Definition 58) fulfill these requirements:

Proposition 68. *$\mathcal{B}_\mathcal{R}$ is $\mathcal{A}$-disconnected and $\mathcal{B}_\mathcal{A}$ is $\mathcal{R}$-disconnected.*

Proof. Trivial, since $\mathcal{R}$ and $\mathcal{A}$ have disjoint signatures and we do not paramodulate into variables. □

Lemma 69. *If there is a binary $\mathcal{U}_\succ$-inference with $C \in \mathcal{B}_\mathcal{R}$ ($C \in \mathcal{B}_\mathcal{A}$) as first parent and $C' \in \mathcal{B}_\mathcal{A}$ ($C' \in \mathcal{B}_\mathcal{R}$) as second parent, then $C \in \mathcal{B}_\mathcal{A}$ ($C \in \mathcal{B}_\mathcal{R}$).*

Proof. If a maximal term in a maximal literal of a clause in $\mathcal{B}_\mathcal{R}$ ($\mathcal{B}_\mathcal{A}$) unifies with a term occurring in a clause in $\mathcal{B}_\mathcal{A}$ ($\mathcal{B}_\mathcal{R}$), then this maximal term must be a constant, and, by Proposition 4, C must be strictly flat, hence $C \in \mathcal{B}_\mathcal{A}$ ($C \in \mathcal{B}_\mathcal{R}$). □

Similarly, these properties hold if either $\mathcal{B}_\mathcal{R}$ or $\mathcal{B}_\mathcal{A}$ is replaced by $\mathcal{B}_{\mathcal{I}_s[n]}$, so that:

Theorem 70. *The two-theories scheme is correct for any combination of the theories of records, integer offsets and arrays.*

In Armando et al. (2009), theories are combined by uniting their presentations in the input of the $\mathcal{SP}$-engine. Modularity of termination (cf. Section 1.2) follows from the observation that the only inferences across theories are paramodulations from constants into constants, that are finitely many, while paramodulations from compound terms are excluded by the disjoint signatures assumption and paramodulations from variables are excluded by variable-inactivity (cf. Definitions 14 and 15 in Armando et al. (2009): no persistent clause features a maximal literal $x \simeq t$ where x is not a variable of t). In the two-theories scheme, theories are combined by uniting the respective outputs of the $\mathcal{SP}$-engine (cf. Fig. 4), and by similarly showing that the only persistent inferences across theories are paramodulations from constants into constants, as done for instance in Lemma 69.

The added value of the two-theories scheme is the possibility of feeding some theories directly to the SMT-solver, which is relevant for theories, such as bitvectors or linear arithmetic, that are better handled by specialized engines. Since SMT-solvers usually combine theories by the Nelson–Oppen method, we show how to interface decision by stages with Nelson–Oppen combination (cf. Section 1.1). The theories of records, integer offsets and arrays are stably infinite and do not share symbols other than equality.[16] We assume that any other theory to be combined satisfies these assumptions.[17] Then, all we need to show is that every disjunction of equalities between constants, that would be entailed, and therefore propagated, by the initial formulation of the problem, is still entailed, and therefore can be propagated, by the formulation of the problem produced by the theorem prover and given in input to the SMT-solver.

Let $\mathcal{T}$ be any theory amenable to decision by stages. A given $\mathcal{T}$-decision problem P is decomposed into S and S', in such a way that $S, S' \subseteq \mathcal{B}$, and $\bar{S}$ is generated by a fair $\mathcal{SP}_\succ$-strategy applied to $\mathcal{T} \cup S$ (cf. Fig. 3). Then, we have:

Theorem 71. *If C is a disjunction of equalities between constants, then $\mathcal{T} \cup S \cup S' \models C$ if and only if $\bar{S} \cup S' \models C$.*

Proof. C is a ground, strictly flat clause. Its negation $\neg C$ is a set of ground (unit) strictly flat clauses. Let G be $\neg C$, so that $\neg G$ is the set of ground, strictly flat clauses that contains only C. We prove the

16 The theories of records and integer offsets are also convex, while the theory of arrays is not.

17 By Lemma 5.2 in Bonacina et al. (2006) a fair $\mathcal{SP}_\succ$-strategy may discover that $\mathcal{T}$ is not stably infinite by deriving a cardinality constraint from $\mathcal{T} \cup S$ in the first stage.

result by applying Corollary 30 with set A instantiated to $\bar{S}$. We check that the hypotheses are satisfied: $S, G \subseteq \mathcal{B}$ holds, because also $G \subseteq \mathcal{B}$, since $\mathcal{B}$ includes all ground strictly flat clauses; $\mathcal{T} \cup S \models \bar{S}$ holds by construction of $\bar{S}$, which is $\mathcal{T}$-congruous with S by Proposition 38; G is $\mathcal{T}$-disconnected, because all ground, strictly flat clauses are, and therefore $\mathcal{T}$-congruous with itself by Proposition 13. Then we have $\mathcal{T} \cup S \models \neg G$ if and only if $\bar{S} \models \neg G$, hence $\mathcal{T} \cup S \models C$ if and only if $\bar{S} \models C$, whence $\mathcal{T} \cup S \cup S' \models C$ if and only if $\bar{S} \cup S' \models C$. □

This result shows that $\bar{S} \cup S'$ is *deduction complete* in the sense of Kirchner et al. (2006), meaning that it entails any ground equation entailed by $\mathcal{T} \cup S \cup S'$.

8. Discussion

Generic theorem provers based on resolution are traditionally strong at reasoning about equality and universally quantified variables, thanks to rewriting and unification, respectively. SMT-solvers are strong at reasoning about disjunction, thanks to the case analysis in DPLL, and theories such as linear arithmetic, thanks to their capacity to embed specialized algorithms such as the simplex method. In this paper we presented an approach to combine the strengths of both kinds of reasoners.

We introduced a *decomposition principle* to decompose a $\mathcal{T}$-decision problem and solve it by pipelining a generic theorem prover and an SMT-solver. The idea is that the theorem prover compiles the theory and generates a transformed problem, that requires no or less theory reasoning, and is fed to the solver. Thus, the prover does most of the reasoning about the equalities and the universally quantified variables in the theory axioms, whereas the solver does most of the reasoning about propositional logic and ground equalities. Furthermore, the decomposition principle applies to combinations of theories, and in such a way that theories that are best handled by algorithmic reasoning, such as linear arithmetic or bitvectors, can be passed on directly to the solver. Dually, non-standard theories can be dealt with by the prover, saving the effort of implementing dedicated decision procedures for each one of them. Correctness of the decomposition rests on a collection of sufficient conditions, termed $\mathcal{T}$*-stability*, that we developed for a completely generic inference system, that employs a well-founded ordering and is refutationally complete for a supported strategy.

In the second part of the paper, we demonstrated how to realize the decomposition principle in a *decision by stages mechanism*, where theory compilation is saturation with respect to the rewrite-based inference system $\mathcal{SP}$. Then we showed how to apply decision by stages to $\mathcal{T}$-decision problems in the theories of *records with or without extensionality*, *integer offsets*, *arrays with or without extensionality* and their combinations, possibly with more theories. Theory compilation can be viewed also as *reduction*: the theories of records and integer offsets are reduced to that of equality, and the theory of arrays to theories of *partial functional equations*, with axioms stating that some uninterpreted functions are equal everywhere except on a given domain. An instantiation-based decision procedure that reduces these theories as well to that of equality was given.

Directions for future work include experimenting with decision by stages to evaluate its efficiency in practice. Since no tight integration is required, state-of-the-art $\mathcal{SP}$-based provers and SMT-solvers can be taken "off the shelf". A complementary direction is to inquire whether our theoretical study can be applied to ensure refutational completeness and termination of a tight integration such as those in de Moura and Bjørner (2008a) and Lynch and Tran (2008) without inference bounds. Because our framework is generic, it can be instantiated to other inference systems or applied to more theories. Since in verification one is interested mostly in showing that formulæ are satisfiable and in exhibiting their models, an important line of research is to study whether decision by stages offers a suitable framework to improve the model building capabilities of SMT-solvers, especially in combinations of theories, by taking advantage of the saturation done in the first stage.

Acknowledgements

The authors thank the referees for their comments and suggestions. The bulk of this work was done when both authors were with the Dipartimento di Informatica of the Università degli Studi di Verona and it appeared in preliminary form in Bonacina and Echenim (2007c).

References

Arkoudas, K., Zee, K., Kuncak, V., Rinard, M., 2004. Verifying a file system implementation. In: Davies, J., Schulte, W., Barnett, M. (Eds.), Proc. 6th ICFEM. In: LNCS, vol. 3308. Springer, pp. 373–390.

Armando, A., Bonacina, M.P., Ranise, S., Schulz, S., 2009. New results on rewrite-based satisfiability procedures. ACM Trans. Comput. Logic 10 (1), 129–179.

Armando, A., Castellini, C., Giunchiglia, E., Maratea, M., 2004. A SAT-based decision procedure for the Boolean combination of difference constraints. In: Proc. SAT-7.

Armando, A., Ranise, S., Rusinowitch, M., 2003. A rewriting approach to satisfiability procedures. Inform. and Comput. 183 (2), 140–164.

Barrett, C., Nieuwenhuis, R., Oliveras, A., Tinelli, C., 2006. Splitting on demand in SAT modulo theories. In: Hermann, M., Voronkov, A. (Eds.), Proc. LPAR-13. In: LNCS, vol. 4246. Springer, pp. 512–526.

Barrett, C.W., Dill, D.L., Stump, A., 2002. Checking satisfiability of first-order formulas by incremental translation to SAT. In: Larsen, K.G., Brinksma, E. (Eds.), Proc. CAV-14. In: LNCS, vol. 2404. Springer, pp. 236–249.

Bonacina, M.P., 1999. A taxonomy of theorem-proving strategies. In: Wooldridge, M.J., Veloso, M. (Eds.), Artificial Intelligence Today — Recent Trends and Developments. In: LNAI, vol. 1600. Springer, pp. 43–84.

Bonacina, M.P., Echenim, M., 2007a. Rewrite-based decision procedures. In: Archer, M., de la Tour, T.B., Munoz, C. (Eds.), Proc. 6th STRATEGIES Workshop, FLoC 2006. In: ENTCS, vol. 174(11). Elsevier, pp. 27–45.

Bonacina, M.P., Echenim, M., 2007b. Rewrite-based satisfiability procedures for recursive data structures. In: Cook, B., Sebastiani, R. (Eds.), Proc. 4th PDPAR Workshop, FLoC 2006. In: ENTCS, vol. 174 (8). Elsevier, pp. 55–70.

Bonacina, M.P., Echenim, M., 2007c. T-decision by decomposition. In: Pfenning, F. (Ed.), Proc. CADE-21. In: LNAI, vol. 4603. Springer, pp. 199–214.

Bonacina, M.P., Echenim, M., 2008. On variable-inactivity and polynomial T-satisfiability procedures. J. Logic Comput. 18 (1), 77–96.

Bonacina, M.P., Ghilardi, S., Nicolini, E., Ranise, S., Zucchelli, D., 2006. Decidability and undecidability results for Nelson–Oppen and rewrite-based decision procedures. In: Furbach, U., Shankar, N. (Eds.), Proc. IJCAR-3. In: LNAI, vol. 4130. Springer, pp. 513–527.

Bouillaguet, C., Kuncak, V., Wies, T., Zee, K., Rinard, M., 2007. Using first-order theorem provers in the Jahob data structure verification system. In: Proc. 8th VMCAI. In: LNCS, vol. 4349. Springer.

Bozzano, M., Bruttomesso, R., Cimatti, A., Junttila, T., Ranise, S., van Rossum, P., Sebastiani, R., 2006. Efficient theory combination via Boolean search. Inform. and Comput. 204 (10), 1493–1525.

Bozzano, M., Bruttomesso, R., Cimatti, A., Junttila, T., van Rossum, P., Schulz, S., Sebastiani, R., 2005. MathSAT: Tight integration of SAT and mathematical decision procedures. J. Automat. Reason. 35 (1–3), 265–293.

Bradley, A.R., 2007. Safety analysis of systems. Ph.D. Thesis, Stanford University.

Bradley, A.R., Manna, Z., 2007. The Calculus of Computation. Springer.

Bradley, A.R., Manna, Z., Sipma, H.B., 2006. What's decidable about arrays? In: Emerson, E.A., Namjoshi, K.S. (Eds.), Proc. VMCAI-7. In: LNCS, vol. 3855. Springer, pp. 427–442.

Brand, D., 1975. Proving theorems with the modification method. SIAM J. Comput. 4 (4), 412–430.

Bruttomesso, R., 2008. RTL verification: From SAT to SMT(BV). Ph.D. Thesis, Università degli Studi di Trento.

Bryant, R.E., Lahiri, S.K., Seshia, S.A., 2002. Modeling and verifying systems using a logic of counter arithmetic with lambda expressions and uninterpreted functions. In: Larsen, K.G., Brinksma, E. (Eds.), Proc. CAV-14. In: LNCS, vol. 2404. Springer, pp. 78–92.

Bryant, R.E., Velev, M.N., 2001. Processor verification using efficient reductions of the logic of uninterpreted functions to propositional logic. ACM TOCL 2 (1), 93–134.

Caferra, R., Leitsch, A., Peltier, N., 2004. Automated Model Building. Kluwer Academic Publishers.

Davis, M., Logemann, G., Loveland, D., 1962. A machine program for theorem-proving. C. ACM 5 (7), 394–397.

Davis, M., Putnam, H., 1960. A computing procedure for quantification theory. J. ACM 7, 201–215.

de Moura, L., Bjørner, N., 2007. Efficient E-matching for SMT-solvers. In: Pfenning, F. (Ed.), Proc. CADE-21. In: LNAI, vol. 4603. Springer, pp. 183–198.

de Moura, L., Bjørner, N., 2008a. Engineering DPLL(T) + saturation. In: Armando, A., Baumgartner, P., Dowek, G. (Eds.), Proc. IJCAR-4. In: LNAI, vol. 5195. Springer, pp. 475–490.

de Moura, L., Bjørner, N., 2008b. Model-based theory combination. In: Krstić, S., Oliveras, A. (Eds.), Proc. 5th SMT Workshop, CAV 2007. In: ENTCS, vol. 198 (2). Elsevier, pp. 37–49.

de Moura, L., Rueß, H., Sorea, M., 2002. Lazy theorem proving for bounded model checking over infinite domains. In: Voronkov, A. (Ed.), Proc. CADE-18. In: LNAI, vol. 2392. Springer, pp. 438–455.

Déharbe, D., Ranise, S., 2003. Light-weight theorem proving for debugging and verifying units of code. In: Proc. SEFM-1. IEEE Computer Society Press, pp. 220–228.

Dershowitz, N., Plaisted, D.A., 2001. Rewriting. In: Robinson, A., Voronkov, A. (Eds.), Handbook of Automated Reasoning, vol. 1. Elsevier, pp. 535–610.

Detlefs, D.L., Nelson, G., Saxe, J.B., 2005. Simplify: A theorem prover for program checking. J. ACM 52 (3), 365–473.

Dutertre, B., de Moura, L., 2006. A fast linear arithmetic solver for DPLL(T). In: Ball, T., Jones, R.B. (Eds.), Proc. CAV-18. In: LNCS, vol. 4144. Springer, pp. 81–94.

Ganzinger, H., Sofronie-Stokkermans, V., Waldmann, U., 2006. Modular proof systems for partial functions with Evans equality. Inform. and Comput. 240 (10), 1453–1492.

Ge, Y., Barrett, C., Tinelli, C., 2007. Solving quantified verification conditions using satisfiability modulo theories. In: Pfenning, F. (Ed.), Proc. CADE-21. In: LNAI, vol. 4603. Springer, pp. 167–182.

Ghilardi, S., Nicolini, E., Ranise, S., Zucchelli, D., 2007. Decision procedure for extensions of the theory of arrays. Ann. Math. Artif. Intell. 50 (3–4), 231–254.

Ghilardi, S., Nicolini, E., Zucchelli, D., 2005. A comprehensive framework for combined decision procedures. In: Gramlich, B. (Ed.), Proc. 5th FroCoS. In: LNAI, vol. 3717. Springer, pp. 1–30.

Henzinger, T.A., Jhala, R., Majumdar, R., Sutre, G., 2002. Lazy abstraction. In: Proc. POPL-29. ACM Press, pp. 58–70.
Jackson, D., Vaziri, M., 2000. Finding bugs with a constraint solver. In: Harrold, M.J. (Ed.), Proc. ISSTA 2000. ACM Press.
Jacobs, S., 2008. Incremental instance generation in local reasoning. In: Baader, F., Ghilardi, S., Hermann, M., Sattler, U., Sofronie-Stokkermans, V. (Eds.), Notes 1st CEDAR Workshop, IJCAR 2008. pp. 47–62.
Kirchner, H., Ranise, S., Ringeissen, C., Tran, D.-K., 2006. Automatic combinability of rewriting-based satisfiability procedures. In: Hermann, M., Voronkov, A. (Eds.), Proc. LPAR-13. In: LNCS, vol. 4246. Springer, pp. 542–556.
Krstić, S., Goel, A., 2007. Architecting solvers for SAT modulo theories: Nelson–Oppen with DPLL. In: Wolter, F. (Ed.), Proc. FroCoS-6. In: LNAI, vol. 4720. Springer, pp. 1–27.
Lahiri, S.K., Musuvathi, M., 2005. An efficient decision procedure for UTVPI constraints. In: Gramlich, B. (Ed.), Proc. FroCoS-5. In: LNAI, vol. 3717. Springer, pp. 168–183.
Lynch, C., Tran, D.-K., 2007. Automatic decidability and combinability revisited. In: Pfenning, F. (Ed.), Proc. CADE-21. In: LNAI, vol. 4603. Springer, pp. 328–344.
Lynch, C., Tran, D.-K., 2008. Smels: Satisfiability modulo equality with lazy superposition. In: Cha, S.D., Choi, J.-Y., Kim, M., Lee, I., Viswanathan, M. (Eds.), Proc. ATVA-6. In: LNCS, vol. 5311. Springer, pp. 186–200.
Meir, O., Strichman, O., 2005. Yet another decision procedure for equality logic. In: Etessami, K., Rajamani, S. (Eds.), Proc. CAV-17. In: LNCS, vol. 3576. Springer, pp. 307–320.
Nelson, G., Oppen, D.C., 1979. Simplification by cooperating decision procedures. ACM TOPLAS 1 (2), 245–257.
Nieuwenhuis, R., Oliveras, A., Tinelli, C., 2006. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). J. ACM 53 (6), 937–977.
Nieuwenhuis, R., Rubio, A., 2001. Paramodulation-based theorem proving. In: Robinson, A., Voronkov, A. (Eds.), Handbook of Automated Reasoning, vol. 1. Elsevier, pp. 371–443.
Plaisted, D.A., Zhu, Y., 1997. The Efficiency of Theorem Proving Strategies. Friedr. Vieweg & Sohns.
Ranise, S., Déharbe, D., 2003. Light-weight theorem proving for debugging and verifying pointer manipulating programs. In: Dahn, I., Vigneron, L. (Eds.), Proc. FTP-2003. No. DSCI–II710/03 in Technical Reports. Universidad Politécnica de Valencia, pp. 119–132.
Sebastiani, R., 2007. Lazy satisfiability modulo theories. J. Satisfiability, Boolean Model. Comput. 3, 141–224.
Seshia, S.A., Lahiri, S.K., Bryant, R.E., 2003. A hybrid SAT-based decision procedure for separation logic with uninterpreted functions. In: Fix, L., Lavagno, L. (Eds.), Proc. 40th DAC. pp. 425–430.
Sofronie-Stokkermans, V., 2005. Hierarchic reasoning in local theory extensions. In: Nieuwenhuis, R. (Ed.), Proc. CADE-20. In: LNAI, vol. 3632. Springer, pp. 219–234.
Stump, A., Barrett, C.W., Dill, D.L., Levitt, J., 2001. A decision procedure for an extensional theory of arrays. In: Halpern, J. (Ed.), Proc. LICS-16. IEEE Computer Society Press.
Waldmann, U., Prevosto, V., 2006. SPASS + T. In: Sutcliffe, G., Schmidt, R., Schulz, S. (Eds.), Proc. ESCoR Workshop, FLoc 2006. In: CEUR Workshop Proceedings, vol. 192. pp. 18–33.
Zhang, J., 2000. Specification analysis and test data generation by solving Boolean combinations of numeric constraints. In: Proc. APAQS-1. IEEE Computer Society Press, pp. 267–274.
Zhang, L., Malik, S., 2002. The quest for efficient boolean satisfiability solvers. In: Voronkov, A. (Ed.), Proc. CADE-18. In: LNCS, vol. 2392. Springer, pp. 295–313.