Information and Computation 209 (2011) 1026–1047

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Automatic decidability and combinability ☆

Christopher Lynch [a], Silvio Ranise [b,1], Christophe Ringeissen [c], Duc-Khanh Tran [d,*,1]

[a] Department of Mathematics and Computer Science, P.O. Box 5815 Clarkson University, Potsdam, NY 13699-5815, USA
[b] FBK-Irst, Via Sommarive 18, I-38100 Povo, Trento, Italy
[c] LORIA – INRIA Lorraine, 615, rue du Jardin Botanique, BP 101, 54602 Villers-les-Nancy Cedex, France
[d] School of Information and Communication Technology, Hanoi University of Science and Technology, 1 Dai Co Viet, Hanoi, Viet Nam

ARTICLE INFO

Article history:
Received 9 July 2007
Revised 28 February 2011
Available online 7 April 2011

Keywords:
Decision procedures
Paramodulation
Schematic Saturation
Combination

ABSTRACT

Verification problems can often be encoded as first-order validity or satisfiability problems. The availability of efficient automated theorem provers is a crucial pre-requisite for automating various verification tasks as well as their cooperation with specialized decision procedures for selected theories, such as Presburger Arithmetic. In this paper, we investigate how automated provers based on a form of equational reasoning, called paramodulation, can be used in verification tools. More precisely, given a theory T axiomatizing some data structure, we devise a procedure to answer the following questions. Is the satisfiability problem of T decidable by paramodulation? Can a procedure based on paramodulation for T be efficiently combined with other specialized procedures by using the Nelson–Oppen schema? Finally, if paramodulation decides the satisfiability problem of two theories, does it decide satisfiability in their union?

The procedure capable of answering all questions above is based on Schematic Saturation; an inference system capable of over-approximating the inferences of paramodulation when solving satisfiability problems in a given theory T. Clause schemas derived by Schematic Saturation describe all clauses derived by paramodulation so that the answers to the questions above are obtained by checking that only finitely many different clause schemas are derived or that certain clause schemas are not derived.

1. Introduction

An increasing number of verification tools, such as verification condition generators [9], software model-checkers [3], or static analyzers [6], require the use of Automated Theorem Provers (ATP) for First-Order Logic (FOL) to implement the back-ends for the automatic analyses of specifications and properties. This is so because verification problems can often be encoded as validity (or, dually, satisfiability) problems and the availability of efficient ATPs becomes a crucial pre-requisite for automating the various verification tasks.

Despite the great progress in the last twenty years in automated theorem proving in FOL, general-purpose ATPs cannot be used off-the-shelf to work with the sort of formulae generated by verification tools. The main reason is that these tools are not interested in validity in general but in validity with respect to some background theory, that fixes the interpretations of certain predicates and function symbols. For instance, in verification problems involving the integers, one is not interested

☆ Preliminary versions of the results in this paper appear in Lynch and Morawska (2002) [18], Kirchner et al. (2006) [16], Lynch and Tran (2007) [19].

* Corresponding author.
E-mail addresses: clynch@clarkson.edu (C. Lynch), ranise@fbk.eu (S. Ranise), Christophe.Ringeissen@loria.fr (C. Ringeissen), khanhtd@soict.hut.edu.vn (D.-K. Tran).

[1] Work done while the author was at LORIA – INRIA Lorraine.

0890-5401/$ – see front matter
doi:10.1016/j.ic.2011.03.005

in showing that the formula $\forall x, y.(x < y \Rightarrow x < y + y)$ is true for all possible interpretations of the non-logical symbols $<$ and $+$, but only for those interpretations in which $<$ is the usual ordering over the integers and $+$ is the addition function. When proving the validity of a formula, general-purpose ATPs have only one way to consider the interpretations allowed by a background theory T: add as a premise to the formula a conjunction of the axioms for T. There are several important theories of data structures admitting finite axiomatizations; e.g., lists, arrays, records, and integer-offsets. Unfortunately, there are also ubiquitous theories in verification such as Presburger Arithmetic which can only be approximated by finite sets of axioms (see, e.g., [6]). However, even when finite axiomatizations exist for the background theory T, the performance of an ATP is usually poor for realistic verification applications when it is used off-the-shelf (see [28] for an extensive discussion on this issue and possible solutions).

There exist specialized reasoning methods for many background theories of interest, such as the ones listed above, which go under the name of Satisfiability Modulo Theories (SMT) solvers, but they are limited to the particular class of FOL formulae without quantifiers. While being able to cope only with quantifier-free formulae is not an obstacle for some verification applications, it may become a serious limitation in the verification of complex data structures. Finding good heuristics for lifting SMT techniques from ground to quantified formulas is a hot line of current (see, e.g., [12]) and future research. On the other hand, ATPs are—at least in principle—capable of handling arbitrary FOL formulae, including those containing quantifiers.

Finally, to make the situation even more complex, most verification problems involve more than one theory, so that methods to combine theories, such as the one pioneered by Nelson and Oppen [22], are required to modularly re-use procedures for the component theories.

Given the large variety of FOL formulae generated by verification tools, especially in the context of software verification where formulae containing quantifiers are quite frequently obtained, it is desirable to make ATPs and specialized decision procedures cooperate so as to augment the degree of automation of verification techniques. In this paper, we consider the problem of embedding ATPs in verification tools and devise methods for their cooperation with other specialized decision procedures. In this respect, it is crucial to develop a *general framework* for

1. establishing the termination of ATPs on selected classes of FOL formulae,
2. guaranteeing the modular termination of an ATP, i.e. a procedure to check the termination of an ATP on the union of two theories when the ATP terminates on each component theory, and
3. providing an efficient way to combine ATPs with *ad hoc* decision procedures so that theories not admitting finite axiomatizations, like Presburger Arithmetic, can be precisely handled.

Automatic decidability. The rewriting approach in [2] proposes a methodology to build satisfiability procedures which consists in showing the termination of a fair theorem proving strategy of a refutation complete calculus (namely, paramodulation [23]) on a set of clauses obtained as the union of the axioms $Ax(T)$ of the background theory T and a finite set S of ground literals in T. A drawback of [2] is that the proof of termination must be repeated for each theory T.

The first contribution of this paper (along the lines of [18]) is a procedure for checking the termination of any fair theorem proving strategy of the paramodulation calculus (denoted with $\mathcal{PC}$). The procedure is based on *Schematic Saturation*, an inference system working on constrained clauses, denoted with $\mathcal{SPC}$. The key insight is that constrained clauses derived by $\mathcal{SPC}$ schematize the clauses that can be obtained by a fair theorem proving strategy of $\mathcal{PC}$ on the axioms $Ax(T)$ of the theory T and a set of constrained clauses, schematizing any finite set of ground literals of a particular form, called "flat". Our main result is that if $\mathcal{SPC}$ halts on the union of $Ax(T)$ and the schematic representation of an arbitrary set of ground flat literals, $\mathcal{PC}$ also halts on the union of $Ax(T)$ and an arbitrary set of ground flat literals. By the refutation completeness of $\mathcal{PC}$, we are entitled to conclude that the satisfiability problem of T is decidable. To illustrate our approach, we show that $\mathcal{SPC}$ halts for the theories in [2] as well as others, such as the theory of selection functions (an approximation of the theory of recursively defined data structures considered by Oppen in [24]).

Moreover, when $\mathcal{SPC}$ halts, it does not only show the decidability of the satisfiability problem for T, but it also gives an upper bound on the number of clauses that will be derived in the limit (also called *persistent clauses*), while applying the fair theorem proving strategy of $\mathcal{PC}$. In general, the number of persistent clauses is exponential in the number of symbols in the input set of clauses. If the set $Ax(T)$ of axioms of the theory T contains only literals, then the bound on the number of persistent clauses is simply polynomial. The time complexity of a paramodulation-based satisfiability procedure can be obtained by refining our bounds on the number of persistent clauses along the lines of [18,10]. We do not do this here to maintain the paper to a reasonable size and since we interested to investigate other problems to satisfy desiderata (2) and (3) above.

Our procedure for checking the decidability of a theory T by paramodulation is a first step to fulfill desideratum (1) above.

Automatic combinability. As observed in [1], if two theories T_1 and T_2 are axiomatized by two finite sets of clauses, then it is possible to use theorem-proving strategies to decide the satisfiability problem in their union. In this setting, combination reduces to modularity of termination, i.e. showing that if a fair theorem proving strategy of $\mathcal{PC}$ decides the satisfiability problem of T_1 and T_2 separately, then such a strategy decides also the satisfiability problem of their union. Under the assumption that the theorem proving strategy of $\mathcal{PC}$ halts on both T_1 and T_2, the problem which may prevent the ter-

mination of the strategy for their union is to have "across-theories inferences," between one clause in T_1 and one in T_2, which may contribute to generate newer and newer persistent clauses. The second contribution of this paper is a method for checking a sufficient condition on the component theories to guarantee the absence of across-theories inferences. This is done in three steps. First, we identify "bad" clauses that may cause across-theories inferences. Second, by restricting component theories (similar to the one in [1]) so that $\mathcal{PC}$ does not derive "bad" clauses, we obtain the termination of $\mathcal{PC}$ for unions of theories. Finally, we give the procedure to check that a single theory satisfies this restriction; by using again $\mathcal{SPC}$ and detecting if constrained clauses schematizing "bad" clauses are derived.

Our procedure to establish the condition preventing "across-theories inferences" by paramodulation fulfills desideratum (2) above.

Although useful, checking for modular termination of paramodulation is not enough as some theories, notably Presburger Arithmetic, do not admit satisfiability procedures based on paramodulation. Fortunately, theories of this kind come with specialized decision procedures for their satisfiability problem, that can be combined with others via the Nelson–Oppen combination schema [22]. The key requirement for the correctness of this schema is that each component theory T is *stably infinite*, i.e. if the set S of ground literals is satisfiable in T, then $T \cup S$ admits a model whose domain is infinite.

The third contribution of this paper is the design of a procedure to establish that a finitely presented theory T is stably infinite. If $\mathcal{SPC}$ halts on the union of $Ax(T)$ and the schematic representation of an arbitrary set of ground flat literals and it does not derive the trivial equality $X = Y$, then the theory T is stably infinite. In this way, $\mathcal{SPC}$ can recognize both decidability and stable infiniteness of a finitely presented theory T and the resulting decision procedure for the satisfiability problem of T can cooperate with others by the Nelson–Oppen schema. Indeed, this is only a first step towards desideratum (3) above. In fact, one of the key problems to efficiently combine procedures *à la* Nelson–Oppen is to derive selected facts which must be exchanged among procedures for their synchronization (see [22] for details). Theoretically, the problem of computing an entailed fact has a simple solution: to derive φ from Γ, it is sufficient to guess φ and then check the satisfiability of its negation in conjunction with Γ. In practice, guessing decreases performances unacceptably (see [8] for an in-depth discussion of this issue), so that we require the procedure to be *deduction complete*, i.e. capable of deriving the facts needed for synchronization. In [15], it is proved that—under certain assumptions—a fair theorem proving strategy of $\mathcal{PC}$ derives enough facts to guarantee the completeness of the Nelson–Oppen schema. This result is not obvious since $\mathcal{PC}$ is not complete for consequence finding.

The fourth contribution of this paper is a method for checking deduction completeness of paramodulation-based procedures. We show how $\mathcal{SPC}$ can check that a Horn theory is deduction complete. This result generalizes [15] where proofs of deduction completeness are repeatedly developed for some selected theories. We also discuss how to obtain deduction completeness for non-Horn theories.

The checks to establish that a finitely presented theory is stably infinite and deduction complete allows us to fulfill desideratum (3) above.

Plan of the paper. The paper is structured as follows. Section 2 introduces some background notions and briefly overviews the main ideas underlying the paramodulation calculus and the Nelson–Oppen combination schema. Section 3 presents Schematic Saturation and its application to check the decidability of the satisfiability problem for finitely presented theories. To ease the understanding we first illustrate Schematic Saturation with equational theories, and then generalize it to non-equational theories. Section 4 gives another application of Schematic Saturation to check the modular termination of fair theorem proving strategies for unions of theories, as well as stable infiniteness of finitely presented theories and deduction completeness of their paramodulation-based decision procedures. Section 5 discusses the relevance of the results and compares them with related work. Finally, Section 6 concludes and draws some perspectives for future work.

2. Background

2.1. First-order logic

We assume the usual first-order syntactic notions of signature, term, position, and substitution, as defined, e.g., in [7].

If l and r are two terms, then $l = r$ is an *equality* and $\neg(l = r)$ (also written as $l \neq r$) is a *disequality*. A *literal* is either an equality or a disequality. A *positive literal* is an equality and a *negative literal* is a disequality. A first-order *formula* is built in the usual way over the universal and existential quantifiers, Boolean connectives, and symbols in a given first-order signature. We call a formula *ground* if it has no variables. A *clause* is a disjunction of literals. A clause $\neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_n$ is sometimes written in sequent style as $\{A_1, \ldots, A_n\} \Rightarrow \{B_1, \ldots, B_m\}$, where the A_i's and B_j's are equalities. A *unit* clause is a clause with only one disjunct, equivalently a literal. The *empty* clause, denoted $\perp$, is the clause with no disjunct, and it is equivalent to an unsatisfiable formula.

Definition 1 *(Elementary clause).* An elementary clause is a clause of the form $x_1 = y_1 \vee \cdots \vee x_n = y_n$, where x_i, y_i are distinct constants or variables for $i = 1, \ldots, n$ and $n \geqslant 1$.

We define a function *depth* such that for a term t, $depth(t) = 0$, if t is a constant or a variable, and $depth(f(t_1, \ldots, t_n)) = 1 + max\{depth(t_i) \mid 1 \leqslant i \leqslant n\}$. A term is *flat* if its depth is 0 or 1. For a literal, $depth(l \bowtie r) = depth(l) + depth(r)$, where $\bowtie \in \{=, \neq\}$. A positive literal is *flat* if its depth is 0 or 1. A negative literal is *flat* if its depth is 0. We use the following

Right Paramodulation

$$\frac{\Gamma \Rightarrow \Delta, l[u'] = r \quad \Pi \Rightarrow \Sigma, u = t}{\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)}$$

if $\sigma(u) \not\preccurlyeq \sigma(t)$, $u = t$ is selected in its clause, $\sigma(l[u']) \not\preccurlyeq \sigma(r)$, and $l = r$ is selected in its clause.

Left Paramodulation

$$\frac{\Gamma, l[u'] = r \Rightarrow \Delta \quad \Pi \Rightarrow \Sigma, u = t}{\sigma(l[t] = r, \Gamma, \Pi \Rightarrow \Delta, \Sigma)}$$

if $\sigma(u) \not\preccurlyeq \sigma(t)$, $u = t$ is selected in its clause, $\sigma(l[u']) \not\preccurlyeq \sigma(r)$, and $l = r$ is selected in its clause.

Reflection

$$\frac{\Gamma, u' = u \Rightarrow \Delta}{\sigma(\Gamma \Rightarrow \Delta)}$$

if $u' = u$ is selected in its clause.

Eq. Factoring

$$\frac{\Gamma \Rightarrow \Delta, u = t, u' = t'}{\sigma(\Gamma, t = t' \Rightarrow \Delta, u = t')}$$

if $\sigma(u) \not\preccurlyeq \sigma(t)$, $u = t$ is selected in its clause, $\sigma(t) \not\preccurlyeq \sigma(t')$ and $\sigma(u') \not\preccurlyeq \sigma(t')$.

Above, σ is the most general unifier of u and u'. In the rules *Left paramodulation* and *Right paramodulation*, u' is not a variable.

Fig. 1. Expansion inference rules $\mathcal{PC}$.

notations: $\equiv$ is identity, l, r, u, t are terms, v, w, x, y, z are variables, all other lower case letters are constant or function symbols.

We also assume the usual first-order notions of model, satisfiability, validity, logical consequence.

A *first-order theory* (over a finite signature) is a set of first-order formulae with no free variables. When T is a finitely axiomatized theory, $Ax(T)$ denotes the set of axioms of T. All the theories in this paper are first-order theories *with equality*, which means that the equality symbol $=$ is always interpreted as the equality relation. A formula is *satisfiable in a theory T* if it is satisfiable in a model of T. The *satisfiability problem* for a theory T amounts to establishing whether any given finite conjunction of literals (or equivalently, any given finite set of literals) is T-satisfiable or not. A *satisfiability procedure* for T is any algorithm that solves the satisfiability problem for T (the satisfiability of any quantifier-free formula can be reduced to the satisfiability of sets of literals by converting to disjunctive normal form and then splitting on disjunctions).

2.2. A paramodulation calculus

The calculus $\mathcal{PC}$ consists of the rules in Figs. 1 and 2. A fundamental feature of $\mathcal{PC}$ is the usage of a *reduction ordering* $\succ$ which is total on ground terms, for example the lexicographic path ordering [7]. The ordering $\succ$ is extended to positive literals by considering them as multisets of terms, and then to the clauses by considering them as multisets of positive literals. $\mathcal{PC}$ uses a selection function *sel* such that for each clause C, $sel(C)$ contains a negative literal in C or all maximal literals in C wrt. $\succ$.

A clause C is *redundant* with respect to a set S of clauses if either $C \in S$ or S can be obtained from $S \cup \{C\}$ by a sequence of application of the contraction rules of Fig. 2. An inference is *redundant* with respect to a set S of clauses if its conclusion is redundant with respect to S. A set S of clauses is *saturated* with respect to $\mathcal{PC}$ if every inference of $\mathcal{PC}$ with a premise in S is redundant with respect to S. A *derivation* is a sequence $S_0, S_1, \ldots, S_i, \ldots$ of sets of clauses where at

Subsumption

$$\frac{S \cup \{C, C'\}}{S \cup \{C\}}$$

if for some substitution θ, $\theta(C) \subseteq C'$.

Simplification

$$\frac{S \cup \{C[l'], l = r\}}{S \cup \{C[\theta(r)], l = r\}}$$

if $l' \equiv \theta(l)$, $\theta(l) \succ \theta(r)$, and $C[l'] \succ (\theta(l) = \theta(r))$.

Tautology Deletion

$$\frac{S \cup \{\Gamma \Rightarrow \Delta, t = t\}}{S}$$

Above, C and C' are clauses and S is a set of clauses.

Fig. 2. Contraction inference rules $\mathcal{PC}$.

each step an inference of $\mathcal{PC}$ is applied to generate and add a clause (cf. expansion rules in Fig. 1) or to delete or reduce a clause (cf. contraction rules in Fig. 2). A derivation is characterized by its *limit*, defined as the set of persistent clauses $S_\infty = \bigcup_{j \geqslant 0} \bigcap_{i > j} S_i$. A derivation $S_0, S_1, \ldots, S_i, \ldots$ with limit S_∞ is *fair* with respect to $\mathcal{PC}$ if for every inference in $\mathcal{PC}$ with premises in S_∞, there is some $j \geqslant 0$ such that the inference is redundant in S_j.

Theorem 1. *(See [23].) If $S_0, S_1, \ldots$ is a fair derivation of $\mathcal{PC}$, then* (i) *its limit S_∞ is saturated with respect to $\mathcal{PC}$,* (ii) *S_0 is unsatisfiable iff the empty clause is in S_j for some j, and* (iii) *if such a fair derivation is finite, i.e. it is of the form $S_0, \ldots, S_n$, then S_n is saturated and logically equivalent to S_0.*

2.3. Paramodulation-based satisfiability procedure

We assume the following:

Assumption 1. If a term t is not a variable or a constant, then for any constant c we have that $t \succ c$.

The paramodulation-based methodology [2] to build satisfiability procedures consists of two phases.

1. *Flattening*: all ground literals are flattened by introducing new constants, yielding an equisatisfiable set of ground *flat* literals.
2. *Ordering selection and termination*: any fair derivation of $\mathcal{PC}$ is shown to be finite when applied to an arbitrary set of ground flat literals together with the axioms of T, provided that $\succ$ satisfies Assumption 1.

If T is a theory for which the paramodulation-based methodology applies, a T-satisfiability procedure can be built by implementing the flattening (this can be done once and for all), and by using a prover mechanizing $\mathcal{PC}$ with a suitable ordering $\succ$. If the final set of clauses returned by the prover contains the empty clause, then the T-satisfiability procedure returns unsatisfiable; otherwise, it returns satisfiable. A satisfiability procedure built using this approach is said paramodulation-based.

2.4. Nelson–Oppen combination method

The Nelson–Oppen combination method [22] allows us to solve the problem of checking the satisfiability of a conjunction Φ of ground literals in the union of two signature-disjoint theories T_1 and T_2 such that a T_i-satisfiability procedure is available, for $i = 1, 2$. Since the literals in Φ may be built over symbols in T_1 or in T_2, we need to *purify* them by introducing fresh constants to abstract subterms. This process leaves us with a conjunction $\Phi_1 \wedge \Phi_2$ which is equisatisfiable to Φ where

Φ_i contains only literals built over the signature of T_i, for $i = 1, 2$. In this way, literals in Φ_i can be *dispatched* to the available decision procedure for T_i. The next step of the method consists in exchanging ground elementary clauses (or equivalently disjunctions of equalities between constants) between the two procedures until either unsatisfiability is derived by one of the component decision procedures, or no more ground elementary clauses can be exchanged. In the first case, we derive the unsatisfiability of the input formula; in the second case, we derive its satisfiability. To show the correctness of the Nelson–Oppen method [21,30,25], the theories T_1 and T_2 must be *stably infinite*.

Definition 2 *(Stably infinite theory).* Let T be a consistent theory. T is *stably infinite* if every T-satisfiable conjunction φ of ground literals is T-satisfiable in an infinite model.

When combining convex theories, the Nelson–Oppen combination method works without affecting completeness by exchanging only ground elementary equalities.

Definition 3 *(Convex theory).* A theory is *convex* if for any conjunction Γ of equalities, a disjunction D of equalities is entailed by Γ if and only if some disjunct of D is entailed by Γ.

Examples of convex theories are the theory of equality, the theory of lists, and Linear Arithmetic over the Rationals.

To be efficiently combined *à la* Nelson–Oppen, the component satisfiability procedures must be capable of deriving sufficiently many ground elementary clauses which are implied by the input set of literals. Such satisfiability procedures are said *deduction complete*.

Definition 4 *(Deduction complete satisfiability procedure).* A T-satisfiability procedure is *deduction complete* if for any T-satisfiable conjunction ϕ of ground literals it returns, in addition to satisfiable, a set S_e of ground elementary clauses such that for every ground elementary clause C, the following holds: $T \models \phi \Rightarrow C$ iff $S_e \models C$.

3. Automatic decidability

Schematic Saturation works by saturating the axioms $Ax(T)$ of a theory T along with the set G_0^T schematizing any finite set of ground flat literals built out of the symbols in the signature Σ_T of T, with respect to the inference system $\mathcal{SPC}$ (see Figs. 3 and 4). If $\mathcal{SPC}$ halts on $Ax(T) \cup G_0^T$, then any saturation of $Ax(T) \cup S$ by $\mathcal{PC}$ is finite, for every set S of ground flat literals built over Σ_T. Consequently the T-satisfiability problem is decidable. Before being able to present Schematic Saturation, we need to introduce a couple of preliminary notions.

Definition 5 *(Constraint).* An *atomic constraint* is of the form $t \preccurlyeq t'$ or $t \not\preccurlyeq t'$. A *constraint* is a conjunction of atomic constraints.

A substitution λ satisfies a constraint ϕ if $\lambda(\phi)$ is true. A constraint ϕ is satisfiable if there exists a substitution λ satisfying ϕ. In the sequel, by $c^\top$, we mean the biggest constant wrt. $\succ$. For example, a constraint of the form $t \preccurlyeq c^\top$ is true if t is a constant, it is false if t is a term of depth at least 1 (i.e. containing a function symbol) and it is satisfiable if t is a variable.

Definition 6 *(Constrained clause).* A *constrained clause* is of the form $C \parallel \phi$, where C is a clause and ϕ is a constraint.

We say that $\lambda(C)$ is an *instance* of $C \parallel \phi$ if λ is a substitution satisfying ϕ.

Definition 7 *(Constrained variable).* A variable x is *constrained* in a constrained clause $C \parallel \phi$ if $x \preccurlyeq c^\top$ is in ϕ; otherwise it is *unconstrained*.

In fact, a constrained variable is a schematization of constants. A unconstrained variable is a universal variable, which is different from a constrained variable. A constrained variable could only be instantiated with a constant, whereas a unconstrained variable could be instantiated with any term.

Definition 8 *(Constraint instance).* We say that $\lambda(C)$ is a *constraint instance* of $C \parallel \phi$ if the domain of λ contains all the constrained variables in $C \parallel \phi$, the range of λ contains only constants, and $\lambda(\phi)$ is satisfiable.

For example, the clause $f(a) = X$ is a constraint instance of the constrained clause $f(x) = X \parallel x \preccurlyeq c^\top$, where a is a constant, x is a constrained variable and X is an unconstrained variable. It is important to underline that we can use a constrained clause to schematize the set of all its constraint instances.

Right Paramodulation

$$\frac{\Gamma \Rightarrow \Delta, l[u'] = r \parallel \phi \quad \Pi \Rightarrow \Sigma, u = t \parallel \varphi}{\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r \parallel \phi \wedge \varphi)}$$

if $\sigma(u) \not\preccurlyeq \sigma(t)$, $u = t$ is selected in its clause, $\sigma(l[u']) \not\preccurlyeq \sigma(r)$, and $l = r$ is selected in its clause.

Left Paramodulation

$$\frac{\Gamma, l[u'] = r \Rightarrow \Delta \parallel \phi \quad \Pi \Rightarrow \Sigma, u = t \parallel \varphi}{\sigma(l[t] = r, \Gamma, \Pi \Rightarrow \Delta, \Sigma \parallel \phi \wedge \varphi)}$$

if $\sigma(u) \not\preccurlyeq \sigma(t)$, $u = t$ is selected in its clause, $\sigma(l[u']) \not\preccurlyeq \sigma(r)$, and $l = r$ is selected in its clause.

Reflection

$$\frac{\Gamma, u' = u \Rightarrow \Delta \parallel \phi}{\sigma(\Gamma \Rightarrow \Delta \parallel \phi)}$$

if $u' = u$ is selected in its clause.

Eq. Factoring

$$\frac{\Gamma \Rightarrow \Delta, u = t, u' = t' \parallel \phi}{\sigma(\Gamma, t = t' \Rightarrow \Delta, u = t' \parallel \phi)}$$

if $\sigma(u) \not\preccurlyeq \sigma(t)$, $u = t$ is selected in its clause, $\sigma(t) \not\preccurlyeq \sigma(t')$ and $\sigma(u') \not\preccurlyeq \sigma(t')$.

Above, σ is the most general unifier of u and u'. In the rules *Left Paramodulation* and *Right Paramodulation*, u' is not a unconstrained variable.

Fig. 3. Expansion inference rules $\mathcal{SPC}$.

Definition 9 *(Constrained variant).* Let $C \parallel \phi$ and $C' \parallel \phi'$ be two constrained clauses. We say that $C \parallel \phi$ is a *constrained variant* of $C' \parallel \phi'$ if there exists a renaming λ from the set of all constrained variables of $C \parallel \phi$ to the one of $C' \parallel \phi'$ such that $\lambda(C') = C$ and $\lambda(\phi') \subseteq \phi$.

For instance, the clause $f(X) = x \vee x = g(Y) \parallel x \preccurlyeq c^{\top} \wedge g(Y) \not\preccurlyeq x \wedge f(X) \not\preccurlyeq x$ is a constrained variant $f(X) = y \vee y = g(Y) \parallel y \preccurlyeq c^{\top}$ as the renaming $\lambda = \{x \to y\}$ satisfies all the conditions in Definition 9.

For a given theory T with signature Σ_T, we define G_0^T as follows:

$$\begin{aligned} G_0^T = {} & \{\bot\} \\ & \cup \left\{x = y \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top}\right\} \\ & \cup \left\{x \neq y \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top}\right\} \\ & \cup \bigcup_{f \in \Sigma_T} \left\{ f(x_1, \ldots, x_n) = x_0 \parallel \bigwedge_{i=0}^{n} x_i \preccurlyeq c^{\top} \right\} \end{aligned}$$

Notice that G_0^T schematizes any set of ground flat equalities and disequalities built over Σ_T, along with the empty clause.

We assume the following:

Subsumption

$$\frac{S \cup \{C \parallel \phi, C' \parallel \phi'\}}{S \cup \{C \parallel \phi\}}$$

if (a) $C \in Ax(T)$, ϕ is empty and for some substitution θ, $\theta(C) \subseteq C'$; or (b) $C \parallel \phi$ and $C' \parallel \phi'$ are renamings of each other.

Simplification

$$\frac{S \cup \{C[l'] \parallel \phi, l = r\}}{S \cup \{C[\theta(r)] \parallel \phi, l = r\}}$$

if $l = r \in Ax(T)$, $l' \equiv \theta(l)$, $\theta(l) \succ \theta(r)$, and $C[l'] \succ (\theta(l) = \theta(r))$.

Tautology Deletion

$$\frac{S \cup \{\Gamma \Rightarrow \Delta, t = t \parallel \phi\}}{S}$$

Deletion

$$\frac{S \cup \{\Gamma \Rightarrow \Delta \parallel \phi\}}{S}$$

if ϕ is unsatisfiable.

Above, $C \parallel \phi$ and $C' \parallel \phi'$ are constrained clauses and S is a set of constrained clauses.

Fig. 4. Contraction inference rules $\mathcal{SPC}$.

Assumption 2. The ordering $\succ$ used in $\mathcal{PC}$ is extended to constrained clauses in such a way that a constrained clause $C \parallel \phi$ is smaller than a constrained clause $C' \parallel \phi'$ if all constraint instances of $C \parallel \phi$ are smaller wrt. $\succ$ than all constraint instances of $C' \parallel \phi'$.

The inference system $\mathcal{SPC}$ (Figs. 3 and 4) is almost identical to $\mathcal{PC}$. The main difference is that all clauses now have constraints; unconstrained clauses are considered to have empty constraints. Constraints are inherited by the conclusions of an inference. In the rules *Left Paramodulation* and *Right Paramodulation* of $\mathcal{SPC}$ (Fig. 3) the condition u' is not a unconstrained variable means that these inferences are allowed to perform into constrained variables. This is so because a constrained variable is a schematization of constants therefore if we exclude *Left Paramodulation* and *Right Paramodulation* into constrained variables in $\mathcal{SPC}$ then we are no longer able to simulate all inferences *Left Paramodulation* and *Right Paramodulation* into constants in $\mathcal{PC}$. Constrained Contraction Inference Rules (of Fig. 4) have different applicability conditions since we cannot simulate every subsumption, or simplification as we cannot assume that ground literals are always present in a saturation of $Ax(T) \cup S$, on which such contraction inferences depend. In other words, subsuming and simplifying clauses must be present in the saturation $Ax(T) \cup S$ for every set S of ground flat literals built over Σ_T, and only clauses in $Ax(T)$ would satisfy this property.

3.1. Schematic Saturation for equational theories

By Schematic Saturation for a theory T, we mean a saturation of $Ax(T) \cup G_0^T$ with respect to the inference system $\mathcal{SPC}$. The key idea underlying Schematic Saturation is the following. We would like to show that there are finitely many clauses generated in a saturation. Since there are finitely many constants in the input, this boils down to prove, by induction on the length of the saturation, that there are finitely many forms of clauses generated. Here we use G_0^T to schematize sets of ground flat literals and show, by induction on the length of the derivation by $\mathcal{PC}$, that for each clause C generated by an inference of $\mathcal{PC}$, there is an inference of $\mathcal{SPC}$ generating a constrained clause schematizing C. In this way, Schematic

Saturation is used to over-approximate any saturation of $\mathcal{PC}$, and therefore if Schematic Saturation halts then any saturation must halt. Let us illustrate this idea by considering an example.

Example 1. In [2], it is shown that the saturation by $\mathcal{PC}$ of any set of ground flat literals and the axioms of the theory of lists is finite. We would like to prove this result by using Schematic Saturation. The theory $\mathcal{L}$ of lists is axiomatized by the following saturated set $Ax(\mathcal{L})$ of axioms:

$$car\big(cons(X, Y)\big) = X \tag{L1}$$
$$cdr\big(cons(X, Y)\big) = Y \tag{L2}$$
$$cons\big(car(X), cdr(X)\big) = X \tag{L3}$$

where X and Y are implicitly universally quantified variables. The set $G_0^{\mathcal{L}}$ consists of the following clauses:

$$x = y \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \tag{L4}$$
$$x \neq y \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \tag{L5}$$
$$car(x) = y \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \tag{L6}$$
$$cdr(x) = y \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \tag{L7}$$
$$cons(x, y) = z \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \wedge z \preccurlyeq c^{\top} \tag{L8}$$

A *Right Paramodulation* of $\mathcal{SPC}$ between (L1) and (L8) yields

$$car(x) = y \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top}$$

which is immediately deleted because it is a constrained variant of the third member of $G_0^{\mathcal{L}}$. We have a similar case for (L2) and (L8).

A *Right Paramodulation* of $\mathcal{SPC}$ between (L3) and (L6) yields

$$cons\big(x, cdr(y)\big) = z \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \wedge z \preccurlyeq c^{\top}$$

Similarly, a *Right Paramodulation* of $\mathcal{SPC}$ between (L3) and (L7) gives

$$cons\big(car(x), y\big) = z \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \wedge z \preccurlyeq c^{\top}$$

Finally, Schematic Saturation contains, apart from all the clauses in $Ax(\mathcal{L}) \cup G_0^{\mathcal{L}}$, the following clauses:

$$cons\big(car(x), y\big) = z \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \wedge z \preccurlyeq c^{\top} \tag{L9}$$
$$cons\big(x, cdr(y)\big) = z \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \wedge z \preccurlyeq c^{\top} \tag{L10}$$

It is easy to see that given a finite set of constants, there are only a finite number of possible instantiations of all the constrained variables in Schematic Saturation for $\mathcal{L}$. We conclude that the saturation of any set of ground flat literals and the axioms of the theory $\mathcal{L}$ of lists is finite. □

We now show that Schematic Saturation provides us with a method of checking the decidability of satisfiability problems modulo an arbitrary equational theory.

Theorem 2. *Let T be a theory axiomatized by a finite set $Ax(T)$ of equalities, which is saturated with respect to $\mathcal{PC}$. Let G_∞^T be the set of all clauses in a saturation of $Ax(T) \cup G_0^T$ by $\mathcal{SPC}$. Then for every set S of ground flat Σ_T-literals, every clause in a saturation of $Ax(T) \cup S$ by $\mathcal{PC}$ is a constraint instance of some clause in G_∞^T.*

Proof. The proof is by induction on the length of derivations of $\mathcal{PC}$. The base case is obvious. For the inductive case, we need to show two facts:

1. each clause added in the process of saturation of $Ax(T) \cup S$ is a constraint instance of some clause in the saturation of $Ax(T) \cup G_0^T$ by $\mathcal{SPC}$, and
2. if a clause is deleted by *Subsumption Tautology Deletion* or *Deletion* from (or simplified by *Simplification* in) the saturation of $Ax(T) \cup G_0^T$ by $\mathcal{SPC}$, then all constraint instances of the latter will also be deleted from (or simplified in) the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$.

Proof of (1). Consider *Right Paramodulation* of $\mathcal{PC}$. By induction hypothesis $l[u'] = r$ and $u = t$ are constraint instances of some clause in G^T_∞, i.e. there is some clause D in G^T_∞ and a substitution θ such that $\theta(D) \equiv l[u'] = r$, and some clause D' in G^T_∞ such that $\theta(D') \equiv u = t$. But then there must exist a *Right Paramodulation* inference of $\mathcal{SPC}$ in the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$, whose premises are D and D' with conclusion D'' such that we can extend θ so that $\theta(D'') \equiv \sigma(l[t] = r)$. Hence $\sigma(l[t] = r)$ is a constraint instance of some clause in the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$.

The rules *Left Paramodulation*, *Reflection* of $\mathcal{PC}$ can be handled in a similar way to *Right Paramodulation* above and therefore omitted. *Eq. Factoring* of $\mathcal{PC}$ does not play any role as every saturation contains only unit clauses.

Proof of (2). Let us consider *Subsumption* of $\mathcal{SPC}$. The case (b) of *Subsumption* is just a matter of deleting duplicates. For case (a), assume that there are a clause A deleted from the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$ and a clause B in the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$, which is a constraint instance of A. Then there must exist a clause $C \in Ax(T)$ and some substitution θ such that $\theta(C) \subseteq A$. Since all the clauses in $Ax(T)$ persist, there must be a substitution θ' such that $\theta'(C) \subseteq B$. Thereby B must also be deleted from the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$.

A similar argument can be given for *Simplification* of $\mathcal{SPC}$. Assume that there are a clause $C[l'] \parallel \phi$ in the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$ simplified by an equality $l = r$ ($l = r \in Ax(T)$) into $C[\theta(r)] \parallel \phi$. Let σ be a substitution such that $\sigma(C[l'])$ is a constraint instance of $C[l'] \parallel \phi$. Since $l = r$ persists in the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$, there must be a simplification of $\sigma(C[l']) = \sigma(C)[\sigma(\theta(l))]$ by $l = r$ into $\sigma(C)[\sigma(\theta(r))] = \sigma(C[\theta(r)])$, which is a constraint instance of $C[\theta(r)] \parallel \phi$.

For the *Tautology Deletion* rule of $\mathcal{SPC}$, it is easy to see that a constraint instance of a tautology is also a tautology.

For the *Deletion* rule of $\mathcal{SPC}$, notice that clauses with an unsatisfiable constraint have no constraint instances. □

Using Schematic Saturation, we can also determine an upper bound on the number of clauses generated in a saturation by $\mathcal{PC}$ by simply counting the number of possible ground instantiations of constrained variables, given a finite set of constants. It is not difficult to see that the number of possible instantiations polynomially depends on the number of constants in the input set of ground flat literals.

Theorem 3. *Let T be a theory axiomatized by a finite set $Ax(T)$ of equalities, which is saturated with respect to $\mathcal{PC}$. Let G^T_∞ be the set of all clauses in a finite saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$. Then for every set S of ground flat Σ_T-literals, the number of clauses in a saturation $Ax(T) \cup S$ by $\mathcal{PC}$ is bounded by $|G^T_\infty| \times |S|^V$, where $|S|$ is the number of constants in S, $|G^T_\infty|$ is the number of literals in G^T_∞ and V is the number of constrained variables in G^T_∞.*

Proof. There are $|S|^V$ ways to instantiate the constrained variables in G^T_∞. So there are at most $|G^T_\infty| \times |S|^V$ literals in a saturation $Ax(T) \cup S$ by $\mathcal{PC}$. By Theorem 2, we can conclude that there are at most $|G^T_\infty| \times |S|^V$ literals. □

We notice that Theorems 2 and 3 straightforwardly generalize to theories axiomatized by sets of unit clauses as it is apparent by inspecting their proofs.

3.2. Extending Schematic Saturation to non-equational theories

We first discuss some difficulties to generalize Theorem 2 to theories presented by a set of (possibly) non-unit clauses.

Bear in mind that we allow inferences (of $\mathcal{SPC}$) into constrained variables as so to simulate all possible inferences (of $\mathcal{PC}$) into constants. Let us illustrate the importance of this by the following example.

Example 2. Let T be the theory axiomatized by the following set of clauses

$$X = Y \vee Y = Z \vee Z = X$$
$$f(X) = g(X)$$

G^T_0 contains the following clauses

$$x = y \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$
$$x \neq y \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$
$$f(x) = y \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$
$$g(x) = y \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$

Suppose that no inference into (constrained and unconstrained) variables is possible in Schematic Saturation. Then it is easy to see that G^T_∞ is exactly $Ax(T) \cup G^T_0$. Now let us consider the set $S = \{f(c) = c'\}$. The saturation of $Ax(T) \cup S$ by $\mathcal{PC}$ generates the clause

$$f(Y) = c' \vee Y = Z \vee Z = c$$

which is neither subsumed by another clause nor schematized by any clause in G^T_∞. The problem is that in Schematic Saturation, inferences into variables are excluded, but in a saturation of $Ax(T) \cup S$ there might be inferences from a variable into a constant.

Now assume that inferences into constrained variables are possible in Schematic Saturation, then the clause

$$f(Y) = c' \vee Y = Z \vee Z = c$$

is schematized by the clause

$$f(Y) = y \vee Y = Z \vee Z = x \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$

which is inferred from the clauses

$$X = Y \vee Y = Z \vee Z = X$$
$$f(x) = y \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top \qquad \square$$

Unfortunately, inferences into constrained variables may have some undesired consequences; especially, when they involve two clauses containing only constrained variables, as illustrated by the following example.

Example 3. Let $\mathcal{SC}^2$ be the theory finitely presented by the following set $Ax(\mathcal{SC})$ of clauses:

$$\begin{aligned}
&(Sel): && s_1\big(c(x_1,\ldots,x_n)\big) = x_1\\
&&&\vdots\\
&&& s_n\big(c(x_1,\ldots,x_n)\big) = x_n\\
&(Injc): && c(x_1,\ldots,x_n) = c(y_1,\ldots,y_n) \Rightarrow x_1 = y_1\\
&&&\vdots\\
&&& c(x_1,\ldots,x_n) = c(y_1,\ldots,y_n) \Rightarrow x_n = y_n
\end{aligned}$$

The clause

$$c(x_1,\ldots,x_n) = x_0 \parallel x_0 \preccurlyeq c^\top \wedge \cdots \wedge x_n \preccurlyeq c^\top$$

will paramodulate with the clause

$$c(x_1,\ldots,x_n) = c(y_1,\ldots,y_n) \Rightarrow x_1 = y_1$$

to generate

$$x_0 = c(y_1,\ldots,y_n) \Rightarrow x_1 = y_1 \parallel x_0 \preccurlyeq c^\top \wedge \cdots \wedge x_n \preccurlyeq c^\top$$

The latter again paramodulates with

$$c(x_1,\ldots,x_n) = x_0 \parallel x_0 \preccurlyeq c^\top \wedge \cdots \wedge x_n \preccurlyeq c^\top$$

to generate

$$x_0 = y_0 \Rightarrow x_1 = y_1 \parallel x_0 \preccurlyeq c^\top \wedge y_n \preccurlyeq c^\top \wedge x_1 \preccurlyeq c^\top \wedge y_1 \preccurlyeq c^\top$$

This clause will paramodulate with a renamed version of itself to generate a bigger clause and so on. Thus Schematic Saturation will diverge, although we can show that any saturation by $\mathcal{PC}$ on instances of the clause terminates. $\square$

Another problem when simulating $\mathcal{PC}$ is that Schematic Saturation may introduce infinitely many new constrained variables within a clause which contains unconstrained variables. Again, saturation may halt but Schematic Saturation does not. To illustrate this point, let us consider the example of the theory of arrays.

[2] The theory we consider here is an adaptation of the theory of recursively data structures considered by D.C. Oppen in [24].

Schematic Deletion

$$\frac{S \cup \{D \vee l_1 \vee \cdots \vee l_n \parallel \phi\}}{S}$$

if

- $D \vee l_1 \vee \cdots \vee l_n \parallel \phi$ is a non-unit clause containing only equalities or disequalities between constrained variables, or else
- $n \geqslant 0$, $D \parallel \phi$ is a constrained variant of some clause $D' \parallel \phi'$ in S, and $l_i \parallel \phi$ is a constrained variant of $l \parallel \phi$, where l is a non-maximal literal in D, for $i = 1, \ldots, n$.

Fig. 5. Schematic deletion rule.

Example 4. The theory $\mathcal{A}$ of arrays is axiomatized by the following finite set $Ax(\mathcal{A})$ of axioms, where A, I, J, E are implicitly universally quantified variables:

$$select\big(store(A, I, E), I\big) = E$$

$$I = J \vee select\big(store(A, I, E), J\big) = select(A, J)$$

In [2], it is shown that for every set S of ground flat $\Sigma_{\mathcal{A}}$-literals, any saturation of $Ax(\mathcal{A}) \cup S$ by $\mathcal{PC}$ is finite.

Unfortunately, Schematic Saturation diverges. In fact, it generates the clause

$$select(x, I) = select(z, I) \vee y = I \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \wedge z \preccurlyeq c^{\top}$$

which will paramodulate with a renamed copy of itself, i.e.

$$select\big(x', I'\big) = select\big(z', I'\big) \vee y' = I' \parallel x' \preccurlyeq c^{\top} \wedge y' \preccurlyeq c^{\top} \wedge z' \preccurlyeq c^{\top}$$

to generate a clause of a new form, namely

$$select(x, I) = select(z, I) \vee y = I \vee w = I \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \wedge z \preccurlyeq c^{\top} \wedge w \preccurlyeq c^{\top}$$

The process continues to generate longer and longer clauses so that Schematic Saturation will diverge. □

To cope with the aforementioned problems, we design the *Schematic Deletion* rule to delete constrained clauses that are not relevant for simulating inferences of $\mathcal{PC}$. The key idea of *Schematic Deletion* (cf. Fig. 5) is the following. We would like to schematize every clause generated by $\mathcal{PC}$ but we do not know in advance how many constants there are in the input set of ground flat literals. On the other hand we would like to avoid inferences introducing new constrained variables because that might cause Schematic Saturation to diverge. This happens when inferences introduce unlimited duplications of literals obtained by renaming constrained variables within the same clause. For example the clause

$$select(x, I) = select(z, I) \vee y = I \vee w = I \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top} \wedge z \preccurlyeq c^{\top} \wedge w \preccurlyeq c^{\top}$$

will make Schematic Saturation to diverge with the current version of $\mathcal{SPC}$. The role of *Schematic Deletion* is precisely to avoid this by deleting unnecessary clauses which may generate longer and longer clauses. However if this is done carelessly, we may loose literals on which inferences may apply; and thereby we may loose track of the conclusion of such inferences. For instance, we cannot delete the clause $X = x \vee X = y \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top}$ in which $X = y$ is obtained from $X = x$ by renaming the constrained variable x into y. This is so because both $X = x$ and $X = y$ are maximal in $X = x \vee X = y \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top}$, and there might by inferences from (resp. into) them.

Considering these observations, constrained clauses which can be deleted by *Schematic Deletion* is a disjunction of (dis)equalities between constrained variables, or a disjunction of a constrained clause in Schematic Saturation and non-maximal literals in this constrained clause. We delete disjunctions of (dis)equalities between constrained variables, as they might paramodulate with themselves to generate infinitely many disjunction of (dis)equalities between constrained variables, as in Example 3. We delete disjunctions of a constrained clause in Schematic Saturation and non-maximal literals in this constrained clause because they might paramodulate with themselves generating infinitely many new disjunctions of this kind, like in Example 4.

From now on, by Schematic Saturation we denote the saturation of $Ax(T) \cup G_0^T$ by $\mathcal{SPC}$ augmented with *Schematic Deletion*. We are now ready to prove that Schematic Saturation can be used to check decidability of finitely presented theories.

Theorem 4 *(Automatic termination). Let T be a theory axiomatized by a finite set $Ax(T)$ of clauses, which is saturated with respect to $\mathcal{PC}$. Let G^T_∞ be the set of all clauses in a saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$. Then for every set S of ground flat Σ_T-literals, every clause in a saturation $Ax(T) \cup S$ by $\mathcal{PC}$ is a clause of the form*

$$C \vee l_1 \vee \cdots \vee l_n \qquad (*)$$

where

- *$n \geqslant 0$, and*
- *C is a constraint instance of some clause C' in G^T_∞, and*
- *l_i is*
  - *either a constraint instance of some non-maximal literal in C', or else*
  - *a constraint instance of some maximal (dis)equality between constrained variables in C', or else*
  - *a non-maximal (dis)equality between constants.*

Proof. The proof is by induction on the length of derivations of $\mathcal{PC}$. The base case is obvious. For the inductive case, we need to show three facts:

1. each clause added in the process of saturation of $Ax(T) \cup S$ by $\mathcal{PC}$ is of the form $(*)$, and
2. if a clause is deleted by *Subsumption* or by *Tautology Deletion* from (or simplified by *Simplification* in) the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$, then all clauses containing a constraint instance of the latter will also be deleted from (or simplified in) the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$, and
3. if a clause is deleted by *Schematic Deletion* from the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$, then all constraint instances of this clause are of the form $(*)$.

Proof of (1). Let us start with the rule *Right Paramodulation* of $\mathcal{PC}$. By induction hypothesis $\Gamma \Rightarrow \Delta, l[u'] = r$ and $\Pi \Rightarrow \Sigma, u = t$ have the form $(*)$, i.e. there are some clause $D \parallel \phi$ and $D' \parallel \phi'$ in G^T_∞ and a substitution θ such that

- $\theta(D \vee l_1 \vee \cdots \vee l_n \parallel \phi) \equiv \Gamma \Rightarrow \Delta, l[u'] = r$, and
- $\theta(D' \vee l'_1 \vee \cdots \vee l'_m \parallel \phi') \equiv \Pi \Rightarrow \Sigma, u = t$.

We consider all the positions where the rule *Right Paramodulation* of $\mathcal{PC}$ can be performed.

- If the *Right Paramodulation* inference of $\mathcal{PC}$ is performed at $\theta(D)$ and $\theta(D')$, then there must exist a *Right paramodulation* inference of $\mathcal{SPC}$ in the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$, whose premises are $D \parallel \phi$ and $D' \parallel \phi'$ with conclusion $D'' \parallel \phi \wedge \phi'$ such that we can extend θ so that $\theta(D'' \vee l_1 \vee \cdots \vee l_n \vee l'_1 \vee \cdots \vee l'_m \parallel \phi \wedge \phi') \equiv \sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)$. We consider two subcases:
  - $\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)$ only contains (dis)equalities between constants: then $D'' \parallel \phi \wedge \phi'$ only contains (dis)equalities between constrained variables, and hence it will be deleted from the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$. However this means that $\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)$ is of the form $(*)$.
  - $\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)$ contains at least a non-constant term: then it is a constraint instance of $D'' \vee l_1 \vee \cdots \vee l_n \vee l'_1 \vee \cdots \vee l'_m \parallel \phi \wedge \phi'$, where $D'' \parallel \phi \wedge \phi'$ persists in the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$. Hence $\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)$ is of the form $(*)$.
- If the *Right Paramodulation* of $\mathcal{PC}$ is performed at $\theta(l_1 \vee \cdots \vee l_n)$ and $\theta(l'_1 \vee \cdots \vee l'_m)$, meaning that selected equalities are in these clauses and they must be maximal equalities between constants (we are considering *Right Paramodulation* and hence omit disequalities). By the induction hypothesis we must have that $D \equiv D_1 \vee x_1 = y_1$ and $D' \equiv D'_1 \vee x'_1 = y'_1$, where x_1, y_1, x'_1, y'_1 are constrained variables. But then there must exist a *Right Paramodulation* inference of $\mathcal{SPC}$ in the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$, whose premises are $D \parallel \phi$ and $D' \parallel \phi'$ with conclusion $(D_1 \vee D'_1 \vee y_1 = y'_1 \parallel \phi \wedge \phi')[x'_1 \to x_1]$. And thus $\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)$ is of the form $(*)$.
- If the *Right Paramodulation* of $\mathcal{PC}$ is performed at $\theta(l_1 \vee \cdots \vee l_n)$ and $\theta(D')$, then the selected equality in $\theta(l_1 \vee \cdots \vee l_n)$ is an equality between constants. By the induction hypothesis we must have that $D \equiv D_1 \vee x_1 = y_1$, where x_1, y_1 are constrained variables. Then there must exist a *Right Paramodulation* inference of $\mathcal{SPC}$ in the saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$, whose premises are $D \parallel \phi$ and $D' \parallel \phi'$ with conclusion $(D_1 \vee D' \parallel \phi \wedge \phi')[x_1 \to y_1]$. And $\sigma(\Gamma, \Pi \Rightarrow \Delta, \Sigma, l[t] = r)$ is still of the form $(*)$.

The rule *Left Paramodulation* of $\mathcal{PC}$ is handled exactly in the same way as *Right Paramodulation* of $\mathcal{PC}$.

For *Reflection* of $\mathcal{PC}$, if the inference is performed in the C part then it is simulated by a *Reflection* inference of $\mathcal{SPC}$ applied to C'. If the inference is performed at the $l_1 \vee \cdots \vee l_n$ part, then the disequalities involved in the *Reflection* inference must be disequalities between constants. So the conclusion is still of the form $(*)$.

Eq. Factoring of $\mathcal{PC}$ can be handled similarly to *Right Paramodulation* of $\mathcal{PC}$.

Proof of (2). Let us consider *Subsumption* of $\mathcal{SPC}$. The case (b) of *Subsumption* is just a matter of deleting duplicates. For case (a), assume that there are a clause A deleted from the saturation of $Ax(T) \cup G_0^T$ by $\mathcal{SPC}$. Then there must exist a clause $C \in Ax(T)$ and some substitution θ such that $\theta(C) \subseteq A$. Now assume that B is a clause in the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$, which contains a constraint instance of A. Since all the clauses in $Ax(T)$ persist, there must be a substitution θ' such that $\theta'(C) \subseteq B$. Therefore B must also be deleted from the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$.

A similar argument can be given for *Simplification* of $\mathcal{SPC}$. Assume that there are a clause $C[l'] \parallel \phi$ in the saturation of $Ax(T) \cup G_0^T$ by $\mathcal{SPC}$ simplified by an equality $l = r$ ($l = r \in Ax(T)$) into $C[\theta(r)] \parallel \phi$. Let σ be a substitution such that $\sigma(C[l'])$ is a constraint instance of $C[l'] \parallel \phi$. Since $l = r$ persists in the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$, any clause containing $\sigma(C[l'])$ in the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$ will be simplified into a clause containing $\sigma(C[\theta(r)])$.

For the *Tautology Deletion* rule of $\mathcal{SPC}$, it is easy to see that a clause containing a constraint instance of a tautology is also a tautology.

For the *Deletion* rule of $\mathcal{SPC}$, notice that clauses with an unsatisfiable constraint have no constraint instances.

Proof of (3). Let us consider two cases of *Schematic Deletion*. In the first case, the fact that $D \vee l_1 \vee \cdots \vee l_n \parallel \phi$ is a non-unit clause containing only equalities or disequalities between constrained variables means that it is a schematization of disjunctions of (dis)equalities between constants. We can easily see that any disjunction of (dis)equalities between constants is of the form $(*)$. In the second case, the fact that $D \parallel \phi$ is a constrained variant of some clause $D' \parallel \phi'$ in S, and $l_i \parallel \phi$ is a constrained variant of some non-maximal literal $l \parallel \phi$ in D means that any constraint instance of $D \vee l_1 \vee \cdots \vee l_n \parallel \phi$ is of the form $(*)$. □

Example 5. Consider again the clauses of Example 3 in the presence of the *Schematic Deletion* rule, the clause

$$x_0 = y_0 \Rightarrow x_1 = y_1 \parallel x_0 \preccurlyeq c^\top \wedge y_0 \preccurlyeq c^\top \wedge x_1 \preccurlyeq c^\top \wedge y_1 \preccurlyeq c^\top$$

will be immediately deleted. Finally it is easy to see that Schematic Saturation will halt and the set of persistent clauses will contain $Ax(\mathcal{SC}) \cup G_0^{\mathcal{SC}}$ and the following clauses:

$$x_0 = c(y_1, \ldots, y_n) \Rightarrow x_1 = y_1 \parallel x_0 \preccurlyeq c^\top \wedge \cdots \wedge x_n \preccurlyeq c^\top$$
$$\vdots$$
$$x_0 = c(y_1, \ldots, y_n) \Rightarrow x_n = y_n \parallel x_0 \preccurlyeq c^\top \wedge \cdots \wedge x_n \preccurlyeq c^\top$$

For Example 4, in the presence of *Schematic Deletion*, the clauses generated by self-paramodulations, i.e. clauses of the form

$$select(x_1, I) = select(x_2, I) \vee x_3 = I \vee \cdots \vee x_n = I \parallel x_1 \preccurlyeq c^\top \wedge \cdots \wedge x_n \preccurlyeq c^\top$$

will be deleted by applying *Schematic Deletion*. This is so because the clause $select(x, I) = select(z, I) \vee y = I \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top \wedge z \preccurlyeq c^\top$ already persists and the literals $x_3 = I, \ldots, x_n = I$ are actually constrained variants of the literal $y = I$. Therefore, the set of persistent clauses will contain $Ax(\mathcal{A}) \cup G_0^{\mathcal{A}}$ and the following clauses:

$$select(x, I) = select(z, I) \vee y = I \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top \wedge z \preccurlyeq c^\top$$
$$select(x, I) = z \vee y = I \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top \wedge z \preccurlyeq c^\top$$

By Theorem 4, we conclude that $\mathcal{PC}$ is a satisfiability procedure for theory $\mathcal{SC}$ and the theory $\mathcal{A}$ of arrays. □

Similarly to the equational case, we can also determine an upper bound on the number of clauses generated in saturation. But this time we have non-unit clauses and hence the bound on the number of persisting clauses in a saturation by $\mathcal{PC}$ becomes exponential.

Theorem 5 *(Automatic complexity). Let T be a theory axiomatized by a finite set $Ax(T)$ of clauses, which is saturated with respect to $\mathcal{PC}$. Let G_∞^T be the set of all clauses in a finite saturation of $Ax(T) \cup G_0^T$ by $\mathcal{SPC}$. Then for every set S of ground flat Σ_T-literals, the number of clauses in a saturation $Ax(T) \cup S$ by $\mathcal{PC}$ is bounded by $2^{|G_\infty^T| \times |S|^V}$, where $|S|$ is the number of constants in S, $|G_\infty^T|$ is the number of literals in G_∞^T and V is the number of constrained variables in G_∞^T.*

Proof. There are $|S|^V$ ways to instantiate constrained variables in G_∞^T. Hence there are at most $|G_\infty^T| \times |S|^V$ literals in a saturation $Ax(T) \cup S$ by $\mathcal{PC}$. According to Theorem 4, all clauses in a saturation $Ax(T) \cup S$ by $\mathcal{PC}$ has the form $(*)$, which is actually a disjunction of literals. If we omit duplications of literals, then the number of such clauses is equal to the number of subsets of the set of all literals. Thus there are at most $2^{|G_\infty^T| \times |S|^V}$ clauses built out of $|G_\infty^T| \times |S|^V$ literals. □

4. Automatic combinability

In this section, we consider two possible approaches for building satisfiability procedures for unions of theories: either using $\mathcal{PC}$ on the union of the axioms of the theories being combined; or modularly combining satisfiability procedures for the component theories by using the Nelson–Oppen combination method [22]. The first method only works for some theories presented by a finite set of formulae while the latter applies to any combination of *stably infinite* theories. We show that $\mathcal{SPC}$ can check whether $\mathcal{PC}$ decide some unions of finitely presented theories, as well as whether a theory can be efficiently combined with other theories using the Nelson–Oppen method.

4.1. Modular termination

We study conditions under which the theory $T_1 \cup T_2$ admits a paramodulation-based satisfiability procedure, provided that T_1 and T_2 are disjoint theories admitting paramodulation-based satisfiability procedures. To this end, we have to consider termination of any saturation of $Ax(T_1) \cup Ax(T_2) \cup S$ by $\mathcal{PC}$ for an arbitrary set of ground flat literals S. Since both T_1 and T_2 admit a paramodulation-based satisfiability procedure, the only source of non-termination when considering their union (i.e., $T_1 \cup T_2$) is due to inferences across theories. More precisely, such inferences can only take place on variables, and constants as T_1 and T_2 are signature disjoint. It is easy to see that inferences on constants generate finitely many clauses while inferences on variables might generate clauses containing mixed terms and hence might cause non-termination. Thus it seems sufficient to exclude across theories inferences on variables to ensure modular termination. Before proving that this is indeed the case, we need to introduce some technical results. The following concept of variable-active clause is taken from [1].

Definition 10 *(Variable-active clause).* A clause C is variable-active with respect to an ordering $\succ$ if C contains a maximal (with respect to $\succ$) literal of the form $X = t$, where X is a variable not occurring in $Var(t)$. A constrained clause is variable-active with respect to $\succ$ if one of its constraint instances is variable-active with respect to $\succ$.

It follows from Definition 10 that checking whether a clause is variable-active or not can be done syntactically.

From now on, when we say that a clause C is variable-active we mean that C is variable-active with respect to the ordering $\succ$ used by $\mathcal{PC}$.

Lemma 1. *Let T be a theory axiomatized by a finite set $Ax(T)$ of clauses, which is saturated with respect to $\mathcal{PC}$. Assume that any saturation of $Ax(T) \cup G_0^T$ by $\mathcal{SPC}$ is finite and does not contain any variable-active clauses. Then for every set S of ground flat literals, any saturation of $Ax(T) \cup S$ by $\mathcal{PC}$ does not contain any variable-active clauses.*

Proof. The proof is by contradiction. Assume that a saturation of $Ax(T) \cup S$ by $\mathcal{PC}$ contains a variable-active clause D. By Theorem 4, D must have the form

$$C \vee l_1 \vee \cdots \vee l_n$$

where

- $n \geqslant 0$, and
- C is a constraint instance of some clause C' in G_∞^T, and
- l_i is
  - either a constraint instance of some non-maximal literal in C', or else
  - a constraint instance of some maximal (dis)equality between constrained variables in C', or else
  - a non-maximal (dis)equality between constants.

Since D is variable-active, D must contain a maximal literal $X = t$ such that $X \notin Vars(t)$. But then $X = t$ must be in C, meaning also that C' is a variable-active constrained clause. The fact that C' in G_∞^T would contradict the hypothesis of the lemma. □

In the spirit of [1], the following lemma provides a sufficient condition for termination of $\mathcal{PC}$ on the union of the axioms of theories along with a set of ground flat literals. This condition is based on the notion of variable-active clauses.

Lemma 2. *Let T_i be a theory axiomatized by a finite set $Ax(T_i)$ of clauses, which is saturated with respect to $\mathcal{PC}$ for $i = 1, 2$. Assume that*

- *the signatures of T_1 and T_2 are disjoint, and*
- *for every set S of ground flat Σ_i-literals, any saturation of $Ax(T_i) \cup S$ by $\mathcal{PC}$ is finite and does not contain any variable-active clauses, for $i = 1, 2$.*

Then for every set S of ground flat $\Sigma_1 \cup \Sigma_2$-literals, any saturation of $Ax(T_1) \cup Ax(T_2) \cup S$ by $\mathcal{PC}$ is finite.

Proof. We consider all possible across-theories inferences between clauses. Since the theories have disjoint signatures, an across-theories inference must be one of the following types:

1. from constants into constants between clauses containing only constants. This kind of inference generate only a finite number of clauses since we have finitely many constants.
2. from constants into constants between clauses containing only constants or variables. This kind of inference is possible only if one of the premises is a variable-active clause, and that would contradict the hypothesis of the lemma.
3. from variables into arbitrary terms. This kind of inference is possible only if one of the premises is a variable-active clause, and that would again contradict the hypothesis of the lemma.

Therefore under the hypothesis of the lemma, across-theories inferences generate finitely many clauses and hence any saturation of $Ax(T_1) \cup Ax(T_2) \cup S$ by $\mathcal{PC}$ is finite. □

Our main result about the modular termination of paramodulation-based satisfiability procedures is an immediate consequence of Lemmas 1 and 2.

Theorem 6 *(Automatic modular termination). Let T_i be a theory axiomatized by a finite set $Ax(T_i)$ of clauses, which is already saturated with respect to $\mathcal{PC}$ for $i = 1, 2$. Assume that*

- *the signatures of T_1 and T_2 are disjoint, and*
- *any saturation of $Ax(T_i) \cup G_0^{T_i}$ by $\mathcal{SPC}$ is finite and does not contain any variable-active clauses, for $i = 1, 2$.*

Then, $\mathcal{PC}$ is a satisfiability procedure for $T_1 \cup T_2$.

Example 6. In [2], it is shown that any saturation of an arbitrary set of ground flat literals and the union of the axioms of the theory $\mathcal{L}$ of lists and the theory $\mathcal{A}$ of arrays is finite. We prove this result by using Schematic Saturation.

In fact, by Examples 1 and 4, Schematic Saturation for both $\mathcal{L}$ and $\mathcal{A}$ is finite and does not contain any variable-active clauses. It follows from Theorem 6, $\mathcal{PC}$ is a satisfiability procedure for $\mathcal{L} \cup \mathcal{A}$. □

Theorem 6 provides a modular decidability result for finitely presented theories having a paramodulation-based decision procedure. However, the absence of variable-active clauses in any finite saturation is too strong a requirement as there exist theories not satisfying the hypothesis of Theorem 6 that can still be combined with other theories using the Nelson–Oppen method if they are stably infinite. Consider the following example.

Example 7. Let T be the theory presented by the clause

$$f(X) = a \vee X = Y \vee f(Y) = a$$

In our settings, G_0^T contains

$$x = y \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$
$$x \neq y \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$
$$x = a \parallel x \preccurlyeq c^\top$$
$$x = b \parallel x \preccurlyeq c^\top$$
$$f(x) = y \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$

and Schematic Saturation will contain the axiom, G_0^T and the following set of clauses

$$X \neq x \vee f(y) = a \vee f(X) = a \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$
$$f(x) = a \parallel x \preccurlyeq c^\top$$
$$f(x) = a \vee f(y) = a \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$
$$X = y \vee x = a \vee f(X) = a \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$$
$$X = y \vee x = a \vee f(X) = a \vee f(z) = a \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top \wedge z \preccurlyeq c^\top$$

By Theorem 4, T has decidable satisfiability problem. However, the theory presentation has a variable-active clause, so Theorem 6 does not apply. But we can show that the theory is stably infinite. Therefore, we can still combine T with theories having similar properties or with theories being stably infinite (and possibly non finitely axiomatizable) using the Nelson–Oppen method. In fact, proving that T is stably infinite by hand is not so straightforward. Fortunately, we provide, in the next section (cf. Theorem 7), a method of checking stable infiniteness of finitely presented theories. □

4.2. Stable infiniteness

The Nelson–Oppen combination method [22] allows us to combine satisfiability procedures for the class of stably infinite theories in a modular way. Although stable infiniteness is undecidable in general (see, e.g., [5] for more details), it is interesting to develop automated techniques to prove it for a subclass of first-order theories, in particular those admitting paramodulation-based satisfiability procedures. Here we develop such a technique by using Schematic Saturation.

Definition 11 *(Elementary constrained clause).* A constrained clause is elementary if one of its constraint instance is elementary.

Definition 12 *(Finite cardinality clause).* A clause is a *finite cardinality clause* if it has the form

$$\bigvee_{0\leqslant j\neq k\leqslant n} (x_j = x_k)$$

where n is a positive integer and x_i is a variable, for $i = 0, \ldots, n$.

The following result follows from the compactness of first-order logic (see, e.g., [33]).

Lemma 3. *Let T be a satisfiable set of formulae. If T has no infinite models then T entails a finite cardinality clause.*

The following result applies to paramodulation calculi, which are *stable under signature extension* (for instance $\mathcal{PC}$), i.e., extending the initial signature with new symbols does not destroy completeness.[3]

Lemma 4. *Let T be a consistent theory axiomatized by a finite set $Ax(T)$ of clauses and S be a finite T-satisfiable set of ground literals. If $T \cup S$ entails a finite cardinality clause, then any saturation of $Ax(T) \cup S$ by $\mathcal{PC}$ contains a non-ground elementary clause.*

Proof. The proof uses the *model generation* technique (see, e.g., [23] for more details). Let S be a set of ground clauses and C be a clause in S. Then $Gen(C) = \{l \to r\}$, and C is said to *generate* the rule $\{l \to r\}$, if and only if, C is of the form $\Gamma \Rightarrow \Delta, l = r$ and the following conditions hold:

1. $R_C^* \not\models C$,
2. $l \succ r$ and $l \succ \Gamma$ and $l = r \succ^{mul} u = v$ for all $u = v \in \Delta$, where $\succ^{mul}$ is the multiset extension of $\succ$ (see, e.g., [23] for more details),
3. l is irreducible by R_C,
4. $R_C^* \not\models r = t'$ for every $l = t' \in \Delta$,

where $R_C = \bigcup_{C \succ D} Gen(D)$, and R_C^* is the congruence induced by R_C. In all other cases, $Gen(C) = \emptyset$. Finally, R denotes the set of all rules generated by clauses of S, that is $R = \bigcup_{D \in S} Gen(D)$.

Now assume that $T \cup S$ entails a finite cardinality clause with n distinct variables. Let S' be the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$. Since S' and $Ax(T) \cup S$ are logically equivalent, we have that S' entails the same finite cardinality clause. This also means that $S' \cup \bigcup_{0\leqslant j\neq k\leqslant n}\{c_j \neq c_k\}$ is unsatisfiable, where $c_0, \ldots, c_n$ are new constants. Let $R_{S'}$ be the set of all rules generated by the clauses in $grd(S')$, where $grd(S')$ denotes the set of all ground instances of the clauses in S'. By model generation technique, $S' \cup \bigcup_{0\leqslant j\neq k\leqslant n}\{c_j \neq c_k\}$ is unsatisfiable only if there exists a constant c_i reducible by $R_{S'}$, because otherwise $S' \cup \bigcup_{0\leqslant j\neq k\leqslant n}\{c_j \neq c_k\}$ is satisfiable. We can without loss of generality assume that c_i is the smallest (wrt. $\succ$) among the constants reducible by $R_{S'}$.

Assume that c_i is reduced by a rule $c_i \to r$ in $R_{S'}$. Then $c_i \to r$ must be in a clause C which generates $c_i \to r$. Since c_i is a constant, r must also be a constant and C must be a disjunction of equalities or disequalities between constants. Assume that C is a ground instance of some clause C' in S'. As c_i is a fresh constant and it is not in S', C generates the rule $c_i \to r$ only if C' contains an equality of the form $x = y$, where at least x must be a variable. Therefore, C' must have the form $x = y \vee x_1 \bowtie y_1 \vee \cdots \vee x_n \bowtie y_n$, where $n \geqslant 0$, $y, x_1, y_1, \ldots, x_n, y_n$ are constants or variables, x is a variable, and $\bowtie \in \{=, \neq\}$.

We prove that C' is a non-ground elementary clause. To this end, it is sufficient to show that C' does not contain any disequalities. Since c_i is the smallest reducible constant, C' must not contain any disequalities containing a constant (occurring in S'), otherwise condition 2 of model generation would not be satisfied and $c_i \to r$ would not be generated. Assume that C' contains a disequality between variables, say $x_i \neq y_i$ ($i \in \{1, \ldots, n\}$). If $x_i \equiv x$ or $y_i \equiv x$ then C contains both $c_i = r$ and $c_i \neq r'$. In this case, condition 2 of model generation would not again be satisfied and consequently C could not

[3] This is not restrictive because many state of the art paramodulation-based provers enjoy this property, except those which interpret ordering constraints as symbolic constraint solving problems in the original signature (see [23,5] for a more detailed discussion).

generate $c_i \to r$. So x_i, y_i must be different from x. But then *Reflection* applies to C' to infer a clause C'' containing $x = y$, and which has one literal less. We now show that there is a ground instance of C'' which contains $c_i = r$ as maximal literal and is smaller than every ground instance of C' which contains $c_i = r$. Indeed, consider the instantiation σ in which x is instantiated with c_i, y is instantiated with r and each other variable is instantiated with the smallest constant wrt. $\succ$. Clearly $\sigma(C'')$ is that ground instance. This means that C could not generate $c_i \to r$ because $\sigma(C'')$ would generate $c_i \to r$. Summing up, in all cases C' must contain no disequalities and this completes the proof of the lemma. □

By Theorem 4, if a non-ground elementary clause C occurs in a saturation of a set S of ground flat literals along with $Ax(T)$, then there must exist an elementary constrained clause C' containing a unconstrained variable in Schematic Saturation for T. By analyzing the form of C' we can show that either Schematic Saturation diverges or it will derive the trivial equality $X = Y$. This result is formally stated in the following theorem.

Theorem 7 *(Automatic stable infiniteness). Let T be a consistent theory axiomatized by a finite set $Ax(T)$ of clauses, which is saturated with respect to $\mathcal{PC}$. Let G^T_∞ be the set of all clauses generated in a finite saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$. If G^T_∞ does not contain $X = Y$, then T is stably infinite.*

Proof. The proof is by contradiction. Assume that T is not stably infinite. Then there exists a T-satisfiable set S of ground flat Σ_T-literals such that $T \cup S$ has no infinite models. By Lemma 3, $T \cup S$ must entail a finite cardinality clause. Let S' be the saturation of $Ax(T) \cup S$ by $\mathcal{PC}$. By Lemma 4, S' contains a non-ground elementary clause C. We consider two cases:

1. C is a unit clause, i.e. $C \equiv X = y$ where X is a variable and y is a variable or a constant. If y is a variable, then by Theorem 4, $X = y$ must be a constraint instance of some clause in G^T_∞, which must be an equality between two unconstrained variables, i.e. $X = Y$; a contradiction. Now, consider the case where y is a constant. Then again by Theorem 4, $X = y$ is a constraint instance of the clause $X = y \parallel y \preccurlyeq c^\top$. But $X = y \parallel y \preccurlyeq c^\top$ will paramodulate with a renamed version of itself to generate $X = X'$ in G^T_∞; a contradiction.
2. C is a non-unit clause. By Theorem 4, C is a clause of the form $D \vee l_1 \vee \cdots \vee l_n$, where D is a constraint instance of some clause D' in G^T_∞, and (for $i = 1, \ldots, n$ and $n \geqslant 0$) either l_i is constraint instance of some literal in D' or a (dis)equality between constants. We argue that D' is non-ground elementary (i.e. containing an unconstrained variable). This is so because if D' is ground any constraint instance of D' would be ground and this would imply that D and C are ground. We can assume wlog. that every equality in D' is non-ground (because otherwise an application of Paramodulation between D' and the disequality $x \neq y \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$ followed by Reflection would generate a new clause D''; if D'' is not already an elementary clause containing only non-ground equalities, then we repeat the same process on D'' and eventually get an elementary clause containing only non-ground equalities). If D' is a unit clause, then we are in case (1). Let us consider D' to be non-unit, then D' has one of the following forms, where X, Y, Z are unconstrained variables:
   (a) $X = x \vee X = y \vee D_1 \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top \wedge \phi$: then D' will paramodulate with a renamed copy of itself, i.e. $X' = x' \vee X' = y' \vee D'_1 \parallel x' \preccurlyeq c^\top \wedge y' \preccurlyeq c^\top \wedge \phi'$, to infer the clause $X = X' \vee Y = y \vee Y' = y' \vee (D_1[x \to x']) \vee D'_1 \parallel ((x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top \wedge \phi \wedge \phi')[x \to x'])$. Notice that the paramodulation is performed at constrained variables, and its conclusion, which contains new unconstrained variables, is longer than its premises. *Schematic Deletion* cannot be applied to the new clause since it contains new unconstrained variables. This new clause will again paramodulate with a renamed version of itself to generate an even longer clause with new unconstrained variables. The process continues infinitely to generate longer and longer clauses with new unconstrained variables. Therefore G^T_∞ will be infinite.
   (b) $X = x \vee X = Z \vee D_1 \parallel x \preccurlyeq c^\top \wedge \phi$: then D' will paramodulate with $x' = y' \parallel x' \preccurlyeq c^\top \wedge y' \preccurlyeq c^\top$ to infer the clause $X = x \vee X = y' \vee (D_1[Z \to x']) \parallel ((x \preccurlyeq c^\top \wedge x' \preccurlyeq c^\top \wedge y' \preccurlyeq c^\top \wedge \phi)[Z \to x'])$, which is in the form (a); and G^T_∞ will be infinite.
   (c) $X = Y \vee X = Z \vee D_1 \parallel \phi$: then D' will paramodulate with $x = y \parallel x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top$ to generate the clause $X = y \vee X = Z \vee (D_1[Y \to x]) \parallel ((x \preccurlyeq c^\top \wedge y \preccurlyeq c^\top \wedge \phi)[Y \to x])$, which is in the form (b). But then G^T_∞ will also be infinite.

Summing up, in all cases if T is not stably infinite, then either the trivial equality $X = Y$ is in G^T_∞ or G^T_∞ is infinite, and this contradicts the hypothesis of the theorem. □

Example 8. Consider again Example 7. We can see that Schematic Saturation for T terminates and does not derive the trivial equality $X = Y$. By Theorem 7, T is stably infinite. □

4.3. Deduction completeness

The crux of the Nelson–Oppen combination method is to exchange entailed equalities between satisfiability procedures. There does not seem to be any problem in using a satisfiability procedure to check whether a set S of literals entails a formula ϕ in a theory since we can check whether S and the negation of ϕ is unsatisfiable. However, to implement the

Nelson–Oppen combination method efficiently, the satisfiability procedure for the component theories must be capable to derive the ground elementary clauses to be exchanged with other procedures.

As mentioned previously, it is sufficient to exchange only ground elementary equalities when combining convex theories without affecting the correctness of the Nelson–Oppen method. The reader may wonder why we do not study the convexity of theories in our framework as it is crucial for the efficiency of combination methods (see, e.g., [22] for a discussion on this point). Fortunately, it is possible to come up with a semi-automatic check for convexity by re-using a result of Lewis [17], which provides a sufficient (and necessary) condition for the existence of a renaming function bringing a set S of clauses to Horn form by checking the consistency of a certain associated set S_a of clauses. Indeed, the consistency of S_a can be checked by using any refutation complete calculus (such as $\mathcal{PC}$). Since any Horn theory is convex, the successful application of Lewis' method entitles us to conclude that S is convex as it can be presented by a Horn theory.

Below, we show that for a theory presented by a set of Horn clauses, if we consider a variant of $\mathcal{PC}$ with negative selection, then its paramodulation-based satisfiability procedure (if exists) is deduction complete.

Theorem 8 *(Automatic deduction completeness). Assume*

- *$\mathcal{PC}$ uses a selection function sel such that for each clause C, $sel(C)$ contains either a negative literal in C, or a maximal positive literal if C does not contain any negative literals; and similarly for $\mathcal{SPC}$;*
- *T to be a theory axiomatized by a finite set $Ax(T)$ of Horn clauses, which is saturated with respect to $\mathcal{PC}$;*
- *G^T_∞ to be the set of all clauses in a finite saturation of $Ax(T) \cup G^T_0$ by $\mathcal{SPC}$ such that G^T_∞ does contains the clause $X = Y$.*

Then, $\mathcal{SP}$ is a deduction complete T-satisfiability procedure with respect to elementary equalities.

Proof. Let S be a finite T-satisfiable set of ground flat literals and S' be a saturation of $Ax(T) \cup S$ by $\mathcal{PC}$. We need to show that for every elementary equality $c = c'$ such that $Ax(T) \cup S \models c = c'$, we have that $S'_e \models c = c'$, where S'_e is the subset containing all elementary equalities in S'. Assume that there is some elementary equality $c = c'$ such that $T \cup S \models c = c'$. Reasoning by refutation, $T \cup S \models c = c'$ iff $S \wedge c \neq c'$ is T-unsatisfiable. Hence, it must be possible to derive the empty clause by applying $\mathcal{PC}$ to the set $S' \cup \{c \neq c'\}$. We are going to show that it must be possible to derive the empty clause using $\mathcal{PC}$ from $S'_e \cup \{c \neq c'\}$. The proof is by induction on the order of $c \neq c'$ wrt. the ordering $\succ$.

The basic case, where c and c' are the same smallest constant wrt. the ordering $\succ$, is obvious. Only $c \neq c'$ would be sufficient to derive the empty clause.

Now let us consider the inductive case. Since S' is T-satisfiable and saturated, only inferences involving both clauses from (or inferred from) S' and $c \neq c'$ can yield the empty clause. Let us analyze such inferences, i.e. inferences between $c \neq c'$ and some clause C' in S'. If there is an inference between $c \neq c'$ and C', then the inference must be a *Left Paramodulation* and the literal selected in C' must be an equality. Also C' only contains constants or variables because c and c' are constants. By the hypotheses of the theorem, *sel* always selects a negative literal in a clause first if the latter contains negative literals. This means that each literal in C' must be an equality. On the other hand, the fact that T is a Horn theory implies that any saturation of $Ax(T) \cup S$ by $\mathcal{PC}$ only contains Horn clauses. Therefore C' must contain exactly one equality. Now if C' contains a variable, then G^T_∞ must contain $X = Y$; that would contradict the assumption of the theorem. If C' only contains constants, then C' is a ground elementary equality, which also means that C' is in S'_e. Moreover the clause inferred from $c \neq c'$ and C' must be a disequality between constants, let us say $c_1 \neq c'_1$. We know that $c_1 \neq c'_1$ is smaller than $c \neq c'$ wrt. the ordering $\succ$. By induction hypothesis, it must be possible to derive the empty clause using $\mathcal{PC}$ from $S'_e \cup \{c_1 \neq c'_1\}$. This means that it must be possible to derive the empty clause using $\mathcal{PC}$ from $S'_e \cup \{c \neq c'\}$, or equivalently, $S'_e \models c = c'$. □

Example 9. Considering again Example 3. Schematic Saturation contains $Ax(\mathcal{SC})$, $G^{\mathcal{SC}}_0$, and the following clauses:

$$x = c(y_1, \ldots, y_n) \Rightarrow x_1 = y_1 \parallel x_1 \preccurlyeq c^\top \wedge x \preccurlyeq c^\top$$
$$\vdots$$
$$x = c(y_1, \ldots, y_n) \Rightarrow x_n = y_n \parallel x_n \preccurlyeq c^\top \wedge x \preccurlyeq c^\top$$

Schematic Saturation is finite and does not contain the equality $X = Y$. By Theorem 8, $\mathcal{SP}$ is a deduction complete $\mathcal{SC}$-satisfiability procedure with respect to elementary equalities. □

For non-Horn theories, the situation becomes more complicated as some inferences on non-unit ground elementary clauses may be blocked due to the ordering used in $\mathcal{PC}$. To illustrate the problem, let us consider the following example.

Example 10. Let S be the following set of clauses

$$i \neq i'$$
$$select(a', i') = e'$$

$$store(a, i, e) = a'$$
$$select(a, i') = e'$$

A saturation of S along with the axioms $Ax(\mathcal{A})$ of the theory of arrays yields $Ax(\mathcal{A}) \cup S$ and the following clauses

$$e' = e \vee i = i'$$
$$select(a, i') = e'$$
$$select(a, i') = e \vee i = i'$$
$$select(a, J) = select(a', J) \vee J = i$$

It is easy to see that $\{i \neq i', e' = e \vee i = i'\}$ implies $e' = e$, and hence we have that S' entails $e' = e$. But the set $\{e' = e \vee i = i'\}$ of ground elementary clauses of S' does not entail $e' = e$ if we consider an ordering $\succ$ such that $e' \succ e \succ i \succ i'$. □

In order to obtain deduction completeness for non-Horn theories, [32] proposes to use a splitting rule (along the lines of [27]) to activate every possible inference among ground elementary clauses and therefore derive sufficiently many disjunctions of ground elementary equalities. The idea of the splitting rule is to split any clause of the form $A \vee B$ into two clauses $A \vee p$ and $B \vee \neg p$, where p is a new propositional variable and A, B do not share any variables. By using this rule, we can split any ground elementary clause into clauses containing exactly one (dis)equality and possibly new propositional variables. Moreover we consider an ordering such that p is the smallest one, and thereby activate every inference on ground elementary (dis)equalities. In this way, as soon as the set of clauses is saturated, we can compute a complete set of ground elementary clauses by eliminating all new propositional variables introduced by splitting. For Example 10, $e' = e \vee i = i'$ would be split into $e' = e \vee p$ and $i = i' \vee \neg p$. But then $i = i' \vee \neg p$ would paramodulate with $i \neq i'$ to yield $i' \neq i' \vee \neg p$ to which *Reflection* applies yielding $\neg p$. Then we can to get rid of the propositional variable p by resolving $e' = e \vee p$ and $\neg p$ to derive $e' = e$. Notice also that if we use splitting, negative selection is no longer necessary. We refer to [32] for more details.

5. Related work

The research described in this paper is in the spirit of the seminal work [20] by McAllester, where it is given a procedure for automatically recognizing presentations of theories whose satisfiability problem can be checked in polynomial time. Other papers (see, e.g., [13,4,10]) have built upon McAllester's work to derive automatic decidability results for larger classes of theories. Unfortunately, the formal framework underlying these works does not allow one to consider equality as built-in, so that available efficient approaches for equational reasoning cannot be used. Our work can be seen as an attempt to overcome this problem by using paramodulation, a state-of-the-art automated reasoning techniques that treats equality as built-in by using efficient rewriting techniques. In contrast with the approaches *à la* McAllester, our investigations do not focus on polynomial-time decidability but are concerned to design automatic checks for properties which are relevant for theorem proving in unions of theories (e.g., stably infiniteness or deduction completeness), which are crucial to embed automated-theorem proving techniques in verification tools.

The work described in this paper unifies and generalizes previous results on paramodulation-based decision procedures for satisfiability problems. In [2], the rewriting-approach to paramodulation-based satisfiability procedures is introduced and applied to some relevant theories for verification such as lists, arrays, and their combination. In [18], an automatic procedure to build paramodulation-based satisfiability procedure is described by using the *meta-saturation* calculus, which simulates (some of) the inferences of paramodulation. In [1], further theories are shown amenable to the rewriting-approach (e.g., records and integer off-sets), the class of variable-inactive theories is defined, and a result for the modular termination of saturation is proved for such a class of theories. In [15], it is shown that—under negative selection—a saturation of theories axiomatized by Horn clauses derives enough implied equalities to guarantee the completeness of the Nelson–Oppen schema. In [16], an automatic method is designed (using the meta-saturation calculus in [18]) for checking a condition for variable-activity (similar to the one proposed in this paper), stable infiniteness, and deduction completeness for finitely presented theories. Finally, an extension of the results in [18,16] has been recently given in [19], to handle theories for which [18, 16] fail (such as the theory of arrays). Schematic Saturation presents a lot of similarities with the meta-saturation of [18]. The main difference is that meta-saturation uses constraints so as to ensure that constrained variables are instantiated by constants and it excludes every inference into variables. However, as illustrated in Example 2, if we exclude inferences into constrained variables, we cannot simulate every inference into constants anymore. Another difference between Schematic Saturation and meta-saturation is the *Schematic Deletion* rule. This rule makes Schematic Saturation terminate on examples where meta-saturation diverges (see e.g., Example 4).

In [16], the authors define the notion of variable-active clause so that if, for a given theory T axiomatized by the set $Ax(T)$ of clauses, any saturation of $Ax(T) \cup S$ halts and does not infer any variable-active clauses, then T is stably infinite. Notice that if for an arbitrary set of ground flat literals S, any saturation of $Ax(T) \cup S$ does not derive any variable-active clauses, then the theory T is variable-inactive, in the sense of [1]. The work in [16] goes two steps beyond that in [1]. First, it shows that the absence of variable-active clauses in any saturation implies stable infiniteness. Second, it provides an

automatic check of the absence of variable-active clauses by meta-saturation. However, this condition is rather too strong a requirement for combination as there exist theories for which paramodulation derives variable-active clauses which are stably infinite as illustrated by Example 7. The results of this paper are more general in this respect, as Schematic Saturation (initially designed in [19]) can automatically check stable infiniteness without relying on the variable-activity condition of [16]. A further generalization is that Schematic Saturation does not need to assume anymore that a negative selection function is used by paramodulation as it was the case in [16]. This is again a rather strong requirement, which narrows the scope of applicability of the method in [16] because there exist theories for which Schematic Saturation halts with one special ordering but not with others. Let us illustrate the problem on an example.

Example 11. The theory presented by the clause

$$Nat(X) \neq True \vee Nat\big(s(X)\big) = True \tag{N1}$$

is stably infinite. However if we consider negative selection, then the clause $Nat(x) = y \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top}$ will paramodulate with (N1) to generate the clause $y \neq True \vee Nat(s(x)) = True \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top}$. *Reflection* applies to the last clause yielding $Nat(s(x)) = True \parallel x \preccurlyeq c^{\top}$. The new clause, in turn, will paramodulate with (N1) again and Schematic Paramodulation will not halt.

As an alternative, we may consider an ordered selection function *sel* such that

$$sel\big(Nat(X) \neq True \vee Nat\big(s(X)\big) = True\big) = \big\{Nat\big(s(X)\big) = True\big\}.$$

It is easy to see that Schematic Saturation will contain (N1), the (finite) set of constrained unit literals G_0^T schematizing an arbitrary set of ground flat literals, and the following set of clauses:

$$Nat(x) \neq True \parallel x \preccurlyeq c^{\top}$$
$$Nat\big(s(x)\big) = True \parallel x \preccurlyeq c^{\top}$$
$$Nat(y) = True \parallel x \preccurlyeq c^{\top} \wedge y \preccurlyeq c^{\top}$$

Hence, the saturated set of clauses is finite and does not contain the trivial equality $X = Y$. By Theorem 7, we are entitled to conclude that the theory axiomatized by (N1) is stably infinite. □

This example illustrates that Schematic Saturation only requires fairness on the computation of saturated sets of clauses and it makes no assumptions on the selection function. Therefore, it is likely to handle more theories than the method in [16].

6. Conclusions and future work

We have introduced Schematic Saturation as a means to over-approximate the inferences that paramodulation can generate while solving the satisfiability problem for a certain theory T, i.e. computing the set of persistent clauses deriving from the union of the set of axioms of T and an arbitrary set of ground flat literals. Schematic Saturation is the key ingredient to design procedures capable of answering the following questions about T. (a) Is T decidable? (b) Is T stably infinite? (c) Is T deduction complete? Also, given two theories T_1 and T_2, (d) can paramodulation decide the satisfiability problem in the union of T_1 and T_2, when it can decide the satisfiability of T_1 and T_2 separately? Being able to answer questions (b) and (c) enable us to combine paramodulation-based procedures with black-box procedures for theories not amenable to the rewriting-approach, such as the quantifier-free fragment of Linear Arithmetic, by the Nelson–Oppen combination schema.

There are two main lines of research for future work. First, we intend to investigate further applications of Schematic Saturation. For example, in [31,26], new combination schemas for non-stably infinite theories are described. It would be interesting to identify sufficient conditions for the correctness of the combination methods in [31,26], which can be checked by Schematic Saturation. Second, we want to find further ways to handle fragments of Presburger Arithmetic in the context of paramodulation. In this respect, it seems promising to design an extension of Schematic Saturation in order to simulate saturations modulo Abelian groups [11,34,29,14].

Acknowledgments

The authors would like to thank the referee for his comments and suggestions that helped to improve the clarity of the paper.

References

[1] A. Armando, M.P. Bonacina, S. Ranise, S. Schulz, On a rewriting approach to satisfiability procedures: extension, combination of theories and an experimental appraisal, in: B. Gramlich (Ed.), Proc. of the 5th Int. Workshop on Frontiers of Combining Systems (FroCoS'2005), in: Lecture Notes in Computer Science, vol. 3717, Springer, Vienna, Austria, September 2005, pp. 65–80.

[2] A. Armando, S. Ranise, M. Rusinowitch, A rewriting approach to satisfiability procedures, J. Inform. Comput. 183 (2) (June 2003) 140–164.
[3] T. Ball, S.K. Rajamani, The SLAM project: Debugging system software via static analysis, in: Principle of Programming Languages (POPL'02), 2002, pp. 1–3.
[4] D.A. Basin, H. Ganzinger, Automated complexity analysis based on ordered resolution, J. ACM 48 (1) (2001) 70–109.
[5] M.P. Bonacina, S. Ghilardi, E. Nicolini, S. Ranise, D. Zucchelli, Decidability and undecidability results for Nelson–Oppen and rewrite-based decision procedures, in: Proc. of the 3rd Int. Conference on Automated Reasoning (IJCAR'06), Seattle, WA, USA, in: Lecture Notes in Artificial Intelligence, vol. 4130, Springer, August 2006, pp. 513–527.
[6] C. Bouillaguet, V. Kuncak, T. Wies, K. Zee, M. Rinard, Using first-order theorem provers in the Jahob data structure verification system, in: Verification, Model Checking and Abstract Interpretation (VMCAI'07), in: Lecture Notes in Computer Science, vol. 4349, Springer, 2007, pp. 74–88.
[7] N. Dershowitz, J.-P. Jouannaud, Handbook of Theoretical Computer Science, vol. B, Rewrite Systems, Elsevier and MIT Press, 1990, pp. 244–320 (Chapter 6).
[8] D. Detlefs, G. Nelson, J.B. Saxe, Simplify: a theorem prover for program checking, J. ACM 52 (3) (2005) 365–473.
[9] J.-C. Filliâtre, C. Marché, The why/krakatoa/caduceus platform for deductive program verification, in: Computer Aided Verification, 19th International Conference, Proceedings, CAV 2007, Berlin, Germany, in: Lecture Notes in Computer Science, vol. 4590, Springer, 2007, pp. 173–177.
[10] H. Ganzinger, D.A. McAllester, Logical algorithms, in: 18th International Conference on Logic Programming (ICLP'02), in: Lecture Notes in Computer Science, vol. 2401, Springer, 2002, pp. 209–223.
[11] H. Ganzinger, U. Waldmann, Theorem proving in cancellative abelian monoids, in: M.A. McRobbie, J.K. Slaney (Eds.), Automated Deduction – CADE-13, 13th International Conference on Automated Deduction, Proceedings, New Brunswick, NJ, USA, 1996, in: Lecture Notes in Computer Science, vol. 1104, Springer, 1996, pp. 388–402.
[12] Y. Ge, C. Barrett, C. Tinelli, Solving quantified verification conditions using satisfiability modulo theories, in: Proc. of the 21st Int. Conference on Automated Deduction (CADE'07), in: Lecture Notes in Artificial Intelligence, vol. 4603, Springer, Bremen, Germany, July 2007, pp. 167–182.
[13] R. Givan, D.A. McAllester, Polynomial-time computation via local inference relations, ACM Trans. Comput. Log. 3 (4) (2002) 521–541.
[14] G. Godoy, R. Nieuwenhuis, Superposition with completely built-in abelian groups, J. Symbolic Comput. 37 (1) (2004) 1–33.
[15] H. Kirchner, S. Ranise, C. Ringeissen, D.-K. Tran, On superposition-based satisfiability procedures and their combination, in: Proc. of Second International Colloquium on Theoretical Aspects of Computing (ICTAC'05), Hanoi, Vietnam, in: Lecture Notes in Computer Science, vol. 3722, Springer, Oct. 2005, pp. 594–608.
[16] H. Kirchner, S. Ranise, C. Ringeissen, D.-K. Tran, Automatic combinability of rewriting-based satisfiability procedures, in: Proc. of the 13th Int. Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'06), in: Lecture Notes in Artificial Intelligence, vol. 4246, Springer, Phnom Penh, Cambodia, November 2006, pp. 542–556 (subseries of Lecture Notes in Computer Science).
[17] H.R. Lewis, Renaming a set of clauses as a horn set, J. ACM 25 (1978) 134–135.
[18] C. Lynch, B. Morawska, Automatic decidability, in: Proc. of 17th IEEE Symposium on Logic in Computer Science (LICS'02), Copenhagen, Denmark, July 22–25, IEEE Computer Society Press, 2002, pp. 7–16.
[19] C. Lynch, D.-K. Tran, Automatic decidability and combinability revisited, in: Proc. of the 21st Int. Conference on Automated Deduction (CADE'07), in: Lecture Notes in Artificial Intelligence, vol. 4603, Springer, Bremen, Germany, July 2007, pp. 328–344.
[20] D.A. McAllester, Automatic recognition of tractability in inference relations, J. ACM 40 (2) (1993) 284–303.
[21] G. Nelson, Techniques for program verification, Technical Report CS-81-10, Xerox Palo Research Center California, USA, 1981.
[22] G. Nelson, D.C. Oppen, Simplification by cooperating decision procedures, ACM Trans. Program. Lang. Syst. 1 (2) (Oct. 1979) 245–257.
[23] R. Nieuwenhuis, A. Rubio, Paramodulation-based theorem proving, in: A. Robinson, A. Voronkov (Eds.), Handbook of Automated Reasoning, Elsevier and MIT Press, 2001, pp. 371–443.
[24] D.C. Oppen, Reasoning about recursively defined data structures, J. ACM 27 (3) (1980) 403–411.
[25] S. Ranise, C. Ringeissen, D.-K. Tran, Nelson–Oppen, Shostak and the extended canonizer: A family picture with a newborn, in: Proc. of First International Colloquium on Theoretical Aspects of Computing (ICTAC'04), Guiyang, China, in: Lecture Notes in Computer Science, vol. 3407, Springer, Sep. 2004, pp. 372–386.
[26] S. Ranise, C. Ringeissen, C.G. Zarba, Combining data structures with nonstably infinite theories using many-sorted logic, in: B. Gramlich (Ed.), Proc. of the 5th Int. Workshop on Frontiers of Combining Systems (FroCoS'05), in: Lecture Notes in Computer Science, vol. 3717, Springer, Vienna, Austria, September 2005, pp. 48–64.
[27] A. Riazanov, A. Voronkov, Splitting without backtracking, in: Proc. of 7th International Joint Conference on Artificial Intelligence (IJCAI'01), Seattle, Washington, USA, August 4–10, 2001, pp. 611–617.
[28] J.M. Schumann, Automated Theorem Proving in Software Engineering, Springer, 2001.
[29] J. Stuber, Superposition theorem proving for abelian groups represented as integer modules, Theoret. Comput. Sci. 208 (1–2) (1998) 149–177.
[30] C. Tinelli, M.T. Harandi, A new correctness proof of the Nelson–Oppen combination procedure, in: F. Baader, K.U. Schulz (Eds.), Frontiers of Combining Systems: Proceedings of the 1st International Workshop (FroCos'96), Munich, Germany, in: Applied Logic, Kluwer Academic Publishers, Mar. 1996, pp. 103–120.
[31] C. Tinelli, C.G. Zarba, Combining nonstably infinite theories, J. Automat. Reason. 34 (3) (Apr. 2005) 209–238.
[32] D.-K. Tran, Conception de Procédure de Décision par Combinaison et Saturation, PhD thesis, LORIA – Université Henri Poincaré, Nancy, France, 2007.
[33] D. van Dalen, Logic and Structure, second edition, Springer, 1989.
[34] U. Waldmann, Superposition for divisible torsion-free abelian groups, in: C. Kirchner, H. Kirchner (Eds.), Automated Deduction – CADE-15, 15th International Conference on Automated Deduction, Proceedings, Lindau, Germany, 1998, in: Lecture Notes in Computer Science, vol. 1421, Springer, 1998, pp. 144–159.