J Autom Reasoning (2013) 51:325–356
DOI 10.1007/s10817-012-9263-4

SMELS: Satisfiability Modulo Equality with Lazy Superposition

Christopher Lynch · Quang-Trung Ta · Duc-Khanh Tran

Received: 5 December 2010 / Accepted: 3 October 2012 / Published online: 18 October 2012

Abstract We consider the problem of checking satisfiability of quantified formulae in First Order Logic with Equality. We propose a new procedure for combining SAT solvers with Superposition Theorem Provers to handle quantified formulae in an efficient and complete way. In our procedure, the input formula is converted into CNF as in traditional first order logic theorem provers. The ground clauses are given to the SAT solver, which runs a DPLL method to build partial models. The partial model is reduced, and then passed to a Superposition procedure, along with justifications of literals. The Superposition procedure then performs an inference rule, which we call Justified Superposition, between the ground literals and the nonground clauses, plus usual Superposition rules with the nonground clauses. Any resulting ground clauses are provided to the DPLL engine. We prove the completeness of our procedure, using a nontrivial modification of the Bachmair and Ganzinger's model generation technique. We have implemented a theorem prover based on this idea by reusing state-of-the-art SAT solver and Superposition Theorem Prover. Our theorem prover inherits the best of both worlds: a SAT solver to handle ground clauses efficiently, and a Superposition theorem prover which uses powerful orderings to handle the nonground clauses. Experimental results are promising, and hereby confirm the viability of our method.

C. Lynch
Department of Mathematics and Computer Science, Clarkson University,
P.O. Box 5815, Potsdam, NY 13699-5815, USA
e-mail: clynch@clarkson.edu

Q.-T. Ta
School of Computing, National University of Singapore, 13 Computing Drive,
Singapore 117417, Singapore
e-mail: taqt@comp.nus.edu.sg

D.-K. Tran (✉)
School of Information and Communication Technology, Hanoi University of Science and Technology, 1 Dai Co Viet, Hanoi, Vietnam
e-mail: khanhtd@soict.hut.edu.vn

Keywords Theorem proving · SAT · Superposition

1 Introduction

Given the outstanding efficiency of DPLL-based SAT solvers [7, 8], substantial research has centered around ways to utilize SAT solvers for first order logic theorem proving. Toward this pursuit, some have lifted the DPLL procedure to first order logic in the form of the Model Evolution calculus [5]. Others have used SAT solvers as auxiliary tools to determine the satisfiability of certain fragments of first order logic via propositional encodings [13, 18, 28]. A third use of SAT solvers in first order logic theorem proving has been in saturation-based instance generation methods which repeatedly call upon a SAT solver to find so-called conflicts between clauses and use instance generation inferences to resolve these conflicts [15, 17].

Another well known line of research in first order logic theorem proving began with Robinson's landmark paper [26] which describes his Resolution principle and unification. Since then, many refinements have been made, e.g. Ordered Resolution and Semantic Resolution [1, 16, 18]. Recent implementations of Resolution have been shown to be the top performer in the CASC competition [29]. A key benefit of Resolution is the ability to restrict the search space with the use of Ordered Resolution. Here, Resolution inferences are only necessary on maximal literals. Selection rules can also be applied which also restrict the search space. Ordered Resolution can be efficient in practice, because it tends to produce literals in the conclusion of an inference that are smaller than in the premises. Ordered Resolution is known to be complete for first order logic (see e.g., [2]). Superposition [24] is an improvement of Resolution to deal efficiently with the equality predicate. An important tool in Superposition is the use of term orderings for restricting the number of inferences. Superposition is complete for first order logic with equality (see e.g., [24]). There exist mature Automated Theorem Provers (ATPs), SPASS [30], Vampire [25], E [27] to name a few, implementing Resolution and Superposition. However, Resolution and Superposition ATPs are believed to not be as fast as SAT solvers on propositional problems having complex boolean structures.

Deciding the satisfiability of a formula with respect to a background first order theory is being recognized as crucial for many computer science problems including verification, scheduling, optimization, etc. There exist specialized reasoning methods for many background theories of interest, such as lists, arrays, records, integer-offsets, and linear arithmetic, etc., which go under the name of Satisfiability Modulo Theories (SMT) solvers, but they used to be limited to the particular class of first order formulae without quantifiers. Finding good heuristics for lifting SMT techniques from ground to nonground formulae is a hot line of current and future research. For instance [4, 10] use heuristics based on the instantiation method of the theorem prover Simplify [14]. However those heuristics are incomplete, i.e. they may fail to prove unsatisfiability of formulae. Another problem is that instantiation-based heuristics usually diverge or are forced to terminate with unknown result when dealing with satisfiable formulae. This is so because they instantiate universally quantified variables, but the set of instantiating terms is infinite most of the time.

In this paper we are concerned with the problem of checking satisfiability of nonground formulae in First Order Logic with Equality. We propose a novel method,

that we call SMELS—Satisfiability Modulo Equality with Lazy Superposition for the satisfiability problem of nonground formulae in an efficient and complete way. Our method combines the best of the two worlds SAT solvers and ATPs: efficiency for ground problems; and completeness for nonground problems.

In SMELS, the input formula is first converted into CNF as in traditional first order logic theorem provers. Then the set of clauses is partitioned into two subsets: a set of ground clauses and a set of nonground clauses. Then we run a DPLL solver to build a partial model, i.e. a set of ground literals, along with justifications of elements in the model. The idea of justification is a generalization of what is done in clause learning in SAT solvers. The partial model is then reduced by the equational theory, and justifications of elements in the reduced partial model are also calculated. The reduced partial model is next checked for consistency along with the nonground clauses using a Superposition procedure. To ensure soundness of SMELS, the Superposition procedure has an inference rule called Justified Superposition, involving a literal from the reduced partial model and a nonground clause, taking into account the justification of the literal. In addition, the procedure also has usual Superposition inferences among nonground clauses. Any ground clauses resulting from the Superposition procedure are provided to the DPLL solver. We do not perform any Superposition inferences among ground clauses because they are already reduced.

We prove the completeness of SMELS, using a nontrivial modification of Bachmair and Ganzinger's model generation technique. In particular, completeness of SMELS implies that one of the following three possibilities will happen when applying our calculus: (i) the original set of clauses is satisfiable, and after a finite number of steps the process will halt, giving a ground model modulo the background theory; or (ii) the original set of clauses is satisfiable, and in the limit, there is a set of clauses for which we can build a model; or (iii) the original set of clauses is unsatisfiable, and after a finite number of steps the process will halt with an unsatisfiable set of ground clauses.

We implement SMELS based on the aforementioned ideas. We reuse MiniSat 2.2.0 (cf. http://minisat.se) as the SAT Solver and SPASS 3.7 (cf. http://www.spass-prover.org) as the Resolution Theorem Prover, since they are both among the best in their respective categories, and they are still being maintained and improved. We experiment SMELS with the TPTP v4.0.1 benchmark. Overall, SMELS performs fairly well in solving problems. The performance in successful cases is acceptable compared to other theorem provers such as SPASS, iProver and Darwin. For some a subclass of problems, SMELS succeeds in a short time while SPASS, iProver and Darwin time out.

1.1 Related Work

Notable in using SAT solvers for theorem proving are the works by Lee and Plaisted [21] who proposes using Davis–Putnam procedure to speed up first order theorem proving, Hooker (et al.) who developed the first complete Partial Instantiation method for the full first order logic called Primal PI [17] and Ganzinger and Korovin who among many other contributions formalized and proved the completeness of the instance generation inference rule, Inst-Gen, and the extension SInst-Gen (Inst-Gen with semantic selection and hyper inferences) in [15]. The efficiency of

saturation-based instance generation methods is demonstrated by Korovin's implementation called iProver [19]. Instantiation theorem proving based on [15] also uses a SAT solver at the bottom to handle ground clauses resulting from instantiations. However neither Inst-Gen nor SInst-Gen of [15] uses an ordering to limit the search space. Instead, SInst-Gen utilizes semantic selection of clauses to be used for instantiations. Section 6 provides a comparative analysis of SMELS and iProver.

The Model Evolution calculus [5] provides another theorem proving method based on model finding. It is a lifted version of the DPLL method to first order logic. From the theoretical point of view, it is not easy to compare Resolution theorem proving with theorem proving based on the Model Evolution calculus. On some problems Resolution methods are better, and on some others Model Evolution methods are better. On satisfiable nonground problems, we suspect that methods like [5] perform better as they are designed to find models. In Section 6, we give the result of comparative experimentation between SMELS and Darwin.

In [22], the authors give a new inference system for first order logic, SIG- Res, which combines together SInst-Gen [15] and Ordered Resolution [2] into a single inference system. Given a set F of first order clauses two sets of clauses P and R are created such that $P \cup R = F$. Under SIG-Res, P is saturated by SInst-Gen, and Resolution is applied to pairs of clauses in F where at least one of the clauses is in R.

Resolution and Superposition theorem provers like SPASS [30] and Vampire [25] use splitting to improve efficiency. Vampire uses explicit propositional symbols to keep track of splitting while splitting in SPASS relies on labels to keep track of the split levels. Since SMELS does not perform any ground inferences in the Superposition procedure but delegates them to an efficient SAT solver along with a simplification engine instead, we believe that it can be better than existing Resolution and Superposition theorem provers on large problems containing mostly ground clauses. If the clauses are mostly nonground, traditional methods would probably work better. We compare the performance of SMELS and SPASS in Section 6.

SMT solvers such as [4, 10], are quite effective and efficient on certain classes of problems. SMT solvers are incomplete on nonground problems, unlike SMELOn, S., satisfiable nonground formulae, SMELS may terminate with the result satisfiable while SMT solvers diverge or are forced to halt with the result unknown. However as soon as efficiency is the matter of consideration, SMT solvers outperform almost every other tools.

SMELS exhibits some similarities with the DPLL(Γ) calculus of de Moura and Bjørner [9]. Basically the DPLL(Γ) calculus combines the DPLL(T) rules of abstract DPLL modulo theories and some other rules which serve as the interface for Superposition rules. In DPLL(Γ) ground equational reasoning is again handled by Superposition rules. In practice DPLL(Γ) can be enhanced by integrating a Congruence Closure algorithm to reduce literals before resorting to Superposition rules, as in SMELNotice, S., that the authors of the DPLL(Γ) calculus have not proven the completeness of the enhanced version. In this respect, SMELS can be viewed as an optimized strategy of DPLL(Γ), which is complete for first order logic with equality.

In [6], the authors propose an approach that decomposes the formula in such a way that its definitional part, including the theory, can be compiled by a Superposition theorem prover, and the residual problem can be decided by an SMT-solver. The resulting decision by stages mechanism may inherit the complementary strengths

of first order provers and SMT-solvers, as in SMELHowever, S., the decision mechanism of [6] can be classified as an eager method while SMELS is rather a lazy one. The compilation phase of [6] must be done once and for all until a weaker form of saturation is reached. It is important to mention that this method does not apply to all finitely definitional first order theories, meaning that the procedure is not always complete. To the best of our knowledge, the method of [6] has not been implemented and experimented with so it is difficult to compare it to SMELS.

The theorem prover haRVey [11] combines a Boolean solver (SAT or BDD) with the equation theorem prover E [27]. The integration is loose, compared to SMELS, because resulting non-unit ground clauses are handled by E, and not by the simplification engine like in SMELS; therefore inferences among ground clauses are allowed, in contrast with SMELS.

1.2 Structure of the Paper

The paper is structured as follows. Section 2 introduces some background notions. Section 3 presents a complete inference system, called SLR, for first order logic. We give this inference system to illustrate some ideas in a simpler setting, and to relate our work to some previous work. Section 4 presents SMELS, which is a complete calculus for first order logic with equality. In Section 5, we give the detailed completeness proof of SMELWe, S., do not prove the completeness of SLR as it is a special case of SMELS. Section 6 describes the implementation of SMELIn, S., Section 7 we report on experimental results of SMELS. Finally, Section 8 concludes and presents some future work.

2 Preliminaries

We assume the usual rewriting definitions such as term, position, substitution, unifier, ordering, etc..., as defined in [12]. Atoms are represented by symbols A, B, literals by L. An atom is also called a positive literal, and the negation of an atom is a negative literal. Equations will be written as $s = t$, disequations as $s \neq t$. The formula $s \bowtie t$ is either $s = t$ or $s \neq t$. Given a set of ground literals M, then M^+ (resp. M^-) represents the positive (resp. negative) literals in M. A clause is a disjunction of literals, thought of as a multiset. If L is A (resp. $\neg A$) then $\neg L$ is $\neg A$ (resp. A).

Given a set of clauses S, let $g(S)$ (resp. $v(S)$) be the set of ground (resp. nonground) clauses in S. Define $Gr(S)$ as the set of all ground instances of S. For a clause C, let $g(C)$ (resp. $v(C)$) be the multiset of ground (resp. nonground) literals in C. Let $GL(S)$ be the set of all ground literals in S.

We will be working on ground instances of clauses, called *closures* [3]. If C is a clause and θ is a ground substitution, we write $C \cdot \theta$ to indicate the θ instance of C. A closure represents a ground instance, but makes it clear which is the original clause and which is the grounding substitution. The clause C is the *skeleton* of the closure, and θ is the *substitution*. When it is convenient, we will treat the closure $C \cdot \theta$ as the instantiated clause $C\theta$.

If $\prec$ is a (strict) ordering then its associated quasiordering is denoted with $\preceq$, and inversely if $\preceq$ is a quasiordering then $\prec$ denotes its associated ordering. We define a quasiordering $\preceq_g$ on closures, so that for clauses C and D and substitutions σ and θ,

$C \cdot \sigma \preceq_g D \cdot \theta$ if (1) C and D are both ground, or (2) C and D are both nonground, or (3) C is ground and D is nonground. Define $\prec_r$ to be a ground total reduction ordering on terms. The ordering $\prec_r$ is extended to equations by considering them as multisets of terms, extended to disequations in such a way that a disequation $s \neq t$ is larger than an equation $s = u$ if $s \succeq_r t$ and $s \succeq_r u$, and then extended to clauses by considering them as multisets of literals. The ordering $\prec_r$ is extended to closures so that $C \cdot \sigma \prec_r D \cdot \theta$ if $C\sigma \prec_r D\theta$. Next we define an ordering $\prec_i$, called an *instance ordering*, defined on closures to be the lexicographic combination of $\prec_g$ and $\prec_r$. So, to compare two closures in the instance ordering, first check if one skeleton is ground and the other is nonground, otherwise apply the substitution and compare using the reduction ordering.

An interpretation M is defined as a set of ground equations. For an interpretation M and an equation $s = t$, we write $M \models s = t$ if $s = t$ is a logical consequence of M. We write $M \models s \neq t$ if $M \not\models s = t$. Given an interpretation M and a ground clause C, $M \models C$ if and only if $M \models L$ for some L in C. Given a set of ground clauses S, an interpretation M is a *model of* S if $M \models C$ for all C in S. If T is a set of literals, we say that T is *satisfiable* if T has a model.

For an interpretation M and a literal L, let $M^L = \{L' \in M \mid L' \preceq_r L\}$. We write $M \models_{\preceq_r} L$ if $M^L \models L$. Let E be a set of equations ordered wrt. $\preceq_r$ and $s = t$ be an equation. We say that $s = t$ has a rewrite proof using E if s and t have the same normal forms with respect to E, and we will write $E \vdash s = t$. For an interpretation M and an equation $s = t$, we write $M \vdash s = t$ if there is a set of equations $E \subseteq M$, ordered wrt. $\preceq_r$, such that $E \vdash s = t$.

For a clause C, $M \models_{\preceq_r} C$ (resp. $M \vdash C$) if $M \models_{\preceq_r} L$ (resp. $M \vdash L$) for some $L \in C$. For a set of ground clauses S, we write $M \models_{\preceq_r} S$ (resp. $M \vdash S$) if $M \models_{\preceq_r} C$ (resp. $M \vdash C$) for all $C \in S$. A set of literals T is *consistent with respect to* $\models_{\preceq_r}$ if there is no disequation $s \neq t \in T$ such that $T \models_{\preceq_r} s = t$, similarly T is *consistent with respect to* $\vdash$ if there is no disequation $s \neq t \in T$ such that $T \vdash s = t$. For a reduction ordering $\prec_r$, a given interpretation M and an equation $s = t$, $M \models s = t$ and $M \vdash s = t$ and $M \models_{\preceq_r} s = t$ are not equivalent in general. For example, consider $M = \{a = b, a = c\}$. If $\preceq_r$ is an ordering such that $b \preceq_r a$ and $c \preceq_r a$, then $M \models b = c$ but $M \not\models_{\preceq_r} b = c$ and $M \not\vdash b = c$.

3 Resolution Inference System

We give a Resolution inference system for first order logic without equality. This can just be viewed as a special case of the Superposition inference system that will be given later. However, we give this inference system to illustrate some ideas in a simpler setting, and to relate our work to some previous work.

In this paper, we will represent a *truth assignment* GM as the set of ground literals made true by GM. GM is a *satisfying truth assignment* for a set of clauses if GM makes all its clauses true. We assume that we have a set of clauses S, where a SAT procedure has been run on $g(S)$ to produce a satisfying truth assignment GM of $g(S)$. SAT procedures also construct justification functions from which lemmas are constructed. Justification functions are formally defined as follows.

Definition 1 Given a set of ground clauses S, we define $cons(S)$ as the set of ground literals implied by S. Let GM be a truth assignment. Let G be a set of ground clauses. A function

$$j : cons(GM) \to \mathcal{P}(GM)$$

is called an (GM, G)-*justification function* if

$$L \in cons(G \cup j(L))$$

for all L in $cons(GM)$.

If $j(L) = \{L\}$, then L is said to be *self-justified*. If all literals in $cons(GM)$ are self-justified, then j is a *self-justification function*.

Let us briefly explain the relevance of justification functions. Our aim is to check consistency of the truth assignment GM along with the set of nonground clauses $v(S)$. However, the literals in GM may not be consequences of S, and anything derived from GM and $v(S)$ using Resolution inferences may not be a consequence of S. Our solution to this problem is to consider the justifications of the literals in GM, that is for a given literal $L \in GM$, $\neg j(L) \vee L$ is a consequence of the input set of clauses S. Given a set of ground clauses G and a truth assignment GM there is always a self-justification function, since GM is always a consequence of GM.

Example 1 Let $G = \{p, \neg p \vee q, r \vee \neg s, \neg q \vee s \vee \neg t\}$. Then $GM = \{p, q, \neg s, \neg t\}$ is a satisfying truth assignment of G. Let j_1 be the function such that $j_1(p) = \emptyset$, $j_1(q) = \emptyset$, $j_1(\neg s) = \{\neg s\}$, and $j_1(\neg t) = \{\neg s\}$. Then j_1 is a justification function. But there are many justification functions. For example, let j_2 be identical to j_1 except that $j_2(\neg t) = \{p, \neg s\}$. Then j_2 is also a justification function. There is also the self-justification function j_3 such that $j_3(L) = \{L\}$ for all L in GM.

The Resolution inference system makes use of an instance ordering $\prec_i$ defined on closures of predicative clauses, which is the lexicographic combination of $\prec_g$ and $\prec_r$ defined on predicative clauses, as in Section 2 for equational clauses.

We need the following notion in the Resolution inference system.

Definition 2 Let C be a clause containing a literal L, and let σ be a substitution, we say that L is σ-*var-maximal* in C if $L \in v(C)$ and there is no literal $L' \in C$ such that $L' \cdot \sigma \succ_i L \cdot \sigma$.

For completeness of our Resolution inference system, we need the following assumption.

Assumption 1 *Each satisfying truth assignment GM of $g(S)$ has been extended to the atoms of $GL(S)$ in any way such that it is defined on all literals of $GL(S)$.*

The $(GM, g(S))$-justification function j must be extended along with the extension of the model.

A simple way to extend the justification function is to make all the additional literals self-justified.

We call our Resolution inference system SLR, Satisfiability with Lazy Resolution. It consists of the inference rules in Fig. 1. The Nonground Resolution and Factoring inference rules differ from the usual Ordered Resolution and Factoring inference rules in two ways: they are only allowed on nonground clauses; and they are not performed on the maximal literals of the premises, but instead on the maximal nonground literals of the premises. The Justified Resolution rule involves one nonground premise C with maximal nonground literal L. It produces a new clause D, where a ground instance of L is replaced by its justification. This is similar to the process in SAT procedures where a lemma is created. The Justified Resolution inference rule could be viewed as a Resolution inference between C and a new clause $C' = L\sigma \vee \neg j(L\sigma)$. However, C' does not need to be explicitly created. By definition of justification function, C' is implied by $g(S)$.

In the case where the literal L is self-justified, we call the inference *Self-justified Resolution*. This corresponds to an inference with $L\sigma \vee \neg L\sigma$, a tautology. The inference then effectively just applies σ to its premise. This is a proper instantiation, because L must be nonground since it is in $v(C)$. Therefore, SLR with a self-justification function can be viewed as a combination of an Instantiation-based inference system such as InstGen [15] with Ordered Resolution, the first such combination we are aware of.

In the inference rules, we have not considered selection functions. Our completeness results can be adapted to deal with selection rules, but we choose to present just the ordered case to make the presentation as simple as possible.

Let us take an example to illustrate how SLR works.

Example 2 Let S_0 contain the following clauses:

$$p$$
$$\neg p \vee q$$
$$r(a) \vee \neg s$$
$$\neg q \vee s \vee \neg t(b)$$
$$\neg r(x) \vee \neg s$$
$$t(y)$$

Nonground Resolution	$\dfrac{\Gamma \vee A \quad \Delta \vee \neg B}{(\Gamma \vee \Delta)\sigma}$	if $\sigma = mgu(A, B)$, $A\sigma \in max(v(\Gamma \vee A)\sigma)$, and $\neg B\sigma \in max(v(\Delta \vee \neg B)\sigma)$
Factoring	$\dfrac{\Gamma \vee A \vee B}{(\Gamma \vee A)\sigma}$	if $\sigma = mgu(A, B)$ and $A\sigma \in max(v(\Gamma \vee A \vee B)\sigma)$
Justified Resolution	$\dfrac{\Gamma \vee \neg L}{\Gamma\sigma \vee \neg j(L\sigma)}$	if $L\sigma \in GM$, and $L\sigma \in max(v(\Gamma \vee L)\sigma))$

where all premises are from $v(S)$, GM is a satisfying truth assignment of $g(S)$ which is defined on $GL(S)$, and j is a $(GM, g(S))$ justification function.

Fig. 1 Inference rules of SLR

Table 1 SLR execution

i	$g(S_i)$	GM_i	$v(S_i)$	$g(S_{i+1}) \setminus g(S_i)$
0	p	$p\{\}$	$\neg r(x) \vee \neg s$	s
	$\neg p \vee q$	$q\{\}$	$t(y)$	
	$r(a) \vee \neg s$	$\neg s\{\neg s\}$		
	$\neg q \vee s \vee \neg t(b)$	$\neg t(b)\{\neg s\}$		
1	p	$p\{\}$	$\neg r(x) \vee \neg s$	$\neg s$
	$\neg p \vee q$	$q\{\}$	$t(y)$	
	$r(a) \vee \neg s$	$s\{\}$		
	$\neg q \vee s \vee \neg t(b)$	$r(a)\{\}$		
2	p		$\neg r(x) \vee \neg s$	
	$\neg p \vee q$		$t(y)$	
	$r(a) \vee \neg s$			
	$\neg q \vee s \vee \neg t(b)$			
	s			
	$\neg s$			

Table 1 recapitulates the execution of SLR on S_0. By running a DPLL procedure $g(S_0)$, we obtain a model GM_0. The Resolution procedure is next applied on $GM_0 \cup v(S_0)$. We have the following *Justified Resolution* inference

$$\frac{t(y)}{s}$$

The new set of clauses obtained is noted S_1. The new ground clauses resulting from the Resolution procedure are those in the line 0 and the column $g(S_{i+1}) \setminus g(S_i)$ of Table 1. Again, we run a DPLL procedure on $g(S_1)$, then the Resolution procedure is applied on $GM_1 \cup v(S_1)$. We have the following *Justified Resolution* inference

$$\frac{\neg r(x) \vee \neg s}{\neg s}$$

The new set of clauses obtained is noted S_2. The new ground clauses resulting from the Resolution procedure are in the line 1 and the column $g(S_{i+1}) \setminus g(S_i)$. Then DPLL outputs unsatisfiable running on the new set of clauses $g(S_2)$ because it contains both s and $\neg s$.

4 Superposition Inference System

Now we extend our inference system to first order logic with equality.

Definition 3 A set M of ground equations and disequations is called *reduced* if M never contains a literal of the form $L[s]$ along with another literal of the form $s = t$, where $s \succ_r t$.

A set M of ground equations and disequations M is called *left-reduced* if there does not exist $u[s] \bowtie v$ and $s = t$ in M with $u[s] \succ_r v$ and $s \succ_r t$.

If M is left-reduced then M^+ is a convergent rewrite system.

Again, we assume a set of clauses S, a satisfying truth assignment GM extended so that it is defined on all atoms of $GL(S)$, and a $(GM, g(S))$ justification function, as in Assumption 1. The only difference between here and the nonequational case is that we now assume in addition the following.

Assumption 2 *GM is reduced.*

The inference rules for SMELS, Superposition Modulo Equality with Lazy Superposition, are given in Fig. 2. The ideas are all the same as in the nonequational case. The inference rules are like the usual Superposition rules on nonground clauses, except that the ordering only involves the nonground literals. The Justified Superposition rules can be viewed as a Superposition between a nonground clause and an implicit ground clause.

Let S be a set of clauses. Let GM be a reduced satisfying truth assignment of $g(S)$, extended so that it is defined on $GL(S)$. Let j be a $(GM, g(S))$ justification function. A SMELS inference system is parameterized by GM and j. So we will refer to SMELS(GM, j) to indicate what the parameters are. In an actual implementation of this inference system when an inference rule adds a new ground clause, that clause will be added to $g(S)$, and a new satisfying truth assignment GM may be created. Therefore, the parameters of SMELS may change as inferences are performed.

A clause C is *redundant* with respect to a set of clauses S if C is properly subsumed by a clause in S or if the following conditions hold for every ground instance $C \cdot \theta$ of C:

1. there is a subset E of $Gr(S)$, where E is a set of equations; and
2. all members of E are smaller than $C \cdot \theta$ with respect to both orderings $\prec_i$ and $\prec_r$; and
3. there is a ground instance $D \cdot \sigma$ such that either $D \cdot \sigma$ is in $Gr(S)$ or $D \cdot \sigma$ is a tautology; and

Nonground Superposition	$\dfrac{\Gamma \vee u[s'] \bowtie v \quad \Delta \vee s = t}{(\Gamma \vee \Delta \vee u[t] \bowtie v)\sigma}$	(i), (ii), (iii), (iv)
Eq. Resolution	$\dfrac{\Gamma \vee s \neq s'}{\Gamma\sigma}$	(v)
Eq. Factoring	$\dfrac{\Gamma \vee s = t \vee s' = t'}{(\Gamma \vee t \neq t' \vee s = t')\sigma}$	(iii), (iv), (vi)
Justified Superposition Into	$\dfrac{\Gamma \vee u[s'] \bowtie v}{(\Gamma \vee \neg j(s = t) \vee u[t] \bowtie v)\sigma}$	(i), (ii), (iii)
Justified Superposition From	$\dfrac{\Delta \vee s = t}{(\Delta \vee \neg j(u[s'] \bowtie v) \vee u[t] \bowtie v)\sigma}$	(i), (iii), (iv)

where all premises are from $v(S)$, GM is a reduced satisfying truth assignment of $g(S)$ which is defined on $GL(S)$, j is a $(GM, g(S))$ justification function, s' is not a variable in Nonground Superposition and Superposition Into, $\sigma = mgu(s, s')$, $s = t \in GM$ in Justified Superposition Into, $u[s'] \bowtie v \in GM$ in Justified Superposition From, and

(i) $u[s']\sigma \not\preceq_r v\sigma$, ***(ii)*** $u[s'] \bowtie v$ is σ-var- maximal in its clause, ***(iii)*** $s\sigma \not\preceq_r t\sigma$, ***(iv)*** $s = t$ is σ-var-maximal in its clause, ***(v)*** $s \neq s'$ is σ-var-maximal in its clause, ***(vi)*** $(s = t)\sigma \not\preceq_r (s' = t')\sigma$ and $s'\sigma \not\preceq_r t'\sigma$.

Fig. 2 Inference rules of *SMELS*

4. $D \cdot \sigma$ is smaller than $C \cdot \theta$ with respect to both orderings $\prec_i$ and $\prec_r$; and
5. for every literal $L\sigma$ in $D\sigma$, there is a literal $L'\theta$ in $C\theta$ such that $L\sigma$ and $L'\theta$ are equivalent modulo E and $L' \cdot \theta$ is larger or equal to $L \cdot \sigma$ with respect to both orderings $\prec_i$ and $\prec_r$.

It is clear that our definition of redundant clauses applies to usual redundancy elimination techniques such as subsumption, tautology deletion. Our definition covers demodulation as well. In the definition, C would be the clause before it is demodulated, and $C \cdot \theta$ is a ground instance of C. D would be a ground instance after C is demodulated, and E would contain the equation used to demodulate. It remains to make sure that demodulation satisfies property 4 of the definition, that is $D \cdot \sigma$ is smaller than or equal to $C \cdot \theta$ with respect to both orderings $\prec_i$ and $\prec_r$. It is easy to see that $D \cdot \sigma$ is smaller than or equal to $C \cdot \theta$ with respect to the reduction ordering $\prec_r$. For the instance ordering $\prec_i$, we know that the equation used to demodulate could not introduce a new variable, otherwise it would not have a right side smaller than the left side, so that means that D is ground whenever C is. In other words, $D \cdot \sigma$ is smaller than or equal to $C \cdot \theta$ with respect to the instance ordering $\prec_i$.

An inference is said to be redundant w.r.t. S if one of its premises is redundant w.r.t. S, or its conclusion is redundant w.r.t. S, or its conclusion is already present in S. We say that S is *saturated with respect to SMELS* if all SMELS(GM, j) inferences are redundant w.r.t. S, for some satisfying truth assignment GM and justification function j.

We define a *SMELS derivation* as a sequence of triples of the form

$$(S_0, G_0, j_0), (S_1, G_1, j_1), \cdots$$

where each S_i is a set of clauses, each G_i is a truth assignment defined over $GL(S_i)$ such that $G_i \models g(S_i)$, and each j_i is a $(G_i, g(S_i))$ justification function. Furthermore, one of the following is true of each S_{i+1}

1. S_{i+1} is formed by adding the conclusion of a SMELS(GM, j_i) inference to S_i, or
2. S_{i+1} is formed by removing a redundant clause from S_i, or
3. S_{i+1} is formed by removing a ground clause C that is implied by $g(S_i) \setminus \{C\}$.

We have assumed the existence of one satisfying truth assignment of the set of ground clauses whenever it is satisfiable. This is trivial if the derivation is finite. If the derivation is infinite, then ground clauses are added infinitely. We need to ensure that there are some truth values for the ground literals that occur infinitely often together, so that we can assume the existence of a single satisfying truth assignment. This motivates the following definitions of persistence, and fairness for infinite derivations.

Given a SMELS derivation $(S_0, G_0, j_0), (S_1, G_1, j_1), \cdots$, we say that a clause C is *persistent* if $C \in \bigcup_{i \geq 0} \bigcap_{j \geq i} S_j$. This is the usual definition of persistent clauses. If L is a ground literal, and M is a set of ground literals, we say that the pair (L, M) is *persistent* if $\{(S_i, G_i, j_i) \mid L \in G_i, j_i(L) = M\}$ is infinite. This means that the ground literal occurs infinitely often in the derivation with the same justification.

A Nonground Superposition, Equality Resolution, or Equality Factoring inference is *persistent* in a SMELS derivation if its premises are persistent. A Justified Superposition Into derivation is *persistent* if its premise is persistent, and the pair $(s = t, j(s = t))$ is persistent. A Justified Superposition From derivation is *persistent* if its premise is persistent, and the pair $(u \bowtie v, j(u \bowtie v))$ is persistent.

Table 2 SMELS execution

i	$g(S_i)$	M_i	GM_i	$v(S_i)$	$g(S_{i+1}) \setminus g(S_i)$
0	$p(a,b) = p_1$ $p(c,d) = p_2$ $p(e,f) = p_3$ $p_1 = p_2 \vee p_1 = p_3$ $a \neq c$ $a \neq e$	$p(a,b) = p_1\{\}$ $p(c,d) = p_2\{\}$ $p(e,f) = p_3\{\}$ $p_1 = p_2\{p_1 = p_2\}$ $a \neq c\{\}$ $a \neq e\{\}$	$p(a,b) = p_2\{p_1 = p_2\}$ $p(c,d) = p_2\{\}$ $p(e,f) = p_3\{\}$ $p_1 = p_2\{p_1 = p_2\}$ $a \neq c\{\}$ $a \neq e\{\}$	$p(x_1,y_1) \neq p(x_2,y_2) \vee x_1 = x_2$	$p_1 \neq p_2 \vee a = c$ $p_1 \neq p_2 \vee p_2 \neq p_3 \vee a = e$
1	$p(a,b) = p_1$ $p(c,d) = p_2$ $p(e,f) = p_3$ $p_1 = p_2 \vee p_1 = p_3$ $a \neq c$ $a \neq e$ $p_1 \neq p_2 \vee a = c$ $p_1 \neq p_2 \vee p_2 \neq p_3 \vee a = e$	$p(a,b) = p_1\{\}$ $p(c,d) = p_2\{\}$ $p(e,f) = p_3\{\}$ $p_1 = p_3\{\}$ $a \neq c\{\}$ $a \neq e\{\}$ $p_1 \neq p_2\{\}$	$p(a,b) = p_2\{p_1 = p_2\}$ $p(c,d) = p_2\{\}$ $p(e,f) = p_3\{\}$ $p_1 = p_3\{\}$ $a \neq c\{\}$ $a \neq e\{\}$ $p_3 \neq p_2\{\}$	$p(x_1,y_1) \neq p(x_2,y_2) \vee x_1 = x_2$ $p_1 \neq p_2 \vee p_2 \neq p(x_2,y_2) \vee a = x_2$ $p_2 \neq p(x_2,y_2) \vee c = x_2$ $p_3 \neq p(x_2,y_2) \vee e = x_2$	$a = e$
2	$p(a,b) = p_1$ $p(c,d) = p_2$ $p(e,f) = p_3$ $p_1 = p_2 \vee p_1 = p_3$ $a \neq c$ $a \neq e$ $p_1 \neq p_2 \vee a = c$ $p_1 \neq p_2 \vee p_2 \neq p_3 \vee a = e$ $a = e$			$p(x_1,y_1) \neq p(x_2,y_2) \vee x_1 = x_2$ $p_1 \neq p_2 \vee p_2 \neq p(x_2,y_2) \vee a = x_2$ $p_2 \neq p(x_2,y_2) \vee c = x_2$ $p_3 \neq p(x_2,y_2) \vee e = x_2$ $p_3 \neq p(x_2,y_2) \vee a = x_2$	

A SMELS derivation is *fair* if for every persistent inference there is some $i \geq 0$ such that this inference is redundant w.r.t. S_i, and there exist an enumeration of all positive literals $A_1, A_2, \cdots$ and an n such that for all $m \geq n$, $A_i \in j_m(A_j)$ implies that $i < j$. In this definition, we call n a *justification stabilizer* of the derivation. This last condition ensures that we will not continually add new literals and use those literals to justify previous literals, which may create a non-well-founded chain that could destroy completeness.

Let us take an example to see how SMELS works.

Example 3 Let S_0 contain the following clauses:

$$\begin{aligned} p(a, b) &= p_1 \\ p(c, d) &= p_2 \\ p(e, f) &= p_3 \\ p_1 = p_2 \vee p_1 &= p_3 \\ a &\neq c \\ a &\neq e \\ p(x_1, y_1) \neq p(x_2, y_2) \vee x_1 &= x_2 \end{aligned}$$

By running a DPLL procedure $g(S_0)$, we obtain a model M_0. Suppose that the DPLL engine finds a model M_0, then the model M_0 is reduced to the model GM_0. Now the Superposition procedure is applied on $GM_0 \cup v(S_0)$. For instance, we have the following *Justified Superposition Into* inference

$$\frac{p(x_1, y_1) \neq p(x_2, y_2) \vee x_1 = x_2}{p_1 \neq p_2 \vee p_2 \neq p(x_2, y_2) \vee a = x_2}$$

where the equation $p(a, b) = p_2$ in GM_0 is used, and its justification is $p_1 = p_2$. After an exhaustive application of inference rules and redundancy deletion, we obtain a new set of clauses, noted S_1. The new ground clauses resulting from the Superposition procedure are those in the line 0 and the column $g(S_{i+1}) \setminus g(S_i)$ of Table 2. Again, we run the DPLL procedure and the reduction algorithm on $g(S_1)$, then the Superposition procedure is applied on $GM_1 \cup v(S_1)$. The new set of clauses obtained is noted S_2. The new ground clauses resulting from the Superposition procedure are in the line 1 and the column $g(S_{i+1}) \setminus g(S_i)$. Then DPLL outputs unsatisfiable running on the new set of clauses $g(S_2)$ because it contains both $a = e$ and $a \neq e$.

5 Completeness

We will show that if S is saturated with respect to SMELS and does not contain the empty clause then there is a model M of S with $GM^+ \subseteq M$. This shows that if the inference rules are applied fairly, then one of the following three things will happen.

1. The original set of clauses is satisfiable, and after a finite number of steps the process will halt with a set of clauses S and a satisfying truth assignment GM of $g(S)$ such that $GM^+ \cup v(S) \models S$.

2. The original set of clauses is satisfiable, and in the limit, there is a set of clauses S and a satisfying truth assignment GM of $g(S)$ such that $GM^+ \cup v(S) \models S$.
3. The original set of clauses is unsatisfiable, and after a finite number of steps the process will halt with a set of clauses S such that $g(S)$ is unsatisfiable.

The first item is the most interesting. Instantiation methods based on E-matching [14] do not have this property, because they instantiate universally quantified variables, and it is never known when it is safe to stop instantiating. Of course, our inference system is only useful if the first item will happen frequently, and we suspect that it will, because of the orderings. In this case, we can think of $v(S)$ as representing a theory, and then GM is actually a satisfying truth assignment of $g(S)$ modulo the theory $v(S)$. This is useful, because the entire model M cannot always be represented with a finite number of ground clauses. In the case of the second item above, we consider the limit of the saturation process. In this case, the satisfying truth assignment GM referred to is a limit of the satisfying truth assignments constructed during the saturation process.

For the completeness proof, we build a model as in Bachmair and Ganzinger's model construction process. However, our model is more complicated because of the satisfying truth assignment GM. We construct a model of $v(S)$, which must be consistent (wrt. $\models_{\preceq_r}$) with GM. Let $Gr(v(S))$ be the set of all ground instances of $v(S)$. As usual, we will create an interpretation M. But M will be created in such a way that $M \models_{\preceq_r} S$ which implies that $M \models S$. Informally, the model is constructed inductively on the clsures of $Gr(v(S))$, using the ordering $\prec_i$. The idea of using $\prec_i$ is that the inference system takes place over nonground clauses, but the completeness proof works over ground instances of those clauses. In order for the ground inference to be able to be lifted, we need to remember whether the clause it was an instance of was ground or not.

Definition 4 Let S be a set of nonground clauses and GM be a satisfying truth assignment of $g(S)$. For a given ground closure $C \cdot \sigma$ of the form $(\Gamma \vee s = t) \cdot \sigma$ in $Gr(v(S))$, define $I_{C\cdot\sigma} = \{(s = t)\sigma\}$ if and only if all the following conditions hold:

1. $(s = t) \cdot \sigma$ is maximal in $C \cdot \sigma$ wrt. $\preceq_i$,
2. $M_{\prec_i C\cdot\sigma} \not\models_{\preceq_r} C\sigma$,
3. there is no $u \neq v \in GM^-$ such that $M_{\prec_i C\cdot\sigma} \cup (s = t)\sigma \models_{\preceq_r} u = v$,
4. $(s = t)\sigma$ is left irreducible by $M_{\prec_i C\cdot\sigma}$, and
5. Γ does not contain an equation $s' = t'$ such that $M_{\prec_i C\cdot\sigma} \models_{\preceq_r} (s = s')\sigma$ implies $M_{\prec_i C\cdot\sigma} \models_{\preceq_r} (t = t')\sigma$,

where $M_{\prec_i C\cdot\sigma} = \bigcup_{D\cdot\theta \prec_i C\cdot\sigma \wedge D\cdot\theta \in Gr(v(S))} I_{D\cdot\theta} \cup GM^+$; otherwise $I_{C\cdot\sigma} = \emptyset$. We say that $C \cdot \sigma$ *produces* $(s = t)\sigma$ when $I_{C\cdot\sigma} = \{(s = t)\sigma\}$, and $C \cdot \sigma$ is called a *productive* closure.

Definition 5 Define $M_{C\cdot\sigma} = M_{\prec_i C\cdot\sigma} \cup I_{C\cdot\sigma}$.
Define $M_\infty = \bigcup_{C\cdot\sigma \in Gr(v(S))} M_{C\cdot\sigma}$.

Let us compare this definition with the usual definition in Bachmair and Ganzinger's model construction process. The first difference from the usual completeness proof is that we build a model using $\models_{\preceq_r}$ instead of $\models$. Recall that in SMELS we begin with a model of the ground clauses, and we extend this to a model

of all the nonground clauses. Therefore the model construction is only defined over the nonground clauses. The second difference is that we begin our construction using GM^+ instead of the empty set as is normally done. The third difference is that whenever we want to add an equation to the model, we can only add it if the model is consistent with GM^-. As a consequence, the completeness proof in our case is more difficult than usual. One of the most difficult tasks is to prove the confluence of the model constructed. In the usual model construction it is trivial, as a result of the fact that only reduced equations are added to the model. Here, it is not so simple. It is true that we only add reduced literals to the model. So literals added during the model construction process can be assumed to be reduced on the left-hand side. Equations in GM^+ are reduced by GM^+ by definition. However, it is possible that we may add an equation during the model construction process that reduces the left hand side of an equation from GM^+. Therefore, the model we are constructing may not be fully reduced. But by saturation, we can show that in the end the model will be confluent. As in usual completeness proof, the confluence of the model constructed is the key to prove completeness of SMELS.

Lemma 1 *Let $C \equiv \Gamma \vee u \bowtie v$ and D be ground instances in $Gr(v(S))$ with $D \succeq_i C$. Then $M_C \models_{\preceq_r} u = v$ if and only if $M_D \models_{\preceq_r} u = v$ if and only if $M_\infty \models_{\preceq_r} u = v$.*

Proof If $M_C \models_{\preceq_r} u = v$ then $M_D \models_{\preceq_r} u = v$ and $M_\infty \models_{\preceq_r} u = v$ because $M_C \subseteq M_D \subseteq M_\infty$.

On the other hand, assume that $M_C \not\models_{\preceq_r} u = v$, i.e. $u = v$ is not implied by smaller equations in M_C. We distinguish two cases, depending whether $u \bowtie v$ is a closure with ground skeleton or nonground skeleton:

1. $u \bowtie v$ has a ground skeleton. Remember that the satisfying truth assignment GM is extended to ground literals in nonground clauses in an arbitrary way. Then either $GM \models u = v$ or $GM \models u \neq v$. Since GM is reduced, $GM \models u = v$ implies that $GM^+ \models_{\preceq_r} u = v$, and hence $M_C \models_{\preceq_r} u = v$, a contradiction. If $GM \models u \neq v$ then $GM^+ \not\models_{\preceq_r} u = v$; therefore $M_D \not\models_{\preceq_r} u = v$ and $M_\infty \not\models_{\preceq_r} u = v$.
2. $u \bowtie v$ has a nonground skeleton. If $s = t$ is an equation in $M_\infty \setminus M_C$, then we know that $s = t \succ_i u \bowtie v$. Therefore $M_D \not\models_{\preceq_r} u = v$ and $M_\infty \not\models_{\preceq_r} u = v$.

In both cases, if $M_C \not\models_{\preceq_r} u = v$ then $M_D \not\models_{\preceq_r} u = v$ and $M_\infty \not\models_{\preceq_r} u = v$. □

Lemma 2 *If a closure of the form $(D \vee s = t) \cdot \sigma$ generates the rule $(s = t)\sigma$, then $M_\infty \not\models_{\preceq_r} D\sigma$.*

Proof By definition of model generation we have that $M_{\prec_i (D \vee s=t)\cdot\sigma} \not\models_{\preceq_r} (D \vee s = t)\sigma$, which implies that $M_{\prec_i (D \vee s=t)\cdot\sigma} \not\models_{\preceq_r} D\sigma$. Since $(s = t) \cdot \sigma \succ_i D \cdot \sigma$, we must have that $M_{(D \vee s=t)\sigma} \not\models_{\preceq_r} D\sigma$. Then, by Lemma 1 we conclude $M_\infty \not\models_{\preceq_r} D\sigma$. □

Lemma 3 *Let S be a set of clauses and let GM be a satisfying truth assignment of $g(S)$, defined on $GL(S)$. Let M_∞ be the model constructed from $Gr(v(S))$. Then $M_\infty \models_{\preceq_r} j(L)$, for every literal L in GM and every $(GM, g(S))$-justification function j.*

Proof By definition $GM \models j(L)$ for every literal L in GM. We know that GM is reduced, therefore if $s = t$ is an equation in $j(L)$ then $GM \models s = t$ iff $GM^+ \vdash s =$

t iff $M_\infty \models_{\preceq_r} s = t$. Similarly if $s \neq t$ is a disequation in $j(L)$ then $GM \models s \neq t$ iff $GM \not\models s = t$ iff $GM^+ \not\vdash s = t$ iff $M_\infty \not\models_{\preceq_r} s = t$ iff $M_\infty \models_{\preceq_r} s \neq t$. □

Lemma 4 *Let S be saturated with respect to SMELS, and M_∞ is the model constructed from $Gr(v(S))$. Let A be a ground instance of an equation, and $M^A = \{B \prec_r A \mid B \in M_\infty\}$, and $Gr(v(S))^A = \{C \in Gr(v(S)) \mid C \prec_i A\}$. Suppose that $M^A \models_{\preceq_r} Gr(v(S))^A$. Then M^A is confluent.*

Proof We create inductively a subset N of M^A using the ordering $\prec_r$. More precisely, let us define $N = \bigcup_{u=v \in M^A} N_{u=v}$, where $N_{u=v}$ is defined as follows:

- if $u = v$ is left-irreducible by $N_{\prec_r u=v}$ then $N_{u=v} = N_{\prec_r u=v} \cup \{u = v\}$,
- otherwise $N_{u=v} = N_{\prec_r u=v}$,

where $N_{\prec_r u=v} = \bigcup_{u'=v' \prec_r u=v \wedge u'=v' \in M^A} N_{u'=v'}$.

Since every rule in N is left-irreducible by other rules, N is clearly confluent. So if we can show that $N \models M^A$ then M^A and N are logically equivalent, which implies that M^A is also confluent.

Suppose that $N \not\models M^A$, then let $u_0 = v_0$ be the smallest equation in M^A such that $N \not\models u_0 = v_0$. So then $u_0 = v_0$ must be left reducible by some equation $s_0 = t_0$ in N, because otherwise $u_0 = v_0$ would be in N. We consider different cases depending on whether $u_0 = v_0$ and $s_0 = t_0$ come from GM^+ or are produced in the model construction process:

1. Both $u_0 = v_0$ and $s_0 = t_0$ are produced during the model construction. Let $u_0 = v_0 \equiv (u = v)\sigma$ and $s_0 = t_0 \equiv (s = t)\sigma$. Let $(C \vee u = v) \cdot \sigma$ and $(C' \vee s = t) \cdot \sigma$ be the two ground closures producing $(u = v)\sigma$ and $(s = t)\sigma$. Since $u_0 = v_0$ is left reducible by $s_0 = t_0$, we have $u\sigma|_p \equiv s\sigma$ for some position p in $u\sigma$. We consider two possibilities, depending on whether p is a (below) variable position:

   Inference. $u|_p$ is not a (below) variable position of u. Then consider an inference *Nonground Superposition*

   $$\frac{C \vee u[s']_p = v \quad C' \vee s = t}{\big(C \vee C' \vee u[t]_p = v\big)\,\sigma'}$$

   where σ' is the most general unifier of s and s', a ground instance of $(C \vee C' \vee u[t]_p = v)\sigma'$ is $(C \vee C' \vee u[t]_p = v)\sigma$, and $(u[t] = v)\sigma \prec_r (u[s'] = v)\sigma$. By Lemma 2, we have $M_\infty \not\models_{\preceq_r} C\sigma$ and $M_\infty \not\models_{\preceq_r} C'\sigma$. If $A \prec_i C \cdot \sigma$ (resp. $A \prec_i C' \cdot \sigma$) then by definition of model construction $M^A \not\models_{\preceq_r} C\sigma$ (resp. $M^A \not\models_{\preceq_r} C\sigma$). Otherwise, it follows from Lemma 1 that $M^A \not\models_{\preceq_r} C\sigma$ (resp. $M^A \not\models_{\preceq_r} C'\sigma$). By the hypothesis of the lemma, we know that $M^A \models_{\preceq_r} (C \vee C' \vee u[t] = v)\sigma$. So we must have that $M^A \models_{\preceq_r} (u[t] = v)\sigma$. To derive a contradiction, we only need to prove that $N \models (u[t] = v)\sigma$. This is done by induction on $\prec_i$. By the hypothesis of the lemma, S is saturated, therefore the inference is redundant, which means that either one of the premises is redundant or the conlusion is redundant, or the conclusion is in the S. Ground instances of the premises are

productive, so they cannot be redundant. Thus either the conlusion is in S, or it is redundant. We consider the following cases:

(a) $(C \vee C' \vee u[t] = v)\sigma$ is not redundant:

i. $(u[t] = v)\sigma$ has been produced. Then $N \models (u[t] = v)\sigma$ and we are done. Otherwise this would contradict the minimality of $u_0 = v_0$.

ii. $(u[t] = v)\sigma$ has not been produced because it is not maximal. Then $(u[t] = v)$ must be implied by smaller equations in M^A, because otherwise we would have $M^A \not\models_{\preceq_r} (C \vee C' \vee u[t] = v)\sigma$. By the induction hypothesis, all those equations are implied by N, which also means $N \models (u[t] = v)\sigma$.

iii. $(u[t] = v)\sigma$ is maximal but has not been produced. We know that $M^A \models_{\preceq_r} (u[t] = v)\sigma$, and therefore $M^{(u[t]=v)\sigma} \models (u[t] = v)\sigma$, i.e. $u[t] = v$ is implied by smaller equations in M^A. By the induction hypothesis, all those equations are implied by N, which also means $N \models (u[t] = v)\sigma$.

(b) $(C \vee C' \vee u[t] = v)\sigma$ is redundant. We consider two cases according to the definition of redundant clauses.

i. $(C \vee C' \vee u[t] = v)\sigma$ is properly subsumed by a clause D in S. Then D must contain $u[t] = v$, because otherwise $M^A \not\models_{\preceq_r} D\sigma$, which contradicts the hypothesis of the lemma. Let D be of the form $D' \vee u[t] = v$. We consider the following cases:

A. $(u[t] = v)\sigma$ has been produced. Then $N \models (u[t] = v)\sigma$ and we are done.

B. $(u[t] = v)\sigma$ has not been produced because it is not maximal. Then $(u[t] = v)\sigma$ must be implied by smaller equations in M^A, because otherwise we would have $M^A \not\models_{\preceq_r} (D' \vee u[t] = v)\sigma$. By the induction hypothesis, all those equations are implied by N, which also means $N \models (u[t] = v)\sigma$.

C. $(u[t] = v)\sigma$ is maximal but has not been produced. We know that $M^A \models_{\preceq_r} (u[t] = v)\sigma$, and therefore $M^{(u[t]=v)\sigma} \models (u[t] = v)\sigma$, i.e. $(u[t] = v)\sigma$ is implied by smaller equations in M^A. By the induction hypothesis, all those equations are implied by N, which also means $N \models (u[t] = v)\sigma$.

ii. There exist a set of equations $E \subseteq Gr(S)$ and a ground instance $D \cdot \sigma'$ such that

- each equation in E is smaller than $(C \vee C' \vee u[t] = v) \cdot \sigma$ wrt. both $\prec_i$ and $\prec_r$, and
- $D \cdot \sigma'$ is in $Gr(S)$ or $D \cdot \sigma'$ is a tautology, and
- $D \cdot \sigma'$ is smaller than $(C \vee C' \vee u[t] = v) \cdot \sigma$ wrt. both $\prec_i$ and $\prec_r$, and

– each literal $L\sigma'$ in $D\sigma'$ is equivalent to a literal $L'\sigma$ in $(C \vee C' \vee u[t] = v)\sigma$ modulo E, and $L \cdot \sigma'$ is smaller or equal to $L' \cdot \sigma$ wrt. both $\prec_i$ and $\prec_r$.

We claim that there is one equation $(u' = v')\sigma'$ in $D\sigma'$ which is equivalent to $(u[t] = v)\sigma$ modulo E, and $(u' = v') \cdot \sigma'$ is smaller than or equal to $(u[t] = v) \cdot \sigma$ wrt. both $\prec_i$ and $\prec_r$. Because otherwise $M^A \not\models_{\preceq_r} D\sigma'$, which contradicts the hypothesis of the lemma. By the minimality of $u_0 = v_0$, we must have $N \models E$. If $(u' = v') \cdot \sigma' \prec_i (u[t] = v) \cdot \sigma$ then by induction hypothesis we also have $N \models (u' = v')\sigma'$, which implies that $N \models (u[t] = v)\sigma$. If $(u' = v') \cdot \sigma'$ is equal to $(u[t] = v) \cdot \sigma$ wrt. both $\prec_i$ and $\prec_r$, then D is of the form $D' \vee u[t] = v$. We now consider the following cases:

A. $(u[t] = v)\sigma$ has been produced. Then $N \models (u[t] = v)\sigma$ and we are done.
B. $(u[t] = v)\sigma$ has not been produced because it is not maximal. Then $(u[t] = v)\sigma$ must be implied by smaller equations in M^A, because otherwise we would have $M^A \not\models_{\preceq_r} D' \vee (u[t] = v)\sigma$. By the induction hypothesis, all those equations are implied by N, which also means $N \models (u[t] = v)\sigma$.
C. $(u[t] = v)\sigma$ is maximal but has not been produced. We know that $M^A \models_{\preceq_r} (u[t] = v)\sigma$, and therefore $M^{(u[t]=v)\sigma} \models (u[t] = v)\sigma$, i.e. $(u[t] = v)\sigma$ is implied by smaller equations in M^A. By the induction hypothesis, all those equations are implied by N, which also means $N \models (u[t] = v)\sigma$.

Lifting. $u|_p$ is a (below) variable position of u. That is $p = p'.p''$ and $u|_{p'}$ is a variable x. Let θ be a ground substitution with the same domain as σ but $x\theta \equiv x\sigma[t\sigma]_{p''}$ and $y\theta \equiv y\sigma$ for all other variable y. Again we only need to prove $N \models (u[t] = v)\theta$ to derive a contradiction. This can be done in a similar way as above.

2. $u_0 = v_0$ is from GM^+ and $s_0 = t_0$ is produced during the model construction. Let $u_0 = v_0 \equiv (u = v)\sigma$ and $s_0 = t_0 \equiv (s = t)\sigma$. Let $(C \vee s = t) \cdot \sigma$ be the ground closure producing $(s = t)\sigma$. Since $u_0 = v_0$ is left reducible by $s_0 = t_0$, we have $u\sigma|_p \equiv s\sigma$ for some position p in $u\sigma$. Then consider an inference *Justified Superposition From*

$$\frac{C \vee s = t}{\big(C \vee \neg j\big(u[s']_p = v\big) \vee u[t]_p = v\big)\,\sigma'}$$

where σ' is the most general unifier of s and s', a ground instance of $(C \vee \neg j(u[s']_p = v) \vee u[t]_p = v)\sigma'$ is $(C \vee \neg j(u[s']_p = v) \vee u[t]_p = v)\sigma$, and $(u[t] = v)\sigma \prec_r (u[s'] = v)\sigma$. The rest of the proof for this case can be handled as in previous case.

3. $u_0 = v_0$ is produced during the model construction, and $s_0 = t_0$ is from GM^+. This is impossible by the definition of model construction.
4. Both $u_0 = v_0$ and $s_0 = t_0$ are from GM^+. This is impossible since GM^+ is reduced.

Summing up N is confluent and so is M^A. □

Lemma 5 *Let S be saturated with respect to SMELS and not contain the empty clause. Let GM be the ground model of $g(S)$, defined on $GL(S)$. Let M_∞ be the model constructed from $Gr(v(S))$. Then $M_\infty \models_{\preceq_r} Gr(v(S))$.*

Proof Assume that there is a smallest ground instance $C \cdot \sigma$ wrt. $\preceq_i$ in $Gr(v(S))$ such that $M_\infty \not\models_{\preceq_r} C\sigma$. We will analyze why $C \cdot \sigma$ has not produced any equation in M_∞. There are several cases to be considered depending on the occurrences of its maximal term s:

1. $C\sigma$ contains the equation $s = s$. This is impossible since $M_\infty \not\models_{\preceq_r} C\sigma$.
2. $C\sigma$ is of the form $(D \vee s \neq s)\sigma$. Then consider an inference *Eq. Resolution*

$$\frac{D \vee s \neq s}{D}$$

where $D \cdot \sigma$ is smaller than $C \cdot \sigma$ wrt. $\preceq_i$. We have that $M_\infty \not\models_{\preceq_r} D\sigma$. If $D \cdot \sigma$ is in $Gr(v(S))$, then we get a smaller counterexample, which contradicts the minimality of $C \cdot \sigma$. If D has been removed because it is redundant, then we consider two cases according to the definition of redundant clauses.

(a) D is properly subsumed by a clause D' in S. Then D' is a counterexample that is smaller than C. And we have a contradiction with the minimality of $C \cdot \sigma$.

(b) For every ground instance $D \cdot \theta$, there exist a set of equations $E \subseteq Gr(S)$ and a ground instance $D' \cdot \theta'$ such that

- each equation in E is smaller than $D \cdot \theta$ wrt. both $\prec_i$ and $\prec_r$, and
- $D' \cdot \theta'$ is in $Gr(S)$ or $D' \cdot \theta'$ is a tautology, and
- $D' \cdot \theta'$ is smaller than $D \cdot \theta$ wrt. both $\prec_i$ and $\prec_r$, and
- each literal $L'\theta'$ in $D'\theta'$ is equivalent to a literal $L\theta$ in $D\theta$ modulo E, and $L' \cdot \theta'$ is smaller or equal to $L \cdot \theta$ wrt. both $\prec_i$ and $\prec_r$.

If $D' \cdot \theta'$ is a tautology then $M_\infty \models_{\preceq_r} D'\sigma$, and hence $M_\infty \models_{\preceq_r} D\sigma$, which implies that $M_\infty \models_{\preceq_r} C\sigma$, a contradiction. Otherwise we have $M_\infty \not\models_{\preceq_r} D'\theta'$, and again we get a contradiction with the minimality of $C \cdot \sigma$.

3. $C\sigma$ is of the form $(D \vee s \neq t)\sigma$. Since $M_\infty \not\models_{\preceq_r} C\sigma$, we must have $M_\infty \models_{\preceq_r} (s = t)\sigma$. By the minimality of $C \cdot \sigma$ as a counterexample, we have $M_\infty \models_{\preceq_r} Gr(v(S))^{(s=t)\cdot\sigma}$, which implies that $M_\infty^{(s=t)\sigma} \models_{\preceq_r} Gr(v(S))^{(s=t)\cdot\sigma}$. By Lemma 4, $M_\infty^{(s=t)\sigma}$ is confluent. This implies that $M_\infty \models_{\preceq_r} (s = t)\sigma$ if and only if $M_\infty^{(s=t)\sigma} \vdash (s = t)\sigma$ or $(s = t)\sigma$ is in M_∞. In both cases, $s\sigma$ must be reducible by some rule

$l = r$ in M_∞ ($l = r$ could be $(s = t)\sigma$), that is $s\sigma|_p \equiv l$ for some position p in $s\sigma$. We consider two cases:

(a) $l = r$ is in GM^+. We consider two possibilities, depending on whether p is a (below) variable position:

Inference. $s|_p$ is not a (below) variable position. Then consider an inference *Justified Superposition Into*

$$\frac{D \vee s\left[l'\right] \neq t}{\left(D \vee \neg j(l = r) \vee s[r] \neq t\right)\sigma'}$$

where σ' is the most general unifier of l and l', a ground instance of $(D \vee \neg j(l = r) \vee s[r] \neq t)\sigma'$ is $(D \vee \neg j(l = r) \vee s[r] \neq t)\sigma$, and $(D \vee \neg j(l = r) \vee s[r] \neq t) \cdot \sigma \prec_i C \cdot \sigma$. Since $M_\infty \not\models_{\preceq_r} C\sigma$ we have that $M_\infty \not\models_{\preceq_r} D\sigma$. By Lemma 3, $M_\infty \models_{\preceq_r} j(l = r)$, or equivalently $M_\infty \not\models_{\preceq_r} \neg j(l = r)$. Therefore $M_\infty \not\models_{\preceq_r} (D \vee \neg j(l = r) \vee s[r] \neq t)\sigma$. Again we consider two cases depending on whether $(D \vee \neg j(l = r) \vee s[r] \neq t)\sigma$ is in $Gr(v(S))$ or redundant. As previously we can show that we have a contradiction in both cases.

Lifting. $s|_p$ is a (below) variable position of s. That is $p = p'.p''$ and $s|_{p'}$ is a variable x. Let θ be a ground substitution with the same domain as σ but $x\theta \equiv x\sigma[r\sigma]_{p''}$ and $y\theta \equiv y\sigma$ for all other variable y. Then we have that $C \cdot \theta \prec_i C \cdot \sigma$ and $M_\infty \not\models_{\preceq_r} C\theta$, a contradiction with the minimality of $C \cdot \sigma$.

(b) $l = r$ is produced by a closure $(D' \vee l' = r') \cdot \theta$ and $l = r \equiv (l' = r')\theta$. We consider two possibilities, depending on whether p is a (below) variable position:

Inference. $s|_p$ is not a (below) variable position. Then consider an inference *Nonground Superposition*

$$\frac{D \vee s\left[l''\right] \neq t \quad D' \vee l' = r'}{\left(D \vee D' \vee s\left[r'\right] \neq t\right)\sigma'}$$

where σ' is the most general unifier of l' and l'', a ground instance of $(D \vee D' \vee s[r'] \neq t)\sigma'$ is $(D \vee D' \vee s[r'] \neq t)\sigma$, and $(D \vee D' \vee s[r'] \neq t) \cdot \sigma \prec_i C \cdot \sigma$. Also, we have that $M_\infty \not\models_{\preceq_r} D\sigma$ and $M_\infty \not\models_{\preceq_r} D'\sigma$. That implies $M_\infty \not\models_{\preceq_r} (D \vee D' \vee s[r'] \neq t)\sigma$. By the same argument as above, we can derive a contradiction with the minimality of $C \cdot \sigma$.

Lifting. $s|_p$ is a (below) variable position of s. That is $p = p'.p''$ and $s|_{p'}$ is a variable x. Let θ be a ground substitution with the same domain as σ but $x\theta \equiv x\sigma[r'\sigma]_{p''}$ and $y\theta \equiv y\sigma$ for all other variable y. Again we can derive a contradiction in a similar way as above.

4. $C\sigma$ is of the form $(D \vee s = t \vee s' = t')\sigma$ and $(s = t)\sigma$ has not been produced because $M_{\prec_i C\cdot\sigma} \models_{\preceq_r} (s = s')\sigma$ implies $M_{\prec_i C\cdot\sigma} \models_{\preceq_r} (t = t')\sigma$. Then consider an inference *Eq. Factoring*

$$\frac{D \vee s = t \vee s' = t'}{\left(D \vee t \neq t' \vee s = t'\right)\sigma'}$$

where σ' is the most general unifier of s and s', a ground instance of $(D \vee t \neq t' \vee s = t')\sigma'$is $(D \vee t \neq t' \vee s = t')\sigma$, and $(D \vee t \neq t' \vee s = t') \cdot \sigma \prec_i C \cdot \sigma$. By Lemma 1, $M_{\prec_i C\cdot\sigma} \models_{\preceq_r} (t = t')\sigma$ implies that $M_\infty \models_{\preceq_r} (t = t')\sigma$, or equivalently $M_\infty \not\models_{\preceq_r} (t \neq t')\sigma$. We also have that $M_\infty \not\models_{\preceq_r} (D \vee s' = t')\sigma$. Therefore $M_\infty \not\models_{\preceq_r} (D \vee t \neq t' \vee s' = t')\sigma$; and again by the same argument as previously we obtain a contradiction.
5. $C\sigma$ is of the form $(D \vee s = t)\sigma$ and $(s = t)\sigma$ has not been produced because s is reducible by some rule $l = r$ in $M_{\prec_i C\cdot\sigma}$ ($l = r$ could be $(s = t)\sigma$). That is $s\sigma|_p \equiv l$ for some position p in $s\sigma$. We consider two cases:

   (a) $l = r$ is in GM^+. We consider two possibilities, depending on whether p is a (below) variable position:

   Inference. $s|_p$ is not a (below) variable position. Then consider an inference *Justified Superposition Into*

   $$\frac{D \vee s[l'] = t}{(D \vee \neg j(l = r) \vee s[r] = t)\,\sigma'}$$

   where σ' is the most general unifier of l and l', a ground instance of $(D \vee \neg j(l = r) \vee s[r] = t)\sigma'$ is $(D \vee \neg j(l = r) \vee s[r] = t)\sigma$, and $(D \vee \neg j(l = r) \vee s[r] = t) \cdot \sigma \prec_i C \cdot \sigma$. Since $M_\infty \not\models_{\preceq_r} C\sigma$ we have that $M_\infty \not\models_{\preceq_r} D\sigma$. By Lemma 3, $M_\infty \models_{\preceq_r} j(l = r)$, or equivalently $M_\infty \not\models_{\preceq_r} \neg j(l = r)$. Therefore $M_\infty \not\models_{\preceq_r} (D \vee \neg j(l = r) \vee s[r] = t)\sigma$. Again we consider two cases depending on whether $(D \vee \neg j(l = r) \vee s[r]t)\sigma$ is in $Gr(v(S))$ or redundant. As previously we can show that we have a contradiction in both cases.

   Lifting. $s|_p$ is a (below) variable position of s. That is $p = p'.p''$ and $s|_{p'}$ is a variable x. Let θ be a ground substitution with the same domain as σ but $x\theta \equiv x\sigma[r\sigma]_{p''}$ and $y\theta \equiv y\sigma$ for all other variable y. Again we have that $C \cdot \theta \prec_i C \cdot \sigma$ and $M_\infty \not\models_{\preceq_r} C\theta$, a contradiction with the minimality of $C \cdot \sigma$.

   (b) $l = r$ is produced by a closure $(D' \vee l' = r') \cdot \theta$ and $l = r \equiv (l' = r')\theta$. We consider two possibilities, depending on whether p is a (below) variable position:

   Inference. $s|_p$ is not a (below) variable position. Then consider an inference *Nonground Superposition*

   $$\frac{D \vee s[l''] = t \quad D' \vee l' = r'}{\left(D \vee D' \vee s[r'] = t\right)\sigma'}$$

   where σ' is the most general unifier of l' and l'', a ground instance of $(D \vee D' \vee s[r'] = t)\sigma'$ is $(D \vee D' \vee s[r'] = t)\sigma$, and

$(D \vee D' \vee s[r'] = t) \cdot \sigma \prec_i C \cdot \sigma$. Also, we have that $M_\infty \not\models_{\preceq_r} D\sigma$ and $M_\infty \not\models_{\preceq_r} D'\sigma$. That implies $M_\infty \not\models_{\preceq_r} (D \vee D' \vee s[r] = t)\sigma$. By the same argument as above, we can derive a contradiction with the minimality of $C \cdot \sigma$.

Lifting. $s|_p$ is a (below) variable position of s. That is $p = p'.p''$ and $s|_{p'}$ is a variable x. Let θ be a ground substitution with the same domain as σ but $x\theta \equiv x\sigma[r'\sigma]_{p''}$ and $y\theta \equiv y\sigma$ for all other variable y. Again we we have that $C \cdot \theta \prec_i C \cdot \sigma$ and $M_\infty \not\models_{\preceq_r} C\theta$, a contradiction with the minimality of $C \cdot \sigma$.

6. $C\sigma$ is of the form $(D \vee s = t)\sigma$ and $(s = t)\sigma$ has not been produced because there is a disequation $u \neq v \in GM^-$ such that $M_{\prec_i C\cdot\sigma} \cup \{(s = t)\sigma\} \models_{\preceq_r} u = v$. Let F be the set containing all the ground instances in $GM \cup \{Gr(v(S))\}$, having the form

$$(\Gamma \vee u' \neq v') \cdot \theta$$

where

- for each literal $L\theta$ in $\Gamma\theta$ we have that $M_\infty \not\models_{\preceq_r} L\theta$ and $L \cdot \theta \prec_i (s = t) \cdot \sigma$; and
- $(u' = v')\theta$ is implied by a set of equations in $M_\infty \cup \{(s = t)\sigma\}$, containing ground instances of equations, which are smaller than or equal to $(u' = v') \cdot \theta$ wrt. $\prec_r$, and smaller than or equal to $(s = t) \cdot \sigma$ wrt. $\prec_i$.

We show that $M_\infty \not\models_{\preceq_r} C'$ for each $C' \in F$. We consider two cases:

(a) $(u' \neq v') \cdot \theta$ is not maximal. Then by definition $M_\infty \not\models_{\preceq_r} (\Gamma \vee u' \neq v')\theta$.
(b) $(u' \neq v') \cdot \theta$ is maximal. Suppose that there exist a smallest closure $(\Gamma \vee u' \neq v') \cdot \theta$ wrt. $\prec_i$ in F such that $M_\infty \models_{\preceq_r} (\Gamma \vee u' \neq v')\theta$. By definition $(u' = v')\theta$ is implied by a set of equations in $M_\infty \cup \{(s = t)\sigma\}$, containing ground instances of equations, which are smaller than or equal to $(u' = v') \cdot \theta$ wrt. $\prec_r$, and smaller than or equal to $(s = t) \cdot \sigma$ wrt. $\prec_i$. This also means that $M_\infty \cup \{(s = t)\sigma\} \vdash (u' = v')\theta$. Let $(l_1 = r_1)\theta_1, \ldots, (l_n = r_n)\theta_n$ be all the rules in the rewrite proof of $(u' = v')\theta$ using $M_\infty \cup \{(s = t)\sigma\}$. Then $u'\theta$ must be reducible by one of the rules. Let us say that this rule is $(l_1 = r_1)\theta_1$ and the productive closure is $(C_1 \vee l_1 = r_1) \cdot \theta_1$. We consider an inference *Justified/Nonground Superposition* involving $\Gamma \vee u' \neq v'$ and $C_1 \vee l_1 = r_1$, whose conclusion D' has a ground instance $D' \cdot \theta'$ smaller than $(\Gamma \vee u' \neq v') \cdot \theta$ wrt. $\prec_i$. Furthermore we can show that $M_\infty \models_{\preceq_r} D'\theta'$. This contradicts the minimality of $(\Gamma \vee u' \neq v') \cdot \theta$.

Now we know that $M_\infty \not\models_{\preceq_r} C'$ for each $C' \in F$. By definition of F, $u \neq v$ must be in F, and hence $M_\infty \not\models_{\preceq_r} u \neq v$, which is clearly a contradiction with the conditions of the model construction. □

Lemma 6 *Let S be saturated with respect to SMELS and not contain the empty clause. Let GM be the ground model of $g(S)$, defined on $GL(S)$. Let M_∞ be the model constructed from $Gr(v(S))$. Then M_∞ is confluent.*

Proof As in Lemma 4, we build a left-reduced subset N of M_∞, which is clearly confluent. Then we show that $N \models M_\infty$.

Suppose that $N \not\models M_\infty$, let $u_0 = v_0$ be the smallest equation in M_∞ such that $N \not\models u_0 = v_0$. So then $u_0 = v_0$ must be left reducible by some equation $s_0 = t_0$ in N, because otherwise $u_0 = v_0$ would be in N. We consider different cases depending on whether $u_0 = v_0$ and $s_0 = t_0$ come from GM^+ or are produced in the model construction process:

1. Both $u_0 = v_0$ and $s_0 = t_0$ are produced during the model construction. Let $u_0 = v_0 \equiv (u = v)\sigma$ and $s_0 = t_0 \equiv (s = t)\sigma$. Let $(C \vee u = v) \cdot \sigma$ and $(C' \vee s = t) \cdot \sigma$ be the two ground closures producing $(u = v)\sigma$ and $(s = t)\sigma$. Since $u_0 = v_0$ is left reducible by $s_0 = t_0$, we have $u\sigma|_p \equiv s\sigma$ for some position p in $u\sigma$. We consider two possibilities, depending on whether p is a (below) variable position:

   Inference. $u|_p$ is not a below variable position of u. Then consider an inference *Nonground Superposition*

   $$\frac{C \vee u[s']_p = v \quad C' \vee s = t}{(C \vee C' \vee u[t]_p = v)\sigma'}$$

   where σ' is the most general unifier of s and s', a ground instance of $(C \vee C' \vee u[t]_p = v)\sigma'$ is $(C \vee C' \vee u[t]_p = v)\sigma$, and $(u[t] = v)\sigma \prec_r (u[s'] = v)\sigma$. By Lemma 2, we have $M_\infty \not\models_{\preceq_r} C\sigma$ and $M_\infty \not\models_{\preceq_r} C'\sigma$. Moreover, by Lemma 5, we know that $M_\infty \models_{\preceq_r} C \vee C' \vee u[t] = v$, which implies that $M_\infty \models_{\preceq_r} u[t] = v$. We consider the following cases:

   (a) $(C \vee C' \vee u[t] = v)\sigma$ is not redundant.

   i. $(u[t] = v)\sigma$ has been produced in M_∞.
   ii. $(u[t] = v)\sigma$ has not been produced in M_∞ because it is not maximal.
   iii. $(u[t] = v)\sigma$ is maximal but has not been produced in M_∞.

   (b) $(C \vee C' \vee u[t] = v)\sigma$ is redundant.

   In each case we can derive a contradiction, as done in Lemma 4.

   Lifting. $u|_p$ is a below variable position of u. That is $p = p'.p''$ and $u|_{p'}$ is a variable x. Let θ be a ground substitution with the same domain as σ but $x\theta \equiv x\sigma[t\sigma]_{p''}$ and $y\theta \equiv y\sigma$ for all other variable y. Again we only need to prove $N \models (u[t] = v)\theta$ to derive a contradiction. This can be done in a similar way as above.

2. $u_0 = v_0$ is from GM^+ and $s_0 = t_0$ is produced during the model construction. Let $u_0 = v_0 \equiv (u = v)\sigma$ and $s_0 = t_0 \equiv (s = t)\sigma$. Let $(C \vee s = t) \cdot \sigma$ be the ground closure producing $(s = t)\sigma$. Since $u_0 = v_0$ is left reducible by $s_0 = t_0$, we have $u\sigma|_p \equiv s\sigma$. Then consider an inference *Justified Superposition From*

   $$\frac{C \vee s = t}{\big(C \vee \neg j\big(u[s']_p = v\big) \vee u[t]_p = v\big)\sigma'}$$

   where σ' is the most general unifier of s and s', a ground instance of $(C \vee \neg j(u[s']_p = v) \vee u[t]_p = v)\sigma'$ is $(C \vee \neg j(u[s']_p = v) \vee u[t]_p = v)\sigma$, and $(u[t] = v)\sigma \prec_r (u[s'] = v)\sigma$. The rest of the proof for this case can be handled as in previous case.

3. $u_0 = v_0$ is produced during the model construction, and $s_0 = t_0$ is from GM^+. This is impossible by the definition of model construction.
4. Both $u_0 = v_0$ and $s_0 = t_0$ are from GM^+. This is impossible since GM^+ is reduced. □

Theorem 1 *Let S be saturated with respect to SMELS. Then, S is satisfiable if and only if it does not contain the empty clause.*

Proof The only if direction is implied by the soundness of SMELS.

For the if direction, we show that if S does not contain the empty clause then we can exhibit a model of S. Let M_∞ be the model constructed from $Gr(v(S))$. We will show that $M_\infty \models S$.

The proof is exactly the same as in Lemma 5, i.e. we derive a smaller counterexample from the minimal counterexample, but this time the counterexample is with respect to $\models$.

Assume that there is a smallest clause C in $Gr(v(S))$ such that $M_\infty \not\models C$. We will analyze the position of occurrences of the maximal term s of C. There are exactly the same cases to consider, as in Lemma 5. In each case, we use the confluence of M_∞, which is stated in Lemma 6, to get a smaller counterexample. □

Lemma 7 *Let S be a set of clauses and C be a clause redundant w.r.t S. Let S' be a set of clauses obtained by adding the conclusion of a SMELS(GM, j) inference with premises from S' to S, or by removing from S a clause which is redundant w.r.t. S. Then C is still redundant w.r.t. S'.*

Proof For the generation case, it is easy to see that if C is redundant w.r.t S then C is redundant w.r.t S' obtained by adding the conclusion of a SMELS(GM, j) inference with premises in S to S itself.

Let us now consider the deletion case. We need to consider the following cases:

1. C is properly subsumed by $C' \in S$, and C' is properly subsumed by $C'' \in S'$. Then C'' must properly subsume C. So C is redundant in $S \setminus \{C'\}$.
2. C is properly subsumed by C', and $C'\theta$ is redundant in $Gr(S)$ for all θ, by properties 1–5 of redundancy. We need to show that, for all θ, $C\theta$ is redundant in $Gr(S \setminus \{C'\})$ by properties 1–5 of redundancy. Since C is properly subsumed by C', that means $C'\theta$ contains a subset of the literals of $C\theta$. Then the set E that makes $C'\theta$ redundant also makes $C\theta$ redundant.
3. $C\theta$ is redundant in $Gr(S)$ for all θ, by properties 1–5 of redundancy. Assume that $C'\theta$ is contained in the E that makes $C\theta$ redundant, and that $C'\theta$ is properly subsumed in S. Let C'' be the clause that subsumes C'. Then $C''\theta$ is a subset of $C'\theta$. If $C'\theta$ is the clause $D\sigma$ in the definition of redundancy of $C\theta$, then $C''\theta$ will also serve as the clause $D\sigma$ used to make $C\theta$ redundant. If $C'\theta$ is not the clause $D\sigma$ then $C''\theta$ can replace $C'\theta$, because $C''\theta$ logically implies $C'\theta$.
4. $C\theta$ is redundant in $Gr(S)$ for all θ by properties 1–5 of redundancy. Assume that $C'\theta$ is contained in the E that makes $C\theta$ redundant and that $C'\theta$ is also redundant in $Gr(S)$ for all θ by properties 1–5 of redundancy. Let E' be the set that makes $C'\theta$ redundant, and let $D'\sigma'$ be the clause $D\sigma$ mentioned in property 3 of redundancy. If $C'\theta$ is the clause $D\sigma$ mentioned in property 3 for the redundancy of $C\theta$, then redundancy of $C\theta$ can be shown by $(E \setminus \{C'\theta\}) \cup E'$,

where $D'\sigma'$ is the clause $D\sigma$ mentioned in property 3. If $C'\theta$ is not the clause mentioned in property 3, then redundancy still holds because E' logically implies $C'\theta$. □

Lemma 8 *Let* $(S_0, G_0, j_0), (S_1, G_1, j_1), \cdots$ *be a fair SMELS derivation. Then* S_∞ *is saturated w.r.t SMELS.*

Proof Let T_0 be the sequence $G_n, G_{n+1}, \cdots$, where n is a justification stabilizer of the derivation. Therefore, T_0 is just the subsequence of the derivation where we can be assured that the justification function is well-founded. We define a sequence of sequences inductively based on the enumeration $A_1, A_2, \cdots$ of the positive literals. We need to define what T_i is, in terms of T_{i-1} and A_i from the enumeration of positive literals. The idea is to make T_i be a subsequence of T_{i-1} such that either A_i or $\neg A_i$ occurs in all ground truth assignments, and has the same justification each time.

We define T_i as follows:

1. If there exists an M such that (A_i, M) is persistent in the sequence T_{i-1}, then T_i is the subsequence of all triples (S_k, G_k, j_k) in T_{i-1} such that $A_i \in G_k$ and $j_k(A_i) = M$. In this case, define $Gprod_i = \{A_i\}$, and define $jprod_i = M$
2. Else if there exists an M such that $(\neg A_i, M)$ is persistent in the sequence T_{i-1}, then T_i is the subsequence of all triples (S_k, G_k, j_k) in T_{i-1} such that $\neg A_i \in G_k$ and $j_k(\neg A_i) = M$. In this case, define $Gprod_i = \{\neg A_i\}$, and define $jprod_i = M$.
3. Else T_i is the subsequence of all triples (S_k, G_k, j_k) in T_{i-1} such that $A_i \notin G_k$ and $\neg A_i \notin G_k$. In this case, define $Gprod_i = \emptyset$.

Let $GM = \bigcup Gprod_i$, and let j be the justification function so that for all $L \in GM$ with $Gprod_i = \{L\}$, we have $j(L) = jprod_i$.

By construction of GM and by definition of justification stabilizer, if L occurs infinitely often in GM then it must occur infinitely often with the same justification.

As usual, if an inference does not involve GM, then this inference will be redundant in some S_i, and it follows from Lemma 7 that this inference will be redundant in S_∞. If there is an inference involving GM, then any nonground clause in that inference is persistent, but also any clause in GM must have appeared infinitely often with the same justification function, so the inference is persistent. By fairness every persistent SMELS(GM, j) inference is redundant w.r.t S_i for some $i \geq 0$. It follows from Lemma 7 that this SMELS(GM, j) inference is redundant w.r.t S_∞.

Summing up this means that every SMELS(GM, j) inference with a premise in S_∞ is redundant w.r.t S_∞. In other words, S_∞ is saturated w.r.t SMELS. □

Theorem 2 *If* $(S_0, G_0, j_0), (S_1, G_1, j_1), \cdots$ *is a fair SMELS derivation then* S_0 *is unsatisfiable if and only if* S_i *contains the empty clause for some* $i \geq 0$.

Proof It follows from Lemma 8 that S_∞ is saturated w.r.t SMELS. By definition of derivation, S_∞ is logically equivalent to S_0, which implies that S_∞ is unsatisfiable if and only if S_0 is unsatisfiable. By Theorem 1 S_∞ is unsatisfiable if and only if S_∞ contains the empty clause, and hence S_i contains the empty clause for some $i \geq 0$. □

6 Implementation

We have implemented SMELS and our implementation follows Algorithm 1. The algorithm starts out (line 2) by splitting the input set of clauses *Input* into two sets of clauses, the first set *Ground* only contains ground clauses and the second set *Nonground* only contains nonground clauses. The set of ground clauses *Ground* is then provided to the SAT solver to find a justified model *JustifiedModel* if there exists one (lines 3 and 4); otherwise SMELS terminates returning UNSAT (line 15). *JustifiedModel* and *Nonground* are sent to the Superposition engine (line 6) to check for satisfiability and to derive new ground clauses (line 7). There are several possibilities to consider. First, the Superposition procedure may not terminate but derive new ground clauses in the meantime. One way to handle this is to stop the Superposition procedure as soon as one new ground clause is derived, and then *GroundDerived*, containing exactly this new ground clause, is added to the current set of ground clauses maintained by the SAT solver to find yet another new model (lines 11 and 12). However proceeding this way is not efficient in practice because restarting the SAT solver and the Superposition procedure is computationally expensive. Therefore we have decided to stop the Superposition procedure after a timeout and then *GroundDerived*, containing all the new ground clauses derived so far, is added to the current set of ground clauses maintained by the SAT solver to find yet another new model (lines 11 and 12). Second, the Superposition procedure does not terminate and does not derive any new ground clauses then SMELS does not terminate. Finally, the Superposition procedure terminates and there are no new ground clauses derived, then the satisfiability result of SMELS is exactly the same as the one of the Superposition engine (lines 8–10). The process continues like this until the SAT solver cannot find any models, and then SMELS terminates with the result UNSAT.

Algorithm 1 SMELS

```
 1: procedure SMELS(Input)
 2:     (Ground, Nonground) := SplitInput(Input);
 3:     SatInput := CreateSatInput(Ground);
 4:     (JustifiedModel, SatReturn) := ExecuteJustifiedSat(SatInput);
 5:     while SatReturn = SAT do
 6:         SupInput := CreateSupInput(JustifiedModel, Nonground);
 7:         (GroundDerived, SupReturn) := ExecuteJustifiedSup(SupInput);
 8:         if Sizeof(GroundDerived) = 0 then
 9:             Return SupReturn;
10:         else
11:             SatInput := CreateSatInput(Ground, GroundDerived);
12:             (JustifiedModel, SatReturn) := ExecuteJustifiedSat(SatInput);
13:         end if
14:     end while
15:     Return UNSAT;
16: end procedure
```

Since we assume that *JustifiedModel* is reduced, we need to have a module for reducing models computed by the SAT solver. There are several ways to implement such a module. One way is to use off-the-shelf Rewriting rules in the Superposition

procedure. The reduction of models is rather simple, Rewriting rules apply as usual to literals in models except that now they must take into account the justification of literals. Another way would be to use a Congruence Closure algorithm, combined with a SAT solver in the DPLL(T) style [16]. This requires us to order terms appropriately to be consistent with the ordering used in the Superposition procedure. At this point, we have chosen to use the former to spare time and effort.

SMELS is flexible and modular enough to be implemented reusing an existing SAT Solver and Superposition Theorem Prover. We choose MiniSat 2.2.0 (cf. http://minisat.se) as the SAT Solver and SPASS 3.7 (cf. http://www.spass-prover.org) as the Superposition Theorem Prover. Since SMELS works on justified models and justified clauses we need to modify MiniSat and SPASS so that each literal in a propositional model has a justification and each clause also has a justification. The inferences in MiniSat and SPASS have been modified in order to meet the requirement of justified models and clauses. SMELS's source code and binary are available at http://is.hut.edu.vn/ khanhtd/project/smels.

6.1 Justified MiniSat

We have to modify MiniSat so that each literal in every propositional model has a justification. This can be done by adding a data structure to MiniSat to store justifications of all literals. In MiniSat all literals are indexed, and propositional models are encoded by boolean vectors. If the boolean element at index i is true then the corresponding literal i is positive in the model and conversely. We have added a vector of vectors of literals in which the vector element at index i represents the justification of literal i. Next we need to calculate the justification of a literal resulting from a step of the DPLL procedure. A literal resulting from a decision has itself as its justification. If a literal results from a propagation, then its justification is the union of all justifications of all literals used in the propagation step. In the case of backjumping, the justification of the value-switching literal must be recalculated. Thus we are guaranteed that a justification of every literal only contains decision literals.

6.2 Justified SPASS

We have made significant modifications to SPASS without Splitting. They concern justification of clauses, the inference rules Nonground Superposition, Equality Resolution, Equality Factoring, Justified Resolution, and Justified Superposition, and crucially redundancy deletion rules.

Adding a justification to each clause is straightforward. SPASS uses a data structure to encode clauses, we only add a new field for justification to this data structure to have justified clauses.

The Nonground Superposition, Equality Resolution, and Equality Factoring inferences work in the same way as usual Resolution and Superposition inferences, except that they are performed on nonground parts of clauses. We only need to modify the ordered selection in SPASS. Justified Resolution and Justified Superposition are similar to usual Resolution and Superposition combined with the addition of the justification of one of the two premises to the conclusion.

Redundancy deletion rules are tricky. One of our purposes is to reuse existing tools as much as possible to spare implementation time and effort. We select usual deletion rules which could be adapted but still obeying the redundancy criteria of SMELFortunately, S., we could reuse most of the deletion rules of SPASS, provided that the applicability conditions must be re-adapted. Examples of applicability conditions for Subsumption are that the subsuming clause has no justification, or that the justification of the subsuming clause is a subset of the subsumed clause.

7 Experimentation

We have experimented with SMELS on TPTP v4.0.1. The experimentation was performed on a Ubuntu 10.10 Linux machine equipped with an Intel Dual Core E2140, 1.6 Ghz processor and 2 Gb RAThe, M., selected results of the experimentation are reported in Tables 3, 4 and 5. Full results of the experimentation can be found at http://is.hut.edu.vn/ khanhtd/project/smels. In each table, experimental results of SMELS are compared to ones of SPASS 3.7 (without/with Splitting), iProver 0.7 and Darwin 1.4.5. The reasons we have chosen these three theorem provers are that they are implemented based on three of the most well-known first order theorem proving methods: Superposition, Instantiation and Model Evolution. We have chosen SPASS as the Superposition theorem prover because we have implemented SMELS on top

Table 3 Number of successful cases

Division	Size	SMELS	SPASS	SPASS + Splitting	iProver	Darwin
AGT	52	17	17	17	18	18
ALG	392	72	168	211	91	57
CAT	130	45	53	54	51	50
COL	239	127	146	148	100	92
COM	46	30	36	36	34	33
CSR	756	94	141	141	391	330
FLD	279	45	107	107	117	107
GEO	589	320	374	375	378	324
GRA	126	8	15	24	25	21
HAL	9	3	2	2	1	0
HWV	83	61	71	71	59	53
KLE	225	88	106	106	23	19
KRS	271	180	195	197	211	200
LAT	727	90	127	127	99	60
LCL	1079	303	352	355	352	433
MGT	156	110	139	136	152	138
MSC	33	22	23	23	25	24
NLP	520	396	407	431	434	363
NUM	1091	234	349	349	267	175
PRO	72	12	23	22	23	23
PUZ	183	77	87	86	92	92
RNG	261	105	128	131	103	48
SEU	1766	275	378	374	398	166
SWC	846	453	541	565	379	122
SYN	1291	769	890	924	1153	1124

of SPASS, and we need to compare SMELS with a Superposition theorem prover with a similar implementation base.

Table 3 shows the number of successful cases of SMELS and other provers. A case is considered successful if the prover returns the correct result within 180 s. In solving problems SMELS is close to SPASS, iProver and Darwin which are more mature in term of implementation investment. In the COL, HAL, HWV, KLE, RNG and SWC divisions SMELS has more successful cases than iProver and Darwin, and especially SMELS is better than all other theorem provers in the HAL division.

Table 4 shows the number of cases that SMELS can solve but other provers can not. We can see that SMELS can sometimes solve problems within the 180 seconds timeout that SPASS, iProver and Darwin cannot. In the PUZ and SYN divisions those problems have more ground clauses than nonground clauses. In the GRA, HAL, KLE, KRS, LAT, LCL, NUM, RNG, SEU and SWC divisions those problems are mostly equational, and the unsatisfiability is found after only several iterations. In other words, all those problems only require SMELS to perform a few instantiations, and therefore SMELS spends most of the time in ground (equational) reasoning.

Table 5 indicates the number of cases in which SMELS is faster than other provers. SMELS is faster than SPASS on some problems in various divisions. In the AGT, PUZ and SYN divisions those problems have more ground clauses than nonground clauses. In the CSR, GEO, KRS, NLP, NUM, RNG, SEU and SWC divisions those problems are mostly equational, and the unsatisfiability is found after only several

Table 4 Number of cases that SMELS is better than other provers in solving problems

Division	Size	vs. SPASS	vs. SPASS + Splitting	vs. iProver	vs. Darwin
AGT	52	0	0	0	0
ALG	392	0	0	22	27
CAT	130	1	1	4	7
COL	239	0	0	45	40
COM	46	1	1	2	3
CSR	756	0	0	1	0
FLD	279	0	0	0	1
GEO	589	1	2	11	38
GRA	126	2	0	0	0
HAL	9	2	2	2	3
HWV	83	1	1	9	11
KLE	225	2	2	67	69
KRS	271	3	3	3	6
LAT	727	1	1	35	41
LCL	1079	5	4	63	44
MGT	156	0	0	1	9
MSC	33	1	1	0	1
NLP	520	11	0	39	107
NUM	1091	4	4	48	92
PRO	72	1	1	2	2
PUZ	183	1	1	3	2
RNG	261	5	4	43	67
SEU	1766	12	12	32	132
SWC	846	94	69	142	337
SYN	1291	6	2	14	28

Table 5 Number of cases that SMELS is faster than other provers

Division	Size	vs. SPASS	vs. SPASS + Splitting	vs. iProver	vs. Darwin
AGT	52	2	1	0	0
ALG	392	4	4	49	41
CAT	130	1	1	13	11
COL	239	1	1	60	47
COM	46	1	2	6	3
CSR	756	14	13	11	5
FLD	279	2	2	2	5
GEO	589	7	5	33	41
GRA	126	3	1	0	0
HAL	9	2	2	3	3
HWV	83	3	2	25	20
KLE	225	2	2	72	78
KRS	271	6	5	16	8
LAT	727	7	7	53	54
LCL	1079	5	4	112	59
MGT	156	1	1	16	11
MSC	33	1	1	0	1
NLP	520	17	4	131	117
NUM	1091	13	16	99	116
PRO	72	1	1	3	2
PUZ	183	3	3	7	3
RNG	261	8	8	63	77
SEU	1766	34	35	110	153
SWC	846	161	148	318	386
SYN	1291	23	20	54	36

iterations. In the GEO, LAT, LCL and PRO divisions the unsatisfiability is found only after a few iterations. Compared to iProver and Darwin, SMELS is faster on many problems. One of the reasons is that for some of those problems SMELS only requires a few instantiations to detect unsatisfiability by ground (equational) reasoning. Another reason is that SMELS is implemented on top of SPASS and SPASS is already faster than iProver and Darwin on some of those problems.

Overall, SMELS performs fairly well in solving problems in TPTP v4.0.1. In many problems we can see that SMELS is faster than iProver and Darwin, and sometimes SMELS is even faster than SPASS even though it is built on top of SPASSMELS, S., can sometimes solve problems that SPASS, iProver and Darwin cannot.

8 Conclusion

We have presented SMELS, which is a novel complete method for solving satisfiability in first order logic with equality. SMELS inherits the best of the two worlds of SAT solvers and ATPs: a DPLL procedure and a ground reduction algorithm to handle efficiently ground equational clauses; and a complete Superposition procedure to efficiently handle nonground equational clauses using powerful orderings. SMELS has several interesting properties compared to other existing methods. SMELS is complete for First Order Logic with Equality, in contrast with SMT solvers, which use incomplete heuristics to handle quantifiers. SMELS does

not perform any inferences among ground clauses in the Superposition procedure but delegates them to an efficient SAT solver and a reduction algorithm instead. SMELS uses powerful orderings to limit the search space and hence prevent many nonterminating cases. Even though SMELS has been recently implemented, its performance is fairly good, compared to SPASS, iProver and Darwin, which are much more mature systems engineered with much more time and effort.

There are some lines of future work. It is worth to implement and experiment another technique for reducing models computed by the SAT solver. At this point we have used the Rewriting rule of SPASS as the model reduction engine. The idea would be to use a Congruence Closure algorithm, coupled with Justified MiniSat in the DPLL(T) style [16] instead. This method would require us to order terms appropriately in Congruence Closure to be consistent with the ordering used in Justified Superposition. An interesting line of future work is to apply the Schematic Saturation technique in [23] to have an efficient implementation of SMELS. The idea is that in Superposition, we perform unification and inferences over and over again. With Schematic Saturation, we just perform matching to derive new clauses. It would also be interesting to study how SMELS can be used to derive decision procedures for finitely presented theories, along the lines of [1]. Finally, it is worth to study how to integrate a solver for linear arithmetic into SMELS. Although, we know that there exists no complete calculus for the first order theory of linear arithmetic and uninterpreted symbols [20], it is interesting to identify subclasses of formulae which enjoy completeness.

Acknowledgements Some ideas presented in this article were discussed while the first and the last authors were visiting the Automation of Logics research group of Max-Plank-Institute für Informatics. The authors would like to thank Christoph Weidenbach for instructing us on SPASS's code, and for answering all our questions during the implementation of SMELS. The authors would like also to thank the anonymous referees for their fruitful comments and suggestions that helped to improve the clarity of the article.

The authors would like to thank the National Foundation of Science and Technology Development for its support under Grant 102.01.30.09.

References

1. Armando, A., Ranise, S., Rusinowitch, M.: A rewriting approach to satisfiability procedures. Inform. J. Comput. **183**(2), 140–164 (2003)
2. Bachmair, L., Ganzinger, H.: Resolution theorem proving. In: Robinson, A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 1, chap. 2, pp. 19–100. The MIT Press (2001)
3. Bachmair, L., Ganzinger, H., Lynch, C., Snyder, W.: Basic paramodulation and superposition. In: Automated Deduction—CADE-11, 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA. Lecture Notes in Computer Science, vol. 607, pp. 462–476. Springer (1992)
4. Barrett, C., Tinelli, C.: CVC3. In: Damm, W., Hermanns, H. (eds.) Proceedings of the 19th International Conference on Computer Aided Verification (CAV'07), Berlin, Germany. Lecture Notes in Computer Science, vol. 4590, pp. 298–302. Springer (2007)
5. Baumgartner, P., Tinelli, C.: The model evolution calculus as a first-order DPLL method. Artif. Intell. **172**, 591–632 (2008)
6. Bonacina, M.P., Echenim, M.: Theory decision by decomposition. Symb. J. Comput. **45**(2), 229–260 (2010)
7. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. ACM **7**(3), 201–215 (1960)

8. Davis, M., Logemann, G., Loveland, D.W.: A machine program for theorem-proving. Commun. ACM **5**(7), 394–397 (1962)
9. de Moura, L., Bjørner, N.: Engineering dpll(t) + saturation. In: Automated Reasoning, 4th International Joint Conference, Sydney, Australia, 12–1 August 2008. Lecture Notes in Computer Science, vol. 5195, pp. 475–490. Springer (2008)
10. de Moura, L., Bjørner, N.: Z3: : An Efficient SMT Solver. In: Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Budapest, Hungary. Lecture Notes in Computer Science, vol. 4963, pp. 337–340. Springer (2008)
11. Déharbe, D., Ranise, S.: Light-weight theorem proving for debugging and verifying units of code. In: Press, I.C.S. (ed.) Proc. of the Int. Conf. on Software Engineering and Formal Methods (SEFM'03) (2003)
12. Dershowitz, N., Jouannaud, J.P.: Rewrite systems. In: Handbook of Theoretical Computer Science, vol. B, chap. 6, pp. 244–320 (1990)
13. Deshane, T., Hu, W., Jablonski, P., Lin, H., Lynch, C., McGregor, R.E.: Encoding first order proofs in SAT. In: Proceedings Automated Deduction—CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, 17–20 July 2007. Lecture Notes in Computer Science, vol. 4603, pp. 476–491. Springer (2007)
14. Detlefs, D., Nelson, G., Saxe, J.B.: Simplify: a theorem prover for program checking. J. ACM **52**(3), 365–473 (2005)
15. Ganzinger, H., Korovin, K.: New directions in instantiation-based theorem proving. In: Proc. 18th IEEE Symposium on Logic in Computer Science (LICS'03), pp. 55–64. IEEE Computer Society Press (2003)
16. Ganzinger, H., Hagen, G., Nieuwenhuis, R., Oliveras, A., Tinelli, C.: DPLL(T): Fast decision procedures. In: Alur, R., Peled, D. (eds.) Proceedings of the 16th International Conference on Computer Aided Verification (CAV'04), Boston, Massachusetts. Lecture Notes in Computer Science, vol. 3114, pp. 175–188. Springer (2004). ftp://ftp.cs.uiowa.edu/pub/tinelli/papers/GanHNOT-CAV-04.pdf
17. Hooker, J.N., Rago, G., Chandru, V., Shrivastava, A.: Partial instantiation methods for inference in first-order logic. J. Autom. Reasoning **28**(5), 371–396 (2002)
18. Jackson, D.: Automating first-order relational logic. In: SIGSOFT FSE, pp. 130–139 (2000)
19. Korovin, K.: iProver—an instantiation-based theorem prover for first-order logic (system description). In: Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, 12–15 August 2008, Proceedings. Lecture Notes in Computer Science, vol. 5195, pp. 292–298. Springer (2008)
20. Korovin, K., Voronkov, A.: Integrating linear arithmetic into superposition calculus. In: Computer Science Logic (CSL'07). Lecture Notes in Computer Science, vol. 4646, pp. 223–237. Springer (2007)
21. Lee, S.J., Plaisted, D.A.: Eliminating duplication with the hyper-linking strategy. J. Autom. Reasoning **9**(1), 25–42 (1992)
22. Lynch, C., McGregor, R.E.: Combining instance generation and resolution. In: Ghilardi, S., Sebastiani, R. (eds.) Frontiers of Combining Systems, 7th International Symposium, FroCoS 2009, Trento, Italy, 16–18 September 2009. Lecture Notes in Computer Science, vol. 5749, pp. 304–318. Springer (2009)
23. Lynch, C., Ranise, S., Ringeissen, C., Tran, D.K.: Automatic decidability and combinability. Inf. Comput. **209**(7), 1026–1047 (2011)
24. Nieuwenhuis, R., Rubio, A.: Paramodulation-based theorem proving. In: Robinson, A., Voronkov, A. (eds.) Hand of Automated Reasoning. The MIT Press (2001)
25. Riazanov, A., Voronkov, A.: The design and implementation of VAMPIRE. AI Commun. **15**(2), 91–110 (2002)
26. Robinson, J.A.: A machine-oriented logic based on the resolution principle. J. ACM **12**(1), 23–41 (1965)
27. Schulz, S.: E– a brainiac theorem prover. J. AI Commun. **15**(2/3), 111–126 (2002)
28. Strichman, O., Seshia, S.A., Bryant, R.E.: Deciding separation formulas with sat. In: Brinksma, E., Larsen, K.G. (eds.) Computer Aided Verification, 14th International Conference, CAV 2002, Proceedings, Copenhagen, Denmark, 27–31 July 2002. Lecture Notes in Computer Science, vol. 2404, pp. 209–222. Springer (2002)
29. Sutcliffe, G.: The cade-22 automated theorem proving system competition—CASC-22. AI Commun. **23**(1), 47–59 (2010)
30. Weidenbach, C.: Spass version 0.49. J. Autom. Reasoning **14**(2), 247–252 (1997)