Integrating equational reasoning into instantiation-based theorem proving

Harald Ganzinger and Konstantin Korovin

MPI für Informatik, Saarbrücken

Abstract. In this paper we present a method for integrating equational reasoning into instantiation-based theorem proving. The method employs a satisfiability solver for ground equational clauses together with an instance generation process based on an ordered paramodulation type calculus for literals. The completeness of the procedure is proved using the the model generation technique, which allows us to justify redundancy elimination based on appropriate orderings.

1 Introduction

The basic idea of instantiation-based theorem proving is to combine clever generation of instances of clauses with propositional satisfiability checking. Thus, it seems to be promising to exploit the impressive performance of modern propositional SAT technology in the more general context of first-order theorem proving. Accordingly, we have seen several attempts recently at designing new first-order prover architectures combining efficient propositional reasoning into instance generation scenarios, cf. [4, 11, 5, 15, 3, 12, 9, 13] among others.

Integration of efficient equational reasoning into such systems has been a challenging problem, important for many practical applications. In this paper we show how to integrate equational reasoning into the instantiation framework developed in [8]. In [8] we presented instance generation inference systems based on selection from propositional models, together with a notion of redundancy based on closure orderings, and showed their refutational completeness.

Our approach of integrating equational reasoning into this framework aims to preserve attractive properties of the instantiation process, in particular:

1. no recombination of clauses,
2. the length of clauses does not grow,
3. optimal efficiency in the ground case,
4. semantic selection,
5. redundancy criteria.

As in our previous work, we will use in a modular fashion a satisfiability solver for ground clauses. Let us remark that in the presence of equality such solvers have received considerable attention and very efficient implementations are available, see e.g., [7]. We also use selection based on models of ground clauses to guide the theorem proving process. Another ingredient of our procedure is a paramodulation-based calculus for reasoning with sets of literals. As we show, with the help of such a calculus it is possible to generate suitable instances of clauses, witnessing unsatisfiability of the selected literals on the ground level. For the completeness proofs we use the model generation technique (see [1, 2, 14]) which allows us to justify redundancy elimination based on entailment from smaller clauses, where "smaller" refers to suitable closure orderings.

Let us briefly compare our method with two other approaches, we aware of, that deal with equational reasoning in the context of instantiation-based theorem proving. In [13] and [16] an equational version of the disconnection calculus is presented, however in their method, literals from different clauses are recombined into a new clause when (superposition-type) equational steps are done. Our instance generation inference systems entirely avoid that recombination which, according to [11], can be a major source of inefficiency in resolution- and superposition-type inference systems. In [15], Plaisted and Zhu consider an extension of their OSHL calculus with equality. It is based on paramodulation with unit clauses, but for non-unit clauses it requires the generally less efficient Brand's transformation method. Our method is applicable for arbitrary first-order clauses with equality.

2 Preliminaries

We shall use standard terminology for first-order clause logic with equality. The symbol "$\simeq$" is used to denote formal equality. By "$\models$" we denote entailment in first-order logic with equality. A *clause* is a possibly empty multiset of literals L_i, usually written $L_1 \vee \ldots \vee L_n$; a *literal* being either an equation $s \simeq t$ or a disequations $u \not\simeq v$ built from terms s, t, u, and v over the given signature. We consider $\simeq$ (and $\not\simeq$) as symmetric syntactically, identifying $s \simeq t$ with $t \simeq s$. We say that C is a sub-clause of D, and write $C \subseteq D$, if C is a sub-multiset of D. The empty clause, denoted by $\square$, denotes falsum. If L is a literal, $\overline{L}$ denotes the complement of L.

A substitution is called a *proper instantiator* of an expression (a term, literal, or clause) if at least one variable of the expression is mapped to a non-variable term. We call D *more specific* than C if $C\tau = D$ for some proper instantiator τ of C. *Renamings* are injective substitutions, sending variables to variables. Two clauses are *variants* of each other if one can be obtained from the other by applying a renaming.

Instance-based theorem proving requires us to work with a refined notion of instances of clauses that we call closures. A *closure* is a pair consisting of a clause C and a substitution σ written $C \cdot \sigma$. We work modulo renaming, that is, do not distinguish between closures $C \cdot \sigma$ and $D \cdot \tau$ for which C is a variant of D and $C\sigma$ is a variant of $D\tau$. Note the distinction between the two notations $C\sigma$ and $C \cdot \sigma$. The latter is a closure *representing* the former which is a clause. A closure is called *ground* if it represents a ground clause. A (ground) closure $C \cdot \sigma$ is called a *(ground) instance* of a set of clauses S if C is a clause in S, and then we say that the closure $C \cdot \sigma$ is a *representation (of the clause $C\sigma$) in S*.

Inference systems and completeness proofs will be based on orderings on ground clauses and closures. Let $\succ_{gr}$ be a total simplification ordering on ground terms. We can assume that $\succ_{gr}$ is defined on ground clauses by a total, well-founded and monotone extension of the order from terms to clauses, as defined, e.g., in [14]. We will extend $\succ_{gr}$ in two ways to orderings on ground closures. The first ordering is $\succ_l$, defined in Section 4, and will be used in reasoning with the unit paramodulation calculus UP. The second is $\succ_{cl}$, defined in Section 5, and will be used in reasoning about instantiations of clauses.

The (Herbrand) *interpretations* we deal with are sometimes partial, given by consistent sets I of ground literals. (As usual, I is called *consistent* if, and only if, $I \not\models \square$.) A ground literal L is called *undefined* in I if neither $I \models L$ nor $I \models \overline{L}$. I is called *total* if no ground literal is undefined in I. A ground clause C is called *true* (or valid) in a partial interpretation I if $I \models C$. This is the same as saying that $J \models C$ for each total extension J of I. C is called *false* in I if $I \models \neg C$, or, equivalently, if $J \not\models C$ for each total extension J of I. Truth values for closures are defined from the truth values of the clauses they represent.

3 An informal description of the procedure

Let us first informally describe our instantiation-based inference process for equational reasoning. We assume that a satisfiability solver for ground equational clauses is given.

Let S be a given set of first-order clauses. We start by mapping all variables in all clauses in S to a distinguished constant $\bot$, obtaining a set of ground clauses $S\bot$. If $S\bot$ is unsatisfiable then S is first-order unsatisfiable and we are done. Otherwise, we non-deterministically select a literal from each clause in S, obtaining a set of literals Lit.

The next natural step, similar to the case without equality, would be to consider applicable ordered paramodulation inferences, but instead of generating paramodulants to generate corresponding instances of clauses. Unfortunately, adding these instances is not sufficient for a solver on ground clauses to detect unsatisfiability, as the following example shows. Consider the unsatisfiable set of literals $S = \{f(h(x)) \simeq c, h(y) \simeq y, f(a) \not\simeq c\}$. The only applicable ordered paramodulation inference is between the first and the second equation, but the resulting instances are the given equations themselves. On the other hand, the set of ground literals $S\bot$ is satisfiable, so a solver for ground literals can not detect the unsatisfiability of S.

Our approach to this problem is to apply separate first-order reasoning with the selected literals Lit. If Lit is first-order satisfiable, then S is satisfiable and we are done. Otherwise, we generate relevant instances of clauses from S witnessing unsatisfiability of Lit at the ground level. This is done using a paramodulation-based system on literals. In particular, relevant instances can be generated by propagating substitutions from proofs of the empty clause in such a system. Finally, we add obtained instances to S, and repeat the procedure.

Let us now modify the previous example and assume that the literals above are among the selected ones in some clauses, e.g.,

$$S = \{f(h(x)) \simeq c \vee h(h(x)) \not\simeq a,\ h(y) \simeq y,\ f(a) \not\simeq c\}$$
$$Lit = \{f(h(x)) \simeq c,\ h(y) \simeq y,\ f(a) \not\simeq c\}.$$

We can derive the empty clause from Lit by first paramodulating the second literal into the first literal, followed by paramodulation of the result into the third literal. Now, from this paramodulation proof we can extract a relevant substitution σ, which maps x and y to a. Then, the new set of clauses is obtained by applying σ to the old clauses: $S' = S \cup \{f(h(a)) \simeq c \vee h(h(a)) \not\simeq a, h(a) \simeq a\}$. Now, the set $S'\bot$ can be shown to be unsatisfiable by a solver for ground clauses, so we conclude that the original set S is first-order unsatisfiable. In the case if $S'\bot$ is satisfiable we would continue the procedure with this new set of clauses. Let us note that usually the search for the proof of the empty clause from the set of literals is done via some kind of saturation process which can generate a lot of inferences. But the proof itself usually involves only some of them, and as we have seen, we need to propagate only substitutions used in the proof.

We use a solver for ground clauses not only for testing unsatisfiability of $S\bot$. In addition, in the case of satisfiable $S\bot$, the instantiation process can be guided by a model $I_\bot$ of $S\bot$. For this, we restrict the selection of literals to the literals L, such that $L\bot$ is true in $I_\bot$.

Now we overview how we are going to prove completeness of such instantiation process. First, in Section 4 we introduce a calculus UP for ground closures of literals based on ordered paramodulation. We will use this calculus to obtain relevant instantiations of clauses. Then, in Section 5 we show that if a set of clauses is saturated enough, then either it is satisfiable, or otherwise its unsatisfiability can be detected by a ground solver. In the subsequent Section 6 we show

how to obtain a saturated set as a limit of a fair saturation process. The problem of how to ensure that a saturation process is fair is considered in Section 7. Up to this point we are working with the UP calculus defined on ground closures. Ground closures allow us to present completeness proofs and fine grained notions of redundancy. Nevertheless, from the practical point of view it is infeasible to work with each ground closure separately, and therefore in Section 8 we present the UPL calculus which is a lifted version of UP. Finally, in Section 9 we consider the issue of how to propagate information on redundant closures to the UPL calculus. This is done via dismatching constraints.

4 Unit paramodulation on literal closures

In this section we introduce an inference system on ground closures of literals, based on ordered paramodulation. This system (and its lifted versions) will be used to guide our instantiation process as shown in the following sections.

Unit-Paramodulation calculus (UP)

$$\frac{(l \simeq r)\cdot\sigma \quad L[l']\cdot\sigma'}{L[r]\theta\cdot\rho}\ (\theta) \qquad\qquad \frac{(s \not\simeq t)\cdot\tau}{\square}\ (\mu)$$

where (i) $l\sigma \succ_{gr} r\sigma$; (ii) $\theta = mgu(l,l')$; (iii) $l\sigma = l'\sigma' = l'\theta\rho$; (iv) l' is not a variable.

where (i) $s\tau = t\tau$; (ii) $\mu = mgu(s,t)$.

An inference in UP is called proper if the substitution θ, (μ) is a proper instantiator and non-proper otherwise. Let us note that a set of literal closures can be contradictory, yet the empty clause is not derivable in UP.

Example 1. Consider a set of literal closures $\mathcal{L} = \{(f(x) \simeq b)\cdot[a/x],\ a \simeq b,\ f(b) \not\simeq b\}$ and assume that $a \succ_{gr} b$. Then, $\mathcal{L}$ is inconsistent but the empty clause is not derivable by UP from $\mathcal{L}$.

UP-*redundancy.* Let R be an arbitrary ground rewrite system and $\mathcal{L}$ be a set of literal closures, we denote $irred_R(\mathcal{L})$ the set of closures $L\cdot\sigma \in \mathcal{L}$ with irreducible σ w.r.t. R. In order to introduce the notion of UP-redundancy we need the following ordering on literal closures. Let $\succ_l$ be an arbitrary total well-founded extension of $\succ_{gr}$ from ground literals to ground closures of literals, such that if $L\sigma \succ_{gr} L'\sigma'$ then $L\cdot\sigma \succ_l L'\cdot\sigma'$.

Let $\mathcal{L}$ be a set of literal closures. We say that $L\cdot\sigma$ is UP-*redundant* in $\mathcal{L}$ if for every ground rewrite system R oriented by $\succ_{gr}$, and such that σ is irreducible w.r.t. R we have $R \cup irred_R(\mathcal{L}_{L\cdot\sigma\succ_l}) \models L\sigma$. Here, $\mathcal{L}_{L\cdot\sigma\succ_l}$ denotes the set of all closures in $\mathcal{L}$ less than $L\cdot\sigma$ w.r.t. $\succ_l$. We denote the set of all UP-redundant closures in $\mathcal{L}$ as $\mathcal{R}_{\mathrm{UP}}(\mathcal{L})$. With the help of this redundancy notion we can justify the following simplification rule.

Non-proper Demodulation

$$\frac{(l \simeq r) \cdot \sigma \quad L[l'] \cdot \sigma'}{L[r]\theta \cdot \sigma'}$$

where (i) $l' = l\theta$, (ii) $l\sigma \succ_{gr} r\sigma$, (iii) θ is a non-proper instantiator, (iv) $l\sigma = l'\sigma'$, (v) $Var(r) \subseteq Var(l)$, (vi) $L[l']\sigma' \succ_{gr} (l \simeq r)\sigma$.

Let us show that non-proper demodulation is a simplification rule, i.e., after adding the conclusion of this rule the right premise becomes UP-redundant.

Lemma 1. *Non-proper demodulation is a simplification rule.*

Proof. Indeed, let $L[r]\theta \cdot \sigma'$ be the conclusion of an application of the non-proper demodulation rule with the premise $(l \simeq r) \cdot \sigma$, $L[l'] \cdot \sigma'$. Now let R be a rewrite system orientable by $\succ_{gr}$ such that σ' is irreducible w.r.t. R. Since θ is a non-proper instantiator, $l' = l\theta$, and $Var(r) \subseteq Var(l)$ we have that σ is also irreducible. Therefore, $L[l']\sigma'$ follows from the smaller closures $(l \simeq r) \cdot \sigma$ and $L[r]\theta \cdot \sigma'$.

Let us show that demodulation with proper unifiers can not be used as a simplification rule in general.

Example 2. Consider the following closures.

$$\begin{array}{ll} (1)\ (g(x) \simeq c) \cdot [f(d)/x] & (3)\ (f(d) \simeq m) \cdot [] \\ (2)\ (g(f(x)) \simeq c) \cdot [d/x] & (4)\ (g(m) \not\simeq c) \cdot [] \end{array}$$

We can derive the empty clause by UP inferences from (2), (3) and (4). But if we simplify (2) by demodulation with (1) we obtain a tautological closure and the empty clause would not be derivable by UP. The reason for this is that the substitution $[f(d)/x]$ in (1) is reducible (by (3)), whereas the substitution $[d/x]$ in (2) is not.

An UP-*saturation process* is a finite or infinite sequence of sets of closures of literals $\{\mathcal{L}_i\}_{i=1}^{\infty}$ where each set $\mathcal{L}_i$ is obtained from $\mathcal{L}_{i-1}$ by either adding a conclusion of an UP-inference with premises from $\mathcal{L}_{i-1}$ or by removing an UP-redundant w.r.t. $\mathcal{L}_{i-1}$ closure. Let us denote by $\mathcal{L}^{\infty}$ the set of *persisting closures*, that is, the lower limit of the sequence $\mathcal{L}_i$. An UP-saturation process $\{\mathcal{L}_i\}_{i=1}^{\infty}$ is called UP-*fair* if for every UP-inference with premises in $\mathcal{L}^{\infty}$, the conclusion is UP-redundant w.r.t. $\mathcal{L}_i$ for some i.

For simplicity, with each set of literal closures $\mathcal{L}$ we associate an arbitrary but fixed UP-fair saturation process $\{\mathcal{L}_i\}_{i=1}^{\infty}$, where $\mathcal{L} = \mathcal{L}_1$. Below, $\mathcal{L}^{sat}$ will always denote the set $\mathcal{L}^{\infty} \setminus \mathcal{R}_{\mathrm{UP}}(\mathcal{L}^{\infty})$, which we call the UP-saturation of $\mathcal{L}$.

5 Completeness for saturated sets of clauses

In this section we prove that if a set of clauses S is saturated enough, then either it can be shown to be unsatisfiable by a ground solver, i.e., $S\bot$ is unsatisfiable, or otherwise S is first-order satisfiable. In the later sections we show how to achieve saturated sets.

First we introduce the notion of Inst-redundancy which will be used to extract relevant closures form clause sets, and also to measure progress in the instantiation process. For this we extend the order $\succ_{gr}$ from ground clauses to ground closures as follows. We say that $C \cdot \tau \succ'_{cl} D \cdot \rho$ if either $C\tau \succ_{gr} D\rho$ or $C\tau = D\rho$ and $C\theta = D$ for a proper instantiator θ. It is straightforward to see that $\succ'_{cl}$ is a well-founded order, so we define $\succ_{cl}$ to be any total well-founded extension of $\succ'_{cl}$.

Let S be a set of clauses and C a ground closure. C is called Inst-*redundant* in S if there exist closures $C_1, \ldots, C_k$ that are ground instances of S such that, (i) for each i, $C \succ_{cl} C_i$, and (ii) $C_1, \ldots, C_k \models C$. A clause C (possibly non-ground) is called Inst-redundant in S if each ground closure $C \cdot \sigma$ is Inst-redundant in S. We denote the set of Inst-redundant closures in S as $\mathcal{R}_{\mathrm{Inst}}(S)$.

Consider a set of clauses S, a model $I_\bot$ of $S\bot$. A *selection function* sel based on $I_\bot$ is a function mapping clauses to literals such that for each $C \in S$, $\mathsf{sel}(C) \in C$ and $\mathsf{sel}(C)\bot$ is true in $I_\bot$. Let us consider a selection function sel based on I. Define a set of S-*relevant* instances of literals $\mathcal{L}_S$ as the set of all literal closures $L \cdot \sigma$ such that

1. $L \vee C \in S$,
2. $(L \vee C) \cdot \sigma$ is not Inst-redundant in S,
3. $L = \mathsf{sel}(L \vee C)$.

Let $\mathcal{L}_S^{sat}$ denote the UP-saturation of $\mathcal{L}_S$ (see Section 4). We say that the set of clauses S is Inst-*saturated* w.r.t. a selection function sel, if $\mathcal{L}_S^{sat}$ does not contain the empty clause.

The following auxiliary lemma about UP is obvious.

Lemma 2. *Let R be a ground rewrite system and* UP *is applicable to $(l \simeq r)\cdot\sigma$, $L[l'] \cdot \sigma'$ with the conclusion $L[r]\theta \cdot \rho$. Then if σ and σ' are irreducible w.r.t. R then ρ is also irreducible.*

Now we are ready to prove our main completeness theorem. Let us remark that in the proof we will use both orderings $\succ_l$ and $\succ_{cl}$. In fact, the model construction will be done in $\succ_l$, but the counterexample reduction in $\succ_{cl}$.

Theorem 1. *If a set S of clauses is* Inst-*saturated, and $S\bot$ is satisfiable, then S is also satisfiable.*

Proof. Let S be an Inst-saturated set of clauses, such that $S\bot$ is satisfiable in a model $I_\bot$, and sel is a selection function based on $I_\bot$. Let $\mathcal{L}_S$ be S-relevant instances of literals. We have that $\mathcal{L}_S^{sat}$ does not contain the empty clause.

By induction on $\succ_l$ we construct a candidate model to S based on $\mathcal{L}_S^{sat}$. Suppose, as an induction hypothesis, that sets of literals ϵ_M have been defined for the ground closures $M \in \mathcal{L}_S^{sat}$ smaller than L in $\succ_l$, and let I_L denote the set $\bigcup_{L \succ_l M} \epsilon_M$. Let R_L denote the ground rewrite system obtained by orienting all positive equations in I_L w.r.t. $\succ_l$. Suppose that $L = L' \cdot \sigma$. Then define $\epsilon_L = \{L'\sigma\}$, if

1. $L'\sigma$ is irreducible by R_L, and
2. $L'\sigma$ is undefined in I_L (i.e. neither $I_L \models L'\sigma$ nor $I_L \models \overline{L}'\sigma$).

In this case we say that L is *productive*. Otherwise, we define $\epsilon_L = \emptyset$. Define I_S to be the set $\bigcup_{L \in \mathcal{L}_S^{sat}} \epsilon_L$ and $R_S = \bigcup_{L \in \mathcal{L}_S^{sat}} R_L$. It is easy to see that I_S is consistent and R_S is a convergent interreduced rewrite system and every $L\sigma \in I_S$ is irreducible by R_S. Let I be an arbitrary total consistent extension of I_S. Now we show that I is a model to all ground instances of S. Assume otherwise.

Let $D = D' \cdot \sigma$ be the minimal w.r.t. $\succ_{cl}$ ground instance of S that is false in I. Let us show that for every variable x in D', $x\sigma$ is irreducible by R_S. Otherwise, let $(l \to r)\tau \in R_L$ and $x\sigma = x\sigma[l\tau]_p$ for some variable x in D'. Then, we can define a substitution σ' by changing σ on x with $x\sigma' = x\sigma[r\tau]_p$. We have that $I \not\models D'\sigma'$ and $D \succ_{cl} D' \cdot \sigma'$, which contradicts to the minimality of the counterexample.

Now we note that D is not Inst-redundant in S. Otherwise it would follow from smaller, w.r.t. $\succ_{cl}$, closures $D_1, \ldots, D_n$. Hence, one of D_i is false in I contradicting to the minimality of the counterexample.

Since D is not Inst-redundant, we have that for some literal L, $D' = L \vee D''$ and $L \cdot \sigma \in \mathcal{L}_S$. And also $L\sigma$ is false in I.

Assume that $L \cdot \sigma$ is UP-redundant in $\mathcal{L}_S^{sat}$. Then, since σ is irreducible by R_S we have

$$R_S \cup \mathit{irred}_{R_S}(\{L' \cdot \sigma' \in \mathcal{L}_S^{sat} | L \cdot \sigma \succ_l L' \cdot \sigma'\}) \models L\sigma.$$

Therefore, there is $L' \cdot \sigma \in \mathit{irred}_{R_S}(\mathcal{L}_S^{sat})$ false in I, (if $L \cdot \sigma$ is not UP-redundant in $\mathcal{L}_S^{sat}$ we take $L' \cdot \sigma = L \cdot \sigma$). Let $M \cdot \tau$ be the minimal w.r.t. $\succ_l$ closure in $\mathit{irred}_{R_S}(\mathcal{L}_S^{sat})$ which is false in I. Let us show that $M \cdot \tau$ is irreducible by R_S. Otherwise, assume that $M \cdot \tau$ is reducible by $l \to r \in R_S$ and $(l' \to r') \cdot \rho \in \mathcal{L}_S^{sat}$ is the closure producing $l \to r$ in R_S. Since τ is irreducible by R_S, UP-inference is applicable to $(l' \to r') \cdot \rho$ and $M[l''] \cdot \tau$ with the conclusion $M[r']\theta \cdot \mu$, where $l'\rho = l''\tau = l''\theta\mu$ and $\theta = \mathit{mgu}(l', l'')$. We have that $M[r']\theta \cdot \mu$

is false in I. Now we show that $M[r']\theta \cdot \mu$ is not UP-redundant in $\mathcal{L}_S^{sat}$. Assume otherwise. From Lemma 2 follows that μ is irreducible by R_S. From definition of UP-redundancy, we have

$$R_S \cup \mathit{irred}_{R_S}(\{M' \cdot \tau' \in \mathcal{L}_S^{sat} | M[r']\theta \cdot \mu \succ_l M' \cdot \tau'\}) \models M[r']\theta\mu.$$

Therefore, there is $M' \cdot \tau' \in \mathit{irred}_{R_S}(\mathcal{L}_S^{sat})$ such that $M \cdot \tau \succ_l M[r']\theta \cdot \mu \succ_l M' \cdot \tau'$ and $M'\tau'$ false in I. This contradicts to the minimality of $M \cdot \tau$. But, if $M[r']\theta \cdot \mu$ is not UP-redundant we have $M[r']\theta \cdot \mu \in \mathcal{L}_S^{sat}$, and since μ is irreducible by R_S, $M[r']\theta \cdot \mu \in \mathit{irred}_{R_S}(\mathcal{L}_S^{sat})$, we again obtain a contradiction to the minimality of $M \cdot \tau$. We conclude that $M \cdot \tau$ is irreducible by R_S.

Now we have that $M \cdot \tau$ is in $\mathcal{L}_S^{sat}$, irreducible by R_S, and not productive. Therefore $I_{M \cdot \tau} \models \overline{M}\tau$. Consider all possible cases. Let $M \cdot \tau$ be an equation $(s \simeq t) \cdot \tau$. We have that $I_{M \cdot \tau} \models (s \not\simeq t)\tau$. Since, all literals in $I_{M \cdot \tau}$ and $s\tau, t\tau$ are irreducible by $R_{M \cdot \tau}$, and $R_{M \cdot \tau}$ is a convergent rewrite system we have $(s \not\simeq t)\tau \in I_{M \cdot \tau}$. Therefore $(s \not\simeq t)\tau$ is produced to $I_{M \cdot \tau}$ by some $(s' \not\simeq t') \cdot \tau'$. But this is impossible since $(s' \not\simeq t')\tau' \succ_{gr} (s \simeq t)\tau = M\tau$, and hence $(s' \not\simeq t') \cdot \tau' \succ_l M \cdot \tau$. Now assume that $M \cdot \tau$ is a disequation $(s \not\simeq t) \cdot \tau$. We have $I_{M \cdot \tau} \models (s \simeq t)\tau$ and since $s\tau$ and $t\tau$ are irreducible by $R_{M \cdot \tau}$ we have $s\tau = t\tau$. But then equality resolution is applicable to $M \cdot \tau$, contradicting that $\mathcal{L}_S^{sat}$ does not contain the empty clause.

Finally we conclude that I is an model for S.

6 Effective Saturation Strategies

In this section we shall investigate how saturation of a set of clauses can be achieved effectively. First we show how saturation is done on closures and later we show how saturation process can be lifted to general clauses.

An Inst-*saturation process* is a sequence of triples $\{\langle S^i, I_\perp^i, \mathsf{sel}^i \rangle\}_{i=1}^\infty$, where S^i is a set of clauses, $I_\perp^i$ a model of $S^i\perp$ and sel^i a selection function based on that model. Given $\langle S^i, I_\perp^i, \mathsf{sel}^i \rangle$, a *successor state* $\langle S^{i+1}, I_\perp^{i+1}, \mathsf{sel}^{i+1} \rangle$ is obtained by one of these steps: (i) $S^{i+1} = S^i \cup N$, where N is a set of clauses such that $S^i \models N$; or (ii) $S^{i+1} = S^i \setminus \{C\}$, where C is Inst-redundant in S^i. If $S^{i+1}\perp$ is unsatisfiable, the process terminates with the result "unsatisfiable". Let us denote by S^∞ the set of persisting clauses, that is, the lower limit of $\{S^i\}_{i=1}^\infty$. In order to ensure that we always reach an Inst-saturated set in the limit of the saturation process we need the notion of Inst-fair saturation.

Consider a finite set of closures $K = \{(L_1 \vee C_1) \cdot \sigma, \ldots, (L_n \vee C_n) \cdot \sigma\}$ of clauses from S^∞. We denote $\mathcal{L} = \{L_1 \cdot \sigma, \ldots, L_n \cdot \sigma\}$. The pair $(K, \mathcal{L})$ is called a *persistent conflict* if $\mathcal{L}^{sat}$ contains the empty clause and for infinitely many i we have $\mathsf{sel}^i(L_j \vee C_j) = L_j$ for $1 \leq j \leq n$.

We call an Inst-saturation process Inst-*fair* if for every persistent conflict $(K, \mathcal{L})$, at least one of the closures in K is Inst-redundant in S_i for some i.

Now our goal is to show that for the limit S^∞ of an Inst-fair saturation process, such that $S^\infty\bot$ is satisfiable, we can build a model $I_\bot$ and a selection function sel, based on $I_\bot$ such that S^∞ is Inst-saturated w.r.t. sel. The main problem here is that when we use selection functions based on truth in propositional models, these models change when we add more instances. Note that it is possible that the limit S^∞ of an Inst-fair saturation process is not Inst-saturated for some model $I_\bot$ of $S^\infty\bot$, likewise it is possible that $I^i_\bot \cup I^j_\bot$ is inconsistent for every $i \neq j$ (so, for example, we can not take union of $I^i_\bot$ for $I_\bot$).

Lemma 3. *Let S^∞ be a set of persistent clauses of an* Inst-*fair saturation process* $\{\langle S^i, I^i_\bot, \mathsf{sel}^i\rangle\}_{i=1}^\infty$, *and $S^\infty\bot$ is satisfiable. Then, there exists a model $I\bot$ of $S^\infty\bot$ and a selection function* sel *based on $I\bot$ such that S^∞ is* Inst-*saturated w.r.t.* sel.

Proof. Let $\{C_i\}_{i=1}^\infty$ be an enumeration of clauses in S^∞. For each n we construct a model J^n of $\{C_i\bot\}_{i=1}^{i=n}$ and a selection function sel^n_J based on J^n, by induction on n. For each n the following invariants will be satisfied.

1. J^n is consistent and sel^n_J is a selection function for clauses $\{C_i\}_{i=1}^{i=n}$ based on J^n.
2. $J^{n-1} \subseteq J^n$ and sel^n_J coincides with sel^{n-1}_J on clauses $\{C_i\}_{i=1}^{i=n-1}$.
3. There are infinitely many k such that $J^n \subseteq I^k$ and for all $1 \leq l \leq n$, $\mathsf{sel}^k(C_l) = \mathsf{sel}^n_J(C_l)$.

If $n = 1$ then we have that there exists $L \in C_1$ such that $L \in \mathsf{sel}^k$ for infinitely many k. We take $J^1 = \{L\bot\}$ and $\mathsf{sel}^1_J(C_1) = \{L\}$. Trivially, all invariants (1–3) are satisfied.

Let $n \geq 1$ and assume that we have a model J^n and sel^n_J for $\{C_i\bot\}_{i=1}^{i=n}$ such that invariants (1–3) are satisfied. Since $C_{n+1} \in S^\infty$ we have that for some m and every $p \geq m$, $C_{n+1} \in S^p$. From this and invariant (3) follows that for some $L \in C_{n+1}$ there are infinitely many k such that $J^n \subseteq I^k$, and $\mathsf{sel}^k(C_l) = \mathsf{sel}^n_J(C_l)$ for all $1 \leq l \leq n$, and $\mathsf{sel}^k(C_{n+1}) = L$. Define $J^{n+1} = J^n \cup \{L\bot\}$ and $\mathsf{sel}^{n+1}_J(C_l) = \mathsf{sel}^n_J(C_l)$ for $1 \leq l \leq n$, $\mathsf{sel}^{n+1}_J(C_{n+1}) = L$. It is easy to see that all invariants (1–3) are satisfied for $J^{n+1}, \mathsf{sel}^{n+1}_J$.

We define $I_\bot = \cup_{i=1}^\infty J_i$ and $\mathsf{sel}(C_i) = \mathsf{sel}^i_J(C_i)$ for $i \geq 1$. From compactness follows that $I_\bot$ is consistent, and sel is a selection function based on $I_\bot$.

Now we need to show that S^∞ is saturated w.r.t. sel. Assume otherwise. Then, there is a finite subset $\mathcal{L}$ of $\mathcal{L}_{S^\infty}$, such that $\mathcal{L}^{sat}$ contains the empty clause. Let $K = \{(L_1 \vee C_1)\cdot\sigma, \ldots, (L_n \vee C_n)\cdot\sigma\}$ be the set of closures of clauses from

S^{∞}, producing $\mathcal{L}$ to $\mathcal{L}_{S^{\infty}}$. Then, from the construction of $I_{\perp}$ and in particular from the invariant (3) follows that there are infinitely many i such that $\mathsf{sel}^i(L_j \vee C_j) = L_j$ for $1 \leq j \leq n$. Hence, $(K, \mathcal{L})$ is a persistent conflict. Since the saturation process is Inst-fair we have that at least one of the closures in K is Inst-redundant in $\mathcal{L}_{S^{\infty}}$. But this is impossible since all closures in $\mathcal{L}$ are S^{∞}-relevant and can not be produced by Inst-redundant closures.

Corollary 1. *Let $\{\langle S^i, I^i_{\perp}, \mathsf{sel}^i\rangle\}_{i=1}^{\infty}$ be an* Inst*-fair saturation process. Then, either (1) for some i we obtain an unsatisfiable $S^i\perp$ and therefore S^1 is unsatisfiable, or (2) for all i, $S^i\perp$ is satisfiable and therefore, (by Lemma 3 and Theorem 1) S^1 is satisfiable, moreover if for some i, S^i is* Inst*-saturated then at this step we can conclude that S^1 is satisfiable.*

In the next sections we consider the issue of how to ensure that an Inst-saturation process is Inst-fair.

7 Relevant instances from proofs

In order to obtain an Inst-fair saturation we need to make closures in persistent conflicts Inst-redundant. A uniform way to make a closure $C \cdot \sigma$ of a clause $C \in S$, Inst-redundant in S, is to add to S a proper (possible nonground) instance of C, which generalises $C\sigma$. Next we will study how to find instantiations which are relevant to the persisting conflicts.

Let us consider a persistent conflict $(K, \mathcal{L})$, where $K = \{(L_1 \vee C_1) \cdot \sigma, \ldots, (L_n \vee C_n) \cdot \sigma\}$ and $\mathcal{L} = \{L_1 \cdot \sigma, \ldots, L_n \cdot \sigma\}$. Since $\mathcal{L}^{sat}$ contains the empty clause we have that there is a proof of the empty clause in UP from closures in $\mathcal{L}$. Our next goal is to show that in any proof at least one inference is a proper UP-inference. To speak more formally about the proofs we assume that proofs are represented as binary trees with nodes labelled by closures together with substitutions from the corresponding inferences. We assume that at each node of a proof, left subproof is variable disjoint from the right subproof.

Example 3. A proof of the empty clause in UP from literal closures.

$$\dfrac{\dfrac{\dfrac{f(x) \simeq g(x) \cdot [h(a)/x] \quad f(y) \simeq h(y) \cdot [h(a)/y]}{g(x) \simeq h(x) \cdot [h(a)/x]}\,[x/y] \qquad g(h(u)) \not\simeq h(h(u)) \cdot [a/u]}{h(h(u)) \not\simeq h(h(u)) \cdot [a/u]}\,[h(u)/x]}{\Box}\,[]$$

Let us consider a proof P and a leaf of this proof with a closure $L \cdot \sigma$. Let $\theta_1, \ldots, \theta_n$ be substitutions along the branch from this leaf to the root of the proof. We call the composition $\theta = \theta_1 \cdots \theta_n$ as a *P-relevant instantiator* and the closure $L\theta \cdot \tau$ as a *P-relevant instance* of $L \cdot \sigma$, where $L\theta\tau = L\sigma$. If we consider the left most leaf of the proof in the example above, then the P-relevant instance will be $(f(h(u)) \simeq g(h(u))) \cdot [a/u]$ with the P-relevant instantiator $[h(u)/x]$.

Lemma 4. *Let P be a proof of the empty clause, and PI be the set of P-relevant instances of all leafs of P. Then, $PI\bot$ is unsatisfiable.*

Corollary 2. *Let $(K, \mathcal{L})$ be a persistent conflict and P is a proof of the empty clause from $\mathcal{L}$ in* UP*, then at least one of P-relevant instantiator is proper.*

For a persistent conflict $(K, \mathcal{L})$, this corollary allows us to make closures in K Inst-redundant by adding their P-relevant proper instances. Let us continue with Example 3. Assume that literal closures at the leafs of P are in $\mathcal{L}$ for a persistent conflict $(K, \mathcal{L})$, so $\mathcal{L} = \{(f(x) \simeq g(x)) \cdot [h(a)/x], \ldots\}$ and, e.g., $K = \{(f(x) \simeq g(x) \vee h(g(x)) \simeq c) \cdot [h(a)/x], \ldots\}$, then we can add a P-relevant proper instance $f(h(u)) \simeq g(h(u)) \vee h(g(h(u))) \simeq c$ to the clause set, making the first closure in K Inst-redundant. Thus, to make an Inst-saturation process Inst-fair we need to UP-saturate literal closures from the persisting conflicts and add proper instantiations of clauses with substitutions that can be obtained from the proofs of the empty clause.

8 From literal closures to literal clauses

So far we have been considering closures as the basic entities for persistent conflicts and the UP calculus. Of course, working with each ground closure separately is of little practical use. This motivates our next step of lifting UP calculus from literal closures to literals.

Unit paramodulation for literals (UPL)

$$\frac{(l \simeq r) \quad L[l']}{L[r]\theta}\ (\theta) \qquad\qquad \frac{s \not\simeq t}{\square}\ (\mu)$$

where (i) $\theta = mgu(l, l')$; (ii) l' is not a variable; (iii) $l\sigma \succ_{gr} r\sigma$ for some grounding substitution σ;

$\mu = mgu(s, t)$;

Proofs in UPL can be represented in the same way as proofs in UP (see Section 7). And in the same way we can define notions of a P-relevant instantiator and a P-relevant instance.

By a simple lifting argument we can prove the following lemma, connecting UPL with UP calculus.

Lemma 5. *Let Lit be a set of literals such that $Lit\bot$ is satisfiable and $\mathcal{L}$ be a set of ground closures of literals from Lit such that the empty clause is derivable in* UP *from $\mathcal{L}$. Then, there is a proof P of the empty clause in* UPL *from Lit such that for at least one closure $L \cdot \sigma \in \mathcal{L}$, P-relevant instance of L is $L\theta$ where θ is a proper instantiator and $L\sigma = L\theta\tau$ for some ground substitution τ.*

This lemma implies that an Inst-saturation process $\{\langle S^i, I^i_\bot, \mathsf{sel}^i\rangle\}_{i=1}^{\infty}$ is Inst-fair if the following holds. Consider a finite set $K = \{(L_1 \vee C_1), \ldots, (L_n \vee C_n)\}$ of clauses from S^∞, such that for infinitely many i we have $\mathsf{sel}^i(L_j \vee C_j) = L_j$ for $1 \leq j \leq n$. Let P be an UPL proof of the empty clause from $\{L_1, \ldots, L_n\}$ and $L_i\theta$ be a proper P-relevant instance. Then, for some step j, all ground closures $(L_i \vee C_i) \cdot \theta\sigma$ are Inst-redundant in S^j.

We can observe that since θ is a proper instantiator, to make all closures $(L_i \vee C_i) \cdot \theta\sigma$ Inst-redundant, we can just add $(L_i \vee C_i)\theta$ to the clause set.

9 Representation of closures via dismatching constraints

We have seen that in the process of obtaining a saturated set we make certain closures Inst-redundant by proper instantiations. It might be desirable to discard these redundant closures when we consider UPL calculus. In this section we show how it can be done with the help of dismatching constraints, defined below. We remark that in the context of resolution and paramodulation various kinds of constraints have been considered (see e.g. [14, 10, 6]).

A *simple dismatching constraint* is a formula $ds(\bar{s}, \bar{t})$, where $\bar{s}$, $\bar{t}$ are two variable disjoint tuples of terms, with the following semantics. A solution to a constraint $ds(\bar{s}, \bar{t})$ is a substitution σ such that for every substitution γ, $\bar{s}\gamma \neq \bar{t}\sigma$, (here $=$ is the syntactic equality). It is easy to see that a constraint $ds(\bar{s}, \bar{t})$ is satisfiable if and only if for all substitutions γ, $\bar{s}\gamma \neq \bar{t}$. In other words, a dismatching constraint $ds(\bar{s}, \bar{t})$ is not satisfiable if and only if there is a substitution μ such that $\bar{s}\mu = \bar{t}$, which is a familiar matching problem. We will use conjunctions of simple dismatching constraints, called just *dismatching constraints*, $\wedge_{i=1}^{n} ds(\bar{s}_i, \bar{t}_i)$, where $\bar{s}_i$ is variable disjoint from all $\bar{t}_j$, and $\bar{s}_k$, for $i \neq k$. Let us note that there is a polynomial time algorithm for testing satisfiability of the dismatching constraints. To check whether a constraint $\wedge_{i=1}^{n} ds(\bar{s}_i, \bar{t}_i)$ is (un)satisfiable, we just need to solve n matching problems.

A *constrained clause* $C \mid [\ D\ ]$ is a clause C together with a dismatching constraint D. We will assume that in a constrained clause $C \mid [\ \wedge_{i=1}^{n} ds(\bar{s}_i, \bar{t}_i)\ ]$, the clause C is variable disjoint from all s_i, $1 \leq i \leq n$. A *constrained clause* $C \mid [\, D\,]$ represents the set of all ground closures $C \cdot \sigma$, denoted as $Cl(C \mid [\, D\,])$, such that σ is a solution to D. For a set S of constrained clauses, $Cl(S)$ denotes the set of all ground closures represented by constrained clauses from S.

Now if we consider a set of clauses S such that $C \in S$ and $C\theta \in S$ for some proper instantiator θ, then we can discard all Inst-redundant ground closures $C \cdot \theta\sigma$, by adding a dismatching constraint to C, obtaining $C \mid [\ ds(\bar{x}\theta, \bar{x})\ ]$. In the general case, when a constrained clause $C \mid [\ D\ ]$ is in S and we add $C\theta$ to S for some proper instantiator θ, then we can discard all Inst-redundant ground closures $C \cdot \theta\sigma$, by extending the dismatching constraint D, obtaining $C \mid [\ D \wedge ds(\bar{x}\theta, \bar{x})\ ]$. We can always assume that all variables in $\bar{x}\theta$ are disjoint from variables in C and D.

The notion of Inst-redundancy can be adapted from clauses to constrained clauses, by saying that a constrained clause $C \mid [\ D\ ]$ is Inst-redundant if all closures in $Cl(C \mid [\ D\ ])$ are Inst-redundant.

Let S be a set of constrained clauses, then $Unc(S)$ denotes the set of all unconstrained clauses obtained from S by dropping all constraints. We say that a set of constrained clauses S is *well-constrained* if $Cl(S) \setminus \mathcal{R}_{\mathrm{Inst}}(Cl(S)) = Cl(Unc(S)) \setminus \mathcal{R}_{\mathrm{Inst}}(Cl(Unc(S)))$. Thus, constraints in well-constrained sets of clauses is just a tool of discarding Inst-redundant closures.

Next we can replace, UPL calculus with the calculus on constrained literals.

Unit paramodulation with dismatching constraints (UPD)

$$\frac{(l \simeq r) \mid [\ D_1\ ] \quad L[l'] \mid [\ D_2\ ]}{L[r]\theta \mid [\ (D_1 \wedge D_2)\theta\ ]}\ (\theta) \qquad\qquad \frac{s \not\simeq t \mid [\ D\ ]}{\Box}\ (\mu)$$

where (i) $\theta = mgu(l, l')$; (ii) l' is not a variable; (iii) for some grounding substitution σ, satisfying $(D_1 \wedge D_2)\theta$, $l\sigma \succ_{gr} r\sigma$;

where (i) $\mu = mgu(s, t)$; (ii) $D\mu$ is satisfiable.

Naturally we can define the notion of UPD-redundancy, saying that a constrained literal $L \mid [\ D\ ] \in LD$ is UPD-*redundant* in LD if all closures in $Cl(L \mid [\ D\ ])$ are UP-redundant in $Cl(LD)$. And in the same way as for UP we can define UPD-saturation process and LD^{sat}.

Now in the place of S-relevant literal closures $\mathcal{L}_S$ we define the set of S-relevant constrained literals LD_S as the set of all constrained literals $L \mid [\ D\ ]$ such that

1. $(L \vee C) \mid [\ D\ ] \in S$,
2. $(L \vee C) \mid [\ D\ ]$ is not Inst-redundant in S,
3. $L = \mathsf{sel}(L \vee C)$.

We say that the set of constrained clauses is Inst-saturated if LD_S^{sat} does not contain the empty clause.

The following lemma can be proved by a simple lifting argument.

Lemma 6. *Let LD be a set of constrained literals. If $Cl(LD)^{sat}$ (saturation w.r.t.* UP*) contains the empty clause, then LD^{sat} (saturation w.r.t.* UPD*) also contains the empty clause.*

From Lemma 6 a lifted version of Theorem 1 from Section 5 follows.

Theorem 2. *If a well-constrained set of clauses S is* Inst*-saturated and $Unc(S)\bot$ is satisfiable, then $Unc(S)$ is also satisfiable.*

References

1. L. Bachmair and H. Ganzinger. Equational reasoning in saturation-based theorem proving. In W. Bibel and P.H. Schmitt, editors, *Automated Deduction — A Basis for Applications*, volume I, chapter 11, pages 353–397. Kluwer, 1998.
2. L. Bachmair and H. Ganzinger. Resolution theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume 1, pages 19–100. Elsevier, 2001.
3. P. Baumgartner. FDPLL – a first-order Davis-Putnam-Logeman-Loveland Procedure. In *Proc. CADE*, volume 1831 of *LNAI*, pages 200–219, 2000.
4. P. Baumgartner and C. Tinelli. The model evolution calculus. In F. Baader, editor, *Proc. CADE-19*, number 2741 in LNAI, pages 350–364. Springer, 2003.
5. J.-P. Billon. The disconnection method: a confluent integration of unification in the analytic framework. In *Tableaux 1996*, volume 1071 of *LNAI*, pages 110–126, 1996.
6. R. Caferra and N. Zabel. A method for simultaneous search for refutations and models by equational constraint solving. *J. of Symbolic Computation*, 13(6):613–641, 1992.
7. H. Ganzinger, G. Hagen, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. DPLL(T): fast decision procedures. In *16th Int. Conf. on Computer Aided Verification*, LNCS, 2004. to appear.
8. H. Ganzinger and K. Korovin. New directions in instantiation-based theorem proving. In *Proc. 18th IEEE Symposium on Logic in Computer Science*, pages 55–64. IEEE, 2003.
9. J.N. Hooker, G. Rago, V. Chandru, and A. Shrivastava. Partial instantiation methods for inference in first order logic. *J. of Automated Reasoning*, 28:371–396, 2002.
10. C. Kirchner, H. Kirchner, and M. Rusinowitch. Deduction with symbolic constraints. *Revue Francaise d'Intelligence Artificielle*, 4(3):9–52, 1990. Special issue on automated deduction.
11. S.J. Lee and D. Plaisted. Eliminating duplication with the Hyper-linking strategy. *J. of Automated Reasoning*, 9:25–42, 1992.
12. R. Letz and G. Stenz. Proof and model generation with disconnection tableaux. In *Proc. LPAR 2001*, volume 2250 of *LNAI*, pages 142–156, 2001.
13. Reinhold Letz and Gernot Stenz. Integration of equality reasoning into the disconnection calculus. In *Tableaux 2002*, volume 2381 of *LNAI*, pages 176–190, 2002.
14. R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 371–443. Elsevier, 2001.
15. D. Plaisted and Y. Zhu. Ordered semantic hyper-linking. *J. of Automated Reasoning*, 25(3):167–217, 2000.
16. G. Stenz. *The Disconnection Calculus*. Logos, 2002. Dissertation, TU München.