New Directions in Instantiation-Based Theorem Proving

Harald Ganzinger, Konstantin Korovin
MPI Informatik, D-66123 Saarbrücken, Germany
E-mail: {hg|korovin}@mpi-sb.mpg.de

Abstract

We consider instantiation-based theorem proving whereby instances of clauses are generated by certain inferences, and where inconsistency is detected by propositional tests. We give a model construction proof of completeness by which restrictive inference systems as well as admissible simplification techniques can be justified. Another contribution of the paper are novel inference systems that allow one to also employ decision procedures for first-order fragments more complex than propositional logic. The decision procedure provides for an approximative consistency test, and the instance generation inference system is a means of successively refining the approximation.

1. Introduction

By Herbrand's theorem and by compactness of first-order logic, a set of first-order clauses is unsatisfiable if, and only if, a finite set of ground instances of that set is propositionally unsatisfiable. Most methods for automated theorem proving interleave, in one way or another, the two processes: (i) the enumeration of suitable instances of clauses, and (ii) a demonstration of their propositional unsatisfiability. For instance, a resolution inference can be viewed as consisting of two steps. First the unifier of complementary literals is applied to two parent clauses to which, then, the propositional cut rule is applied to generate the resolvent. The resolvent is kept, but the instances of the premises of the inference are discarded.

It has been argued that this fine-grained interleaving of instance generation and propositional unsatisfiability checking might not always be appropriate. Lee & Plaisted (1992) have observed that the duplication and recombination of literals in the cut step of resolution often leads to duplication of work in subsequent inferences. It has also been argued that with the remarkable improvement in the performance of Sat procedures in recent years propositional unsatisfiability checking should be handled by calling upon such procedures, cf. (Lee & Plaisted 1992, Plaisted & Zhu 2000, Hooker, Rago, Chandru & Shrivastava 2002), among others. The theorem proving methods described in these papers generate successively larger ground instances of the clauses and employ a Sat checker to test their satisfiability. The set of ground clauses generated at each step of the process can be viewed as a sound propositional approximation of the given first-order problem. Solving safe approximations (relaxations) of difficult problems is a standard technique for restricting search in many areas, including combinatorial optimization, and program analysis and verification. For instance the program analysis method proposed in (Jackson 2000) and (Jackson 2002) advocates a form of bounded model checking based on generating propositional approximations of first-order program specifications.

Some work in instantiation-based theorem proving is targeted at interleaving propositional methods and instance generation without recombination of clauses. The Davis-Putnam-Loveland-Logeman procedure (Davis & Putnam 1960, Davis, Logemann & Loveland 1962) has been one of the main methods for Sat solving. Baumgartner (2000) exploits that fact and combines instance generation with Davis/Putnam-style satisfiability testing. Billon (1996) and Letz & Stenz (2001) combine instance generation based on resolution with an efficient tableau data structure (without rigid variables) for satisfiability checking. In other approaches the interleaving is based on semantics, and inferences are guided by propositional models. Plaisted & Zhu (2000) compute inferences that reduce the minimal counterexample to the perfect model of the set of ground instances computed so far. Hooker et al. (2002) consider so-called conflicts to a certain model hypothesis, again derived from the current propositional abstraction.

Nevertheless one can say that instantiation-based theorem proving has been much less studied than the classical methods of resolution and semantic tableaux. Far less is known regarding how instance generation can be effectively restricted and guided, how global redundancies can be eliminated, and how effective and fair saturation strategies can be efficiently implemented. This paper gives new answers to some of these questions by studying some fundamental inference systems for instance generation. We prove the completeness of these calculi by showing that they have the property of counterexample reduction for certain candidate models. This implies that redundancy elimination based on implication from smaller clauses is admissible similar to what is the case for most versions of resolution inference systems. In most resolution provers, the deductive core accounts for a rather small part of the system, while most of the complexity of the prover derives from the implementation of powerful, yet efficient redundancy elimination and simplification techniques. The degree of sophistication of the latter methods usually distinguishes experimental prototypes from practically useful tools. With our results instance generation-based provers can now also be equipped with similarly powerful techniques for simplification. The inference systems presented here have been inspired by the work of Hooker et al. (2002). Compared to the results of the latter paper, our inference systems are more flexible (we also allow for hyper-inferences), are compatible with many of the standard simplification techniques, and we identify more general criteria for when instance generation processes are fair (and thus refutationally complete). We also believe that our completeness results, involving redundancy elimination criteria, are correct, whereas the main redundancy elimination criterion in Hooker et al. (2002) destroys refutational completeness of the system (see Section 5 below).

A second contribution of this paper is directed at employing decision procedures for first-order fragments more complex than propositional logic. Instance generation as it has been considered previously ships ground clauses to a Sat solver. We describe methods whereby also non-ground clauses can be forwarded to a decision procedure for a decidable non-ground clausal fragment. The intended benefit of this is that fewer instances have to be generated explicitly as the decision procedure can handle more precise abstractions of the given clauses. Our preliminary results presented in this paper may open up novel ways of fruitfully using the large number of decision procedures for known decidable fragments—see (Fermüller, Leitsch, Tammet & Zamov 1993, Börger, Grädel & Gurevich 1997, Fermüller, Leitsch, Hustadt & Tammet 2001) for overviews—also in situations where problems fall outside the respective fragment.

Technically this paper makes heavy use of model construction methods for counterexample reducing inference systems as developed in (Bachmair & Ganzinger 1994, Bachmair & Ganzinger 2001) and applied, among others, in (Nieuwenhuis & Rubio 1995, Degtyarev & Voronkov 2000, Hähnle, Murray & Rosenthal 2001). We have attempted at keeping the paper self-contained and hope that the reader will not find the proofs too sketchy in some places.

2. Preliminaries

We shall use standard terminology for first-order clause logic. In particular, a *clause* is a possibly empty multiset of literals denoting their disjunction and is usually written as $L_1 \vee \ldots \vee L_n$; a *literal* being either an atomic formula or the negation thereof. We say that C is a subclause of D, and write $C \subseteq D$, if C is a submultiset of D. Variables are usually denoted by x, y, and z, whereas, unless indicated otherwise, letters a, b and c denote constants. If L is a literal, $\overline{L}$ denotes the complement of L.

Substitutions are defined as usual and will be denoted by letters ρ, σ, τ, and θ. A substitution is called a *proper instantiator* of an expression (a literal or clause) if at least one variable of the expression is mapped to a non-variable term. *Renamings* are injective substitutions, sending variables to variables.[1] The result of applying a substitution σ to an expression E is denoted $E\sigma$. Two clauses are *variants* of each other if one can be obtained from the other by applying a renaming. By $\lessdot$ we denote the (strict) *subsumption ordering* on clauses defined as follows: $C \lessdot D$ iff $C\tau \subseteq D$ for some substitution, but $D\rho \subseteq C$ for no substitution ρ. In that case we say that C *strictly subsumes* D. We call C *more specific* than D if $D\tau = C$ for some proper instantiator τ of D. Note that if C is more specific than D, then D strictly subsumes C, but the converse is not true in general, not even in the case of a single literal. In fact $p(x, y) \lessdot p(x, x)$, but $p(x, x)$ is not more specific than $p(x, y)$.

Instance-based theorem proving requires us to work with a refined notion of instances of clauses 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 clause generally has more than one representation by closures. A closure is called *ground* if it represents a ground clause.

Inference systems and completeness proofs will be based on orderings on ground closures. A *closure ordering* is any ordering $\succ$ on closures (modulo alpha renaming) that is total and well-founded. In addition we require that $C \cdot \sigma \succ D \cdot \tau$ whenever $C\sigma = D\tau$ and $C\rho = D$ for some proper instantiator ρ of C. Hence when we compare two ground closures $C \cdot \sigma$ and $D \cdot \tau$ where the former is a *more specific representation* than the latter of the same ground clause then $C \cdot \sigma$ is necessarily greater than $D \cdot \tau$. For example, we have $(p(x) \vee q(y)) \cdot [a/x, b/y] \succ (p(a) \vee y(z)) \cdot [b/z]$ where both ground clauses denoted are the same, but the

[1] Some papers in the area, including (Hooker et al. 2002), allow renamings to be non-injective. A substitution that is not a renaming in this non-standard sense is a proper instantiator according to our definition. We prefer to work with the standard concepts of subsumption and renaming.

clause $p(x) \vee q(y)$ is less specific than $p(a) \vee y(z)$. When the clauses represented are different, we may order them arbitrarily. 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*. The representation is said to be *minimal* if there exists no other representation of $C\sigma$ in S which is smaller in the closure ordering.

The (Herbrand) *interpretations* we deal with are sometimes partial, given by consistent sets I of ground literals. A ground literal L is called *undefined* in I if neither L nor $\overline{L}$ is in I. I is called *total* if for each ground literal I either contains the literal or its complement. A ground clause C is called *true* (or valid) in a partial interpretation I, written $I \models C$, if C is true in each total extension of I, and is called *false* in I, otherwise. Truth values for closures are defined from the truth values of the clauses they represent. Implication (also denoted by $\models$) between (sets of) clauses is defined as usual based on this definition of validity.

3. Completeness of Resolution-Based Instantiation

This section gives a simple model-construction proof of the fact that the combination of generating instances based on resolution together with propositional consistency checking is refutationally complete. Resolution-based instance generation is given by this inference rule (in which the variables in the two premises are assumed to be renamed apart):

Inst-Gen

$$\frac{C \vee L \qquad D \vee \overline{K}}{(C \vee L)\sigma \quad (D \vee \overline{K})\sigma}$$

where σ is the mgu of L and K such that σ is a proper instantiator of L or of K.

The inference has two resolvable premises. Yet it does not generate the resolvent, but rather the two clauses obtained from applying the unifier to the premises. Clearly the inference is sound, and the two conclusions imply propositionally the resolvent $(C \vee D)\sigma$ of the two premises. That suggests if a set of clauses is closed under Inst-Gen, propositional reasoning might suffice to test satisfiability. This result will be stated and proved below. To formalize what is meant by propositional reasoning, let $\perp$ denote both a distinguished constant and the substitution that maps each variable to $\perp$. If S is a set of clauses, by $S\perp$ we denote all ground clauses obtained by applying $\perp$ to each clause in S.

Suppose that S is a set of clauses such that $S\perp$ is satisfied in a model $I_\perp$. Let $\succ$ be a closure ordering. By induction over $\succ$ we associate an interpretation I_S, called *candidate model*, with S that is intended to be a model of "many" ground instances of S. Suppose, as an induction hypothesis, that sets of literals ϵ_D have been defined for the ground closures D smaller than C in $\succ$, and let I_C denote the set $\bigcup_{C \succ D} \epsilon_D$. ($I_C$ is intended to be a model for the closures smaller than C, and ϵ_C is an increment to also make C become true in the final interpretation. Only if S is closed under sufficiently many inferences will these constructions, however, achieve the desired effect.) Suppose that $C = C' \cdot \sigma$. Then define $\epsilon_C = \{L\sigma\}$, if

(i) C is false in I_C;
(ii) C is the minimal representation of $C'\sigma$ in S; and
(iii) L is a literal in C' such that $L\sigma$ is undefined in I_C, and $L\perp \in I_\perp$.

(In that case we say that $L\sigma$ is *produced* by C.) If no such L can be found, we define $\epsilon_C = \emptyset$. If more than one choice for L can be made, we choose one arbitrarily. Finally define I_S to be the set $\bigcup_C \epsilon_C$. We say that a ground closure D is a counterexample (to I_S) if D false in I_S. Obviously, I_S depends on $\succ$ so that we will also write $I_S^{\succ}$ if $\succ$ is not clear from the context. In (iii) we generalize the truth value of $L\perp$ in $I_\perp$ to ground instances $L\sigma$ of L, where, according to the closure ordering, more specific representations of $L\sigma$ are considered first. In that regard our construction is closely related to the model construction of Letz & Stenz (2001).

THEOREM 3.1 Let S be a set of clauses closed under Inst-Gen. Then S is satisfiable if, and only if, $S\perp$ is satisfiable.

Proof. Suppose $S\perp$ is satisfiable. We show that the candidate model I_S is a model of all ground instances of S. Suppose, for the purpose of deriving a contradiction, that there exists a total extension I of I_S that is not a model of S, and let $D = D' \cdot \sigma$ be the minimal ground instance of S that is false in I. As D is not productive (otherwise it would be true) there exists no literal L' in D' such that $L'\sigma$ is undefined in I_D and $L'\perp \in I_\perp$. On the other hand, $L\perp \in I_\perp$ for some L in D' as $D'\perp$ is true in $I_\perp$. Let us choose one such L, thus $D' = L \vee D''$. We conclude that $\overline{L}\sigma$ is in I_D. Therefore, some ground instance $C' \cdot \tau$ of S produces $\overline{L}\sigma$ into I_S; hence $C' = C'' \vee K$, $K\tau = \overline{L}\sigma$, and $K\perp \in I_\perp$. Consider the inference by Inst-Gen from the premises $C'' \vee K$ and $D'' \vee L$, with ρ the mgu of K and L, generating the clauses $(C'' \vee K)\rho$ and $(D'' \vee L)\rho$. Also, let σ' be a substitution for which $(C'' \vee K)\rho\sigma' = C'\tau$ and $(D'' \vee L)\rho\sigma' = D'\sigma$.

We first show that ρ is a proper instantiator of L or K so that the inference is legal. If ρ neither instantiates L nor K properly, $K\perp = \overline{L}\perp$, but both $K\perp$ and $L\perp$ are in $I_\perp$ which is impossible. Suppose that ρ is a proper instantiator of a variable in K. Then, as S is closed under Inst-Gen, $(C'' \vee K)\rho \cdot \sigma'$ is smaller than $C' \cdot \tau$ so that $C' \cdot \tau$ would not be the most specific representation of $C'\tau$, and hence would not be productive, which is a contradiction. On the other hand, if ρ is a proper instantiator of a variable in L, then $(D'' \vee L)\rho \cdot \sigma'$ is a smaller closure than D and false in I, contradicting the minimality of D as a counterexample. □

The significance of this theorem is that instance generation with Inst-Gen, together with propositional satisfiability checking, is a refutationally complete proof method for first-order clauses. Although this result follows from several results proved in the literature it is somewhat surprising that, to our knowledge, up to now it has not been formulated in this basic form.

4. Redundancy Elimination

The proof of Theorem 3.1 shows that Inst-Gen has a property related to the reduction property for counterexamples as formally defined in (Bachmair & Ganzinger 2001). This suggests that the standard notion of redundancy (based on implication from smaller clauses or closures) might be compatible with Inst-Gen in the sense that inferences from redundant clauses can be ignored. Standard redundancy in our case is to be defined as follows. Let S be a set of clauses and C a ground closure. C is called *redundant* in S if there exist closures $C_1, \ldots, C_k$ that are ground instances of S such that, (i) for each i, $C \succ C_i$, and (ii) $C_1, \ldots, C_k \models C$. A clause C (possibly non-ground) is called redundant in S if each ground closure $C \cdot \sigma$ is redundant in S. An inference from premises C_i and with a unifier θ (thus deriving conclusions $C_i\theta$) is *redundant* in S if for any substitution σ grounding all the $C_i\theta$ there exists an index i_0 such that $C_{i_0} \cdot \theta\sigma$ is redundant in S. A set of clauses S is called *saturated up to redundancy* wrt. Inst-Gen if all inferences in Inst-Gen from premises in S are redundant in S.

THEOREM 4.1 Let S be a set of clauses saturated up to redundancy under Inst-Gen. Then S is satisfiable if, and only if, $S\bot$ is satisfiable.

Unfortunately this result does not follow directly from Theorem 3.1 together with the general results in (Bachmair & Ganzinger 2001). The reason is that Inst-Gen inferences, as required in the proof of Theorem 3.1, do not always reduce counterexamples to the candidate model. The two conclusions $C\sigma$ and $D\sigma$ of needed inferences are instances of their respective premises C and D where one represents a counterexample and the other, if added to the set of clauses, would become productive in the model construction. Depending on the unifier only one of the two conclusions needs to be smaller than its corresponding premise. Therefore in some cases a smaller productive clause rather than a smaller counterexample will be computed. The theorem will be a consequence of the completeness of an even more restrictive inference system to be defined below.

What is redundant and what is not depends on the closure ordering that is chosen. Tautologies are redundant for any such choice. In practical applications one would typically choose an ordering in which proper subclosures of a closure (the clauses represented by the closures are in the strict subclause relation) are smaller. More specifically there exist closure orderings for which $C \cdot \sigma \succ D \cdot \rho\sigma$, for every σ, whenever (i) $C \supset D\rho$, or (ii) $C = D\rho$, with ρ neither a renaming nor a proper instantiator of D. (Remember that if $C = D\rho$, with ρ a proper instantiator of C, then necessarily $C \cdot \sigma \prec D \cdot \rho\sigma$ in any closure ordering.) Therefore, if the ordering satisfies (i) and (ii), C is redundant in any set of clauses containing D. For example, $p(a) \vee q(f(a, x))$ can be deleted when $q(f(y, x))$ is present, as can $p(x) \vee q(f(x, x))$ when $p(y) \vee q(f(x, y))$ exists. *Subsumption resolution* is a special case of resolution where the resolvent subsumes a subclause of one of its premises. That is the case if the premises are of the form $C \vee L$ and $D \vee C\sigma \vee \overline{L}\sigma$. Adding the $D \vee C\sigma$ (after deleting one of the copies of $C\sigma$ in the resolvent) makes the second premise become redundant if the ordering satisfies (i). In such a case it appears preferable to simplify $D \vee C\sigma \vee \overline{L}\sigma$ to $D \vee C\sigma$ rather than applying Inst-Gen, and standard redundancy justifies these simplifications.

If closures denote the same ground clause, more general representations become redundant in the presence of more specific ones. For example, if a unary f and a constant a are the only function symbols in the signature, then the unit clause $p(x)$ is redundant in any clause set containing the facts $p(a)$ and $p(f(y))$. Employing elaborate constraint notations such as the ones proposed by Caferra & Zabel (1992), might be useful in this regard.

Redundancy of an inference is based on redundancy of those ground instances of its premises that are compatible with the unifier. In particular, if one of the premises of an inference is redundant, so is the inference. For effective saturation it is important to observe another consequence of the definition:

PROPOSITION 4.2 Let $C\theta$ be a conclusion of an inference and a proper instance of its respective premise C. If $C\theta$ is in S, or is redundant in S, the inference is redundant.

Proof. Let σ be a grounding substitution. If θ properly instantiates C we have that $C\theta \cdot \sigma \prec C \cdot \theta\sigma$, with both closures denoting the same ground clause. If $C\theta$ is in S or is redundant in S, $C\theta \cdot \sigma$ follows from closures smaller than or equal to $C\theta \cdot \sigma$. Therefore, $C \cdot \theta\sigma$ follows from closures strictly smaller than $C \cdot \theta\sigma$ and, hence, is redundant. The redundancy of the inference now follows by definition. □

The significance of the proposition is that by generating (one of) those conclusions of an inference that are proper instances of their respective premises one can make the inference become redundant. Note that at least one of the conclusions of any inference must be of this kind. Conclusions that are no proper instances of their premises may be ignored and need not be added to the clause set to achieve saturation.

In general, redundancy for inferences is even stronger than these special cases. Let us discuss one of the more specific aspects with an example. If we have these three clauses (written as implications)

$$\begin{array}{rcll} & & e(a,b) & (1)\\ e(x,y) & \supset & e(x,f(y)) & (2)\\ e(a,y) & \supset & e(a,f(y)) & (3) \end{array}$$

then an inference between (1) and (2) produces the instance $e(a,b) \supset e(a,f(b))$ of (2), applying the unifier $[a/x,b/y]$ to (2). The inference is redundant as the latter clause is implied by (3), with $(3)\cdot[b/y] \prec (2)\cdot[a/x,b/y]$. Actually we may think of (3) as derived by the inference between (1) and (2), followed by abstracting b to y. This abstraction is sound, and for the inference to be reducing it is sufficient that one variable be properly instantiated. The inferences between (2) and (3) are redundant for similar reasons. What the example demonstrates is, essentially, that we may flexibly choose the degree of instantiation in SInst-Gen. The amount of instantiation to choose will depend on the particular application. In the section 7 below we will discuss certain specific instantiation strategies.

5. Hyper-Inferences with Semantic Selection

Based on unrestricted resolution, instance generation as represented by Inst-Gen is a rather prolific inference, and we would like to restrict it by schemes for selecting resolvable literals. The candidate model construction on which the proof of Theorem 3.1 is based starts out from any propositional model $I_\perp$ of $S\perp$ and generalizes it to an interpretation for all ground instances of literals occurring in S. The idea is that any ground instance $L\sigma$ of L should become true if $L\perp$ is true in $I\perp$. But this generalization might give rise to *conflicts*. Suppose that $p(f(x,a))$ and $\neg p(f(b,y))$ occur both in S. A conflict arises if both $p(f(\perp,a))$ and $\neg p(f(b,\perp))$ are true in $I_\perp$, as assigning true to both $p(f(b,a))$ and $\neg p(f(b,a))$ would result in an inconsistent interpretation. Any conflict leads to a resolution inference. Conversely, only resolution inferences arising from conflicts need to be considered, as we shall prove below. In other words, inferences can be guided by the propositional model for the approximation $S\perp$ of S. Also we shall move from binary inferences to hyper-resolution inferences (with binary inferences as special cases) to also accommodate macro steps whenever desired. Let us define this more formally now.

Let S be a set of clauses such that $S\perp$ is propositionally consistent. Let $I_\perp$ be a propositional model of $S\perp$. For each clause $C \in S$ define $\mathsf{sat}_\perp(C) = \{L \in C \mid I_\perp \models L\perp\}$. We consider *selection functions* sel on clauses (modulo renaming) for which $\emptyset \neq \mathsf{sel}(C) \subseteq \mathsf{sat}_\perp(C)$. Literals L in $\mathsf{sel}(C)$ are called *selected* in C (by sel). Thus, selection functions select some or all of the literals in a clause for which the $\perp$-instance is true in $I_\perp$. Selection functions satisfying these restrictions are said to be *based* on the interpretation $I_\perp$. Instance generation, based on a given selection function sel, is defined as follows.

SInst-Gen

$$\frac{\overline{L}'_1 \vee C_1 \qquad \ldots \qquad \overline{L}'_k \vee C_k \qquad L_1 \vee \ldots \vee L_k \vee D}{(\overline{L}'_1 \vee C_1)\sigma \quad \ldots \quad (\overline{L}'_k \vee C_k)\sigma \quad (L_1 \vee \ldots \vee L_k \vee D)\sigma}$$

where (i) σ is the most general unifier of $L_1 = L'_1, \ldots, L_k = L'_k$, (ii) the literals L_i in the last premise are exactly the ones selected by sel in that clause; and (iii) each $\overline{L}'_i$ is selected in its respective clause by sel, for $1 \leq i \leq k$.

This inference rule resolves conflicts only, and in this regard is related to the system in (Hooker et al. 2002). Our system is more flexible, however, in that we allow for hyper-inferences, and one may choose, by selection, for which clauses hyper-inferences are enabled.

PROPOSITION 5.1 In any inference by SInst-Gen, the mgu σ is a proper instantiator for at least one of the variables in each of the pairs (L_i, L'_i) of unified literals.

Proof. If for some i, σ does not properly instantiate one of the variables in L_i or L'_i then $L_i\perp = L'_i\perp$. As selection is based on some model $I_\perp$ of $S\perp$, we would have that both $L_i\perp$ and $\overline{L}'_i\perp$ are in $I_\perp$ which is a contradiction. □

The proposition shows that SInst-Gen is a more restrictive version of Inst-Gen for selection functions that select exactly one literal in each clause.

THEOREM 5.2 Let S be a set of clauses such that $S\perp$ is satisfiable. If S is saturated up to redundancy by SInst-Gen for a selection function based on a model of $S\perp$ then S is satisfiable.

Proof. The proof relies on a candidate model construction that is the same as the one given in Section 3, except for the slight modification that for a closure $(C \vee L)\cdot\sigma$ to produce $L\sigma$, in addition L must be selected in that closure. Let I_S be the interpretation associated with S by that construction. We will show, by induction on the closure ordering, that if C is a closure in S that is not productive then C is true in I_C. As productive closures C are true in $I_C \cup \epsilon_C$ thus, by monotonicity of the candidate model construction, true in I_S, the theorem follows.

Suppose, for the purpose of deriving a contradiction that the closure $C = C' \cdot \sigma$ is the minimal (in $\succ$) ground instance of S that is not productive and false in I_C. By induction hypothesis all closures D in S smaller than C are true in $I_D \cup \epsilon_D$, hence are true in I_C. Therefore C cannot be redundant in S. If $\mathsf{sel}(C') = \{L_1, \ldots, L_k\} \subseteq \mathsf{sat}_\perp(C)$ then for each L_i we have $\overline{L}_i\sigma \in I_C$. (Otherwise, since C is in particular a minimal representation, C would be productive.) These $\overline{L}_i\sigma$ are produced by closures $D'_i = (\overline{L}'_i \vee D_i) \cdot \tau_i$ smaller than C. False in $I_{D'_i}$, the closures D'_i cannot be redundant. (In fact, if one D'_i were implied by smaller instances of S, as those by induction hypothesis are true in $I_{D'_i}$, D'_i would be true in $I_{D'_i}$.) Let G_i denote the clause $\overline{L}'_i \vee D_i$. It is straightforward to check that we can apply the rule SInst-Gen to $G_1, \ldots, G_k$ and C', generating conclusions $G_1\theta, \ldots, G_k\theta$ and $C'\theta$, where θ is the most general unifier of $L_1 = L'_1, \ldots, L_k = L'_k$. There exist substitutions μ and ν_i such that $C'\theta\mu = C'\sigma$ and $G_i\theta\nu_i = G_i\tau_i$, $1 \leq i \leq k$. In other words, the closures C and D'_i are instances of the premises of the inference which we have shown to not be redundant in S. This contradicts the assumption that S be saturated. $\square$

Instance generation with semantic selection is complete and compatible with redundancy. Let us at this point briefly discuss the redundancy elimination criterion suggested in (Hooker et al. 2002). Consider this inconsistent set S of clauses:[2]

$$
\begin{array}{llllll}
\underline{p(a,z)} & \vee & q(z), & \underline{\neg p(y,z)} & \vee & r(z), \\
p(a,c) & \vee & \underline{q(c)}, & \neg p(a,c) & \vee & \underline{r(c)}, \\
\underline{\neg q(d)}, & & & \underline{\neg r(d)} & &
\end{array}
$$

$S\perp$ is satisfied in the interpretation in which $p(a,\perp)$, $\neg p(\perp,\perp)$, $p(a,c)$, $q(c)$, $r(c)$, $\neg q(d)$, and $\neg r(d)$ are true and the remaining literals in $S\perp$ are false. An admissible selection function would be one selecting in each clause C the underlined literal. Then only one inference is possible in SInst-Gen, namely between the first two clauses, generating the instance $\neg p(a,z) \vee r(z)$ of the second clause. The other conclusion is a variant of the first clause. According to (Hooker et al. 2002) the inference should be ignored as this new instance is a generalization of the more specific fourth clause. With this criterion no inference is possible, and the inconsistency remains undetected. Analysing the situation with our notion of redundancy, for the instance by $[c/z]$ we find that $(p(a,z) \vee q(z)) \cdot [c/z] \succ (p(a,c) \vee q(c)) \cdot []$, demonstrating that the c-instance of the clause is, in fact, redundant. However, we cannot show that the d-instance is redundant, too. Therefore according to our definition, the inference is not redundant.

[2] Chr. Lynch informed us about this problem with Theorem 4.10 in (Hooker et al. 2002) by giving a similar counterexample.

6. Effective Saturation Strategies

In this section we shall investigate how saturation of a set of clauses can be achieved effectively. 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. We must make sure that all inferences from persistent clauses as they are required by selection based on the limit model of a saturation process are considered. In order to show this a notion of depth-restricted saturation will come in handy.

The *depth* of a clause is the maximal depth of its terms. We say that a clause is n-*bounded* if its depth is less or equal to n. Likewise a closure $C \cdot \sigma$ is n-*bounded* if $C\sigma$ is n-bounded. We say that an inference is n-*bounded* if all clauses in the conclusion are n-bounded. A set of clauses S is called n-*consistent* if the set of all n-bounded ground instances of clauses from S is consistent. From Herbrand's theorem it follows that the set of clauses is consistent if and only if it is n-consistent for each natural number n. An n-bounded closure is called n-redundant in S if it is redundant in the set of all n-bounded ground instances of S. A clause is called n-redundant in S if all its n-bounded closures are redundant in S. Thus, for an n-bounded ground closure C to be n-redundant smaller n-*bounded* closures have to be found that imply C. An n-bounded inference with unifier θ is called n-redundant in S if for each grounding substitution σ for which the instance of the inference by σ is n-bounded there exists a premise C of the inference such that $C \cdot \theta\sigma$ is n-redundant in S.

n-redundancy, for all n, implies redundancy, but the converse is not true in general. However when the closure ordering is such that $C \cdot \sigma \succ D \cdot \tau$ whenever the depth of $C\sigma$ is greater than the depth of $D\tau$, then redundancy implies n-redundancy for every n. If, in addition, the closure ordering is compatible with the subclause ordering, the main redundancy elimination and simplification methods (subclause subsumption, subsumption resolution and tautology elimination) are admissible also with regard to n-redundancy. It is obvious that closure orderings that are both compatible with clause depth and with the subclause ordering exist for any signature.

Let S be a set of clauses such that $S\perp$ is satisfiable. Let $I_\perp$ be a model of $S\perp$ and sel a selection function based on $I_\perp$. Then S is called n-*saturated* up to redundancy wrt. SInst-Gen if all n-bounded SInst-Gen inferences from premises in S are n-redundant in S. Theorem 5.2 also holds in the n-bounded case:

THEOREM 6.1 Let S be a set of clauses such that $S\perp$ is satisfiable. If S is n-saturated up to redundancy wrt. SInst-Gen with a selection function based on a model of $S\perp$ then S is n-satisfiable.

Proof. The proof is a modification of the proof of Theorem 5.2 by applying the candidate model construction to the n-bounded ground instances of S. Note that when we lift a ground resolution inference from n-bounded ground instances of clauses to the clauses themselves, these, as well as their instances by the unifier of the lifted inference, are also n-bounded. In other words, the lifted inference is n-bounded itself. □

A *saturation process* is a finite or infinite sequence of triples $\langle S^i, I^i_\perp, \mathsf{sel}^i\rangle$, where S^i is a set of clauses, $I^i_\perp$ a model of $S\perp$ and sel^i a selection function based on that model. Given $\langle S^i, I^i_\perp, \mathsf{sel}^i\rangle$, a *successor state* $\langle S^{i+1}, I^{i+1}_\perp, \mathsf{sel}^{i+1}\rangle$ is obtained by one of these steps: (i) $S^{i+1} = S^i \cup N$, where N is a finite set of clauses such that $S^i \models N$;[3] or (ii) $S^{i+1} = S^i \setminus \{C\}$, where C is 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 the sequence S^i. A saturation process is called *fair* if whenever an inference from persisting clauses persists, in that there are infinitely many indexes j such that the inference satisfies the selection restrictions with respect to sel^j then there exists an index k such that the inference is redundant in S^k. We call inferences of this kind persistent conflicts (from $C_1, \ldots, C_k, C$). The process is called *length-bounded* if there exists a number λ such that all clauses have length smaller than or equal to λ. Note that if we start with a finite set of clauses and never add any clauses longer than the longest initial clause, the process is length-bounded. In particular inferences by instance generation do not increase clause length, and neither do simplifications by subsumption resolution.

THEOREM 6.2 Suppose $\langle S^i, I^i_\perp, \mathsf{sel}^i\rangle$, $i = 0, 1, 2, \ldots$, is a length-bounded, fair saturation process based on a closure ordering compatible with clause depth, and with S^0 a finite set of clauses. If the process is infinite, then S^0 is satisfiable.

Proof. Let S^0 be a finite set of clauses and $\{\langle S^i, I^i_\perp, \mathsf{sel}^i\rangle\}_{i=0}^\infty$ be an infinite saturation process satisfying the assumptions. Suppose that S^0 is unsatisfiable. Then it is not n-consistent for some natural number n. We show that for some k, S^k contains an n-saturated wrt sel^k (up to redundancy) subset NR such that all n-bounded ground instances of S^0 follow from n-bounded instances of NR. Let NR be the set of all n-bounded clauses which are in $\cup_i S^i$ and not redundant in any S^i, $i = 0, 1, 2, \ldots$. Since there exists only a finite number of n-bounded clauses of a fixed length (and since the process is length-bounded) we may infer that NR is finite and for some m and all $i \geq m$, $NR \subseteq S^i$. Since there are only finitely many conflicts possible in the finite set NR, there exists a $k \geq m$ such that for all $i \geq k$ the inferences corresponding to any conflict in NR are either not admissible with respect to sel^i, or are redundant in NR. In other words, NR is n-saturated w.r.t. sel^k. From Theorem 6.1 it follows that NR is n-consistent.

Now let us show that all n-bounded ground instances of $\cup_i S^i$ follow from n-bounded instances of NR. Suppose $C = C' \cdot \sigma$ is the minimal n-bounded closure in $\cup_i S^i$ which does not follow from n-bounded instances of NR. Then C' is redundant in S^j for some j (for otherwise C' would be in NR). Therefore there exist closures $C_1, \ldots, C_k$ that are ground instances of S^j such that, (i) for each i, $C \succ C_i$, (and since $\succ$ is compatible with clause depth C_i is an n-bounded closure) and (ii) $C_1, \ldots, C_k \models C$. Now we have that at least one closure C_i does not follow from n-bounded instances of NR, which contradicts to the minimality of C.

We have, in particular, shown that all n-bounded instances of S^0 follow from n-bounded instances of NR and NR is n-consistent, therefore S^0 is n-consistent which is a contradiction. □

7. Partial Instantiations and the Use of Non-propositional Decision Procedures

We start with a motivating example.

EXAMPLE 7.1 Suppose we are given the following set of Horn clauses (written as implications):

$$\begin{array}{rcll} & & e(a,b) & (1) \\ e(\underline{x},y) & \supset & e(\underline{x},f(y)) & (2) \\ & & p(a) & (3) \\ p(x) & \supset & p(f(x)) & (4) \\ e(\underline{x},y), p(y) & \supset & \perp & (5) \end{array}$$

This satisfiable (in infinite Herbrand models) set of clauses is almost monadic, except for the occurrences of the binary e. Suppose we classify as (indicated by underlining) variable x in clause (2) and in clause (5) as $\perp$-variables. If we (only) replace $\perp$-variables by $\perp$ then the set of non-propositional clauses obtained by this partial instantiation is in the monadic clause class (Bachmair, Ganzinger & Waldmann 1993, Fermüller et al. 2001), and satisfiability is decidable. If we generate sufficiently many (non-ground) instances which instantiate $\perp$-variables then by checking decidability of these monadic instances we should obtain a complete method. Instance generation should be needed only for resolvable clauses in which the unifier "properly" instantiates one of the $\perp$-variables. We give formal definitions later.

In the example, the following process would be fair: Resolving (1) and (2) gives us the instance $e(a,b) \supset e(a,f(b))$ of (2), where both the $\perp$-variable x and the non-$\perp$-variable

[3] It is sound to add any logical consequence, including those generated by SInst-Gen. The fairness requirement below will make sure that sufficiently many of these inferences are considered.

y of (2) are instantiated. Since the instantiation of non-$\perp$-variables is not strictly required, we abstract it back to y which is sound, yielding

$$e(a, y) \quad \supset \quad e(a, f(y)). \quad (2a)$$

As argued at the end of Section 4, adding $(2a)$ makes the inference between (1), (2) and (2a) become redundant. At this point two more resolution inference need to be considered ((1) and (2a) into (5)) that instantiate a $\perp$-variable. They generate these instances of (5):

$$\begin{array}{rcll} e(a,b), p(b) & \supset & \perp & (6) \\ e(a, f(y)), p(f(y)) & \supset & \perp & (7) \end{array}$$

No more inferences exist that instantiate a $\perp$-variable. Sending, in the remaining set of clauses, the $\perp$-variables to $\perp$ (which we choose to be a) leaves us with this set of clauses (where we have renamed the atoms $e(s,t)$ into $e_s(t)$, with new predicates e_s, to make them become explicitly monadic):

$$\begin{array}{rcll} & & e_a(b) & (1) \\ e_a(y) & \supset & e_a(f(y)) & (2), (2a) \\ & & p(a) & (3) \\ p(x) & \supset & p(f(x)) & (4) \\ e_a(y), p(y) & \supset & \perp & (5) \\ e_a(b), p(b) & \supset & \perp & (6) \\ e_a(f(y)), p(f(y)) & \supset & \perp & (7) \end{array}$$

Hence the procedure terminates with "satisfiable". Instead of computing (6) and (7), by the same argument that lead us to compute (2a) rather than the more specific instance of (2), we could have abstracted b at the non-$\perp$ position in (7) into y and generated

$$e(a, y), p(y) \quad \supset \quad \perp \quad (6')$$

which would have made both inferences into (5) redundant.

From the example we are led to the following definitions. For each clause we classify all variables in that clause to be either a $\perp$-variable or a non-$\perp$-variable. We write $x_\perp$ to indicate that x is a $\perp$-variable. As before $\perp$ also denotes a substitution which now maps all $\perp$-variables to the constant $\perp$ and is the identity on non-$\perp$ variables.

We say that a substitution σ is a *proper instantiator* if it maps a $\perp$-variable to a nonvariable term or to a non-$\perp$ variable; or if it maps a non-$\perp$-variable to a term containing a $\perp$-variable. It is easy to see that a substitution is not a proper instantiator if it maps $\perp$-variables to $\perp$-variables and non-$\perp$ variables to terms not containing any $\perp$-variables. We say that a proper instantiator is *a subs-proper* if it is not a renaming, and call it *$\perp$-proper* otherwise. For example $[y/\underline{x},\ f(u)/z]$ is a subs-proper instantiator, as is $[y/z,\ y/\underline{x}]$, whereas $[y/\underline{x}]$ is a $\perp$-proper instantiator.

PROPOSITION 7.2 Let L be a literal, σ a grounding substitution for L and ρ a substitution such that $\sigma = \rho\sigma'$ for some σ'. If ρ is not a proper instantiator of L then $L\perp\sigma = L\rho\perp\sigma'$.

Proof. Under the given assumptions, $x\perp = \perp = x\rho\perp$ for any $\perp$-variable x in L. For non-$\perp$ variables y in L we obtain $y\rho\perp = y\rho$ as $y\rho$ cannot contain a $\perp$-variable, hence $y\rho\perp\sigma' = y\rho\sigma' = y\sigma = y\perp\sigma$. $\square$

Now we consider the generalization of Inst-Gen to the more general notion of proper instantiators. We assume that all clauses are variable disjoint, which can be achieved by renamings preserving $\perp$-ness of the variables.

When we unify two atoms we need to keep track of $\perp$-variables. To be able to do so we assume that all unifiers are idempotent and do not introduce new variables, i.e. they are the identity on the variables not occurring in the unifying atoms.

GInst-Gen

$$\frac{C \vee L \qquad D \vee \overline{K}}{(C \vee L)\sigma \qquad (D \vee \overline{K})\sigma}$$

where σ is the mgu of L and K such that σ is a proper instantiator of L or of K. In addition we require:

(i) if σ is subs-proper then the variables in the conclusion are classified arbitrarily; (ii) if σ is $\perp$-proper then all $\perp$-variables of the premises, and all variables $\perp$-variables are mapped to, are classified as $\perp$-variables in the conclusion.

Let us note that if we classify all variables in all clauses as $\perp$-variables then GInst-Gen coincides with Inst-Gen.

In order to show completeness of GInst-Gen we have to work with closures with classified variables. We again do not distinguish between a closure and any variant obtained from a renaming preserving the classification of variables. Closure orderings are as before except that they have to be compatible with the more general notion of proper instantiation. In other words, closure orderings satisfy $C\cdot\sigma \succ D\cdot\tau$ whenever $C\sigma = D\tau$ and (i) $C\rho = D$, with ρ a subst-proper instantiator of C, or (ii) C and D are variants of each other and there exists a renaming ρ preserving $\perp$-ness of variables such that the set of $\perp$-variables of $C\rho$ is a strict subset of the set of $\perp$-variables of $D\rho$. According to (ii), $C = p(\underline{x}, y) \cdot [a/\underline{x}, b/y] \succ D = p(\underline{x}', \underline{y}') \cdot [a/\underline{x}', b/\underline{y}']$, as after renaming $\rho = [\underline{x}'/\underline{x}]$ we have that the set of $\perp$-variables of C is a strict subset of the the set of $\perp$-variables of D.

Closure orderings are less restricted now compared to the initial definition in Section 2. Consequently more cases of subsumption can be made compatible with a closure ordering. For instance we may find closure orderings where $p(\underline{x}, f(y)) \cdot [s/x, t/y] \succ p(\underline{x}, z) \cdot [s/x, f(t)/z]$, for all ground terms s and t, the reason being that $[\underline{x}/\underline{x},\ f(y)/z]$

is not a proper instantiator. One would typically prefer closure orderings that are compatible with strict subsumption for non-$\perp$ variables.

Redundancy of clauses and inferences is defined as in Section 4, now referring to closure orderings satisfying the properties we just indicated. As before we obtain that adding the properly instantiated conclusions to a set of clauses makes the inference redundant:

PROPOSITION 7.3 Let $C\theta$ be a conclusion of an inference and a proper instance of its respective premise C. If $C\theta$ is in S, or is redundant in S, the inference is redundant.

Proof. Let σ be a grounding substitution. If θ properly instantiates C and is subs-proper, we have that $C\theta \cdot \sigma \prec C \cdot \theta\sigma$, with both closures denoting the same ground clause, and the proof proceeds as for Proposition 4.2.

Otherwise, if θ is $\perp$-proper then either for some non-$\perp$-variable x in C we have that x is mapped to a $\perp$-variable, or else a $\perp$-variable in C is mapped to a non-$\perp$ variable x occurring in C or in the other premise of the inference. According to (ii) in the definition of GInst-Gen, the occurrence of x in $C\theta$ must be classified as $\perp$. That happens with all variables that are mapped to a variable of different classification. Comparing the variables in C with the corresponding variables in $C\theta$, we find that C has more $\perp$-variables than $C\theta$. Again we may infer that $C\theta \cdot \sigma \prec C \cdot \theta\sigma$, and the rest of the proof follows as in the first case. □

We present a modified candidate model construction for sets of clauses S. Suppose that $S\perp$ is satisfiable with a model $I_\perp$. Let $\succ$ be a closure ordering. By induction over $\succ$ we construct a sequence of interpretations as follows. Let C be any ground instance of S and assume, as an induction hypothesis, that sets of literals ϵ_D have been defined for the ground closures D smaller than C in $\succ$, and let I_C denote the set $\bigcup_{C \succ D} \epsilon_D$. Suppose that $C = C' \cdot \sigma$. Then define $\epsilon_C = \{L\sigma\}$, if
(i) C is false in I_C;
(ii) C is the minimal representation of $C'\sigma$ in S; and
(iii) L is a literal in C' such that $L\sigma$ is undefined in I_C, and $L\perp\sigma \in I_\perp$.
(Again we say that $L\sigma$ is *produced* by C.) If no such L can be found, we define $\epsilon_C = \emptyset$. If more than one choice for L can be made, we choose one arbitrarily. The difference from the previous construction is that since $L\perp$ can be a non-ground literal, we check whether the ground instance by σ is true in the model of $S\perp$ and then make $L\sigma$ true in the interpretation we construct. Again define I_S to be the set $\bigcup_C \epsilon_C$. We say that a ground closure D is a counterexample (to I_S) if D false in I_S.

THEOREM 7.4 Let S be a set of clauses saturated up to redundancy with respect to GInst-Gen. Then S is satisfiable if, and only if, $S\perp$ is satisfiable.

Proof. The proof is essentially the same as the proof of Theorem 5.2. The proposition 7.2 is used to infer that inferences between a productive clause and a minimal counterexample have unifiers that are proper instantiators. □

Since the concepts leading to the more restrictive inference are orthogonal to issues of redundancy elimination and effective saturation, Theorems 5.2, 6.1, and 6.2 can be accordingly generalized. On the other hand, since models of $S\perp$ are now infinite in general how to effectively approximate semantic selection is not obvious. This connects our work to results about model building, cf. (Caferra & Zabel 1992, Fermüller & Leitsch 1996), among others.

The method suggested by Theorem 7.4 is to choose the $\perp$-variables in clauses so that with some decision procedure at our disposal, satisfiability of $S\perp$ can be decided. Hence if S is a set of clauses that "almost" falls into one of the known decidable classes, an appropriate choice of the $\perp$-variables might make $S\perp$ conform to the criteria of that fragment. In fact for many decidable clause classes certain constraints about variable occurrences is what makes them decidable. In the monadic class one essentially has a fixed list of variables of which all argument lists of (Skolem) functions must be prefixes of. In the guarded fragment, all variables of a clause must occur in a guard atom, that is as arguments of negatively occurring predicate. Very often, variables have to occur linearly to make a class decidable. In all these cases one may attempt to classify those variables as $\perp$-variables that fail to satisfy the constraints. $\perp$-variables are instantiated with ground terms before the clause is passed to the decision procedures. Many decidable clause classes don not constraint where ground terms may occur in a clause.

8. Future Work

There are many directions for future work. Calculi for equality should be developed, possibly using results from (Letz & Stenz 2002).

Semantic selection is based on the ability to evaluate ground atoms in given models. For infinite models as they might be computed by decision procedures for fragments that do not have the finite model property, model generation and evaluation methods have to be be suitably adapted.

Implementation based on substitution or context-trees (Graf 1995, Ganzinger, Nieuwenhuis & Nivela 2001) should be developed as these data structures should give good performance to instance generation.

The ideas underlying the partial instantiation method of Section 7 should be extended to more general concepts of abstraction for first-order clauses. The non-ground instances resulting from grounding only the $\perp$-variables represent a more precise abstraction of the given clauses than those obtained by any finite set of propositional instances.

This appears as a semantic counterpart to proof-theoretic concepts of abstraction in theorem proving as put forward by Giunchiglia & Walsh (1992).

One might want to combine instance generation with resolution in order to exploit its lemma generation capabilities for speeding up proof search. Our concept of fair processes allows one to add resolvents, but doing so might not render the corresponding instance generation inference redundant.

Finally, it should be interesting to see what kind of decision procedures for which fragments can be obtained by the combination of existing decision procedures with instance generation made possible through Theorem 7.4.

References

Bachmair, L. & Ganzinger, H. (1994), 'Rewrite-based equational theorem proving with selection and simplification', *J. Logic and Computation* **4**(3), 217–247. Revised version of Research Report MPI-I-91-208, 1991.

Bachmair, L. & Ganzinger, H. (2001), Resolution theorem proving, *in* A. Robinson & A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. 1, North Holland, chapter 2, pp. 19–100.

Bachmair, L., Ganzinger, H. & Waldmann, U. (1993), Superposition with simplification as a decision procedure for the monadic class with equality, *in* G. Gottlob, A. Leitsch & D. Mundici, eds, 'Proc. of Third Kurt Gödel Colloquium, KGC'93', Vol. 713 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 83–96. Revised version of Research Report MPI-I-93-204.

Baumgartner, P. (2000), FDPLL — A first-order Davis-Putnam-Logeman-Loveland procedure, *in* D. McAllester, ed., 'Automated Deduction – CADE-17, 17th International Conference on Automated Deduction', LNAI 1831, Springer-Verlag, Pittsburgh, PA, USA, pp. 200–219.

Billon, J.-P. (1996), The disconnection method: a confluent integration of unification in the analytic framework, *in* 'Tableaux 1996', Vol. 1071 of *LNAI*, pp. 110–126.

Börger, E., Grädel, E. & Gurevich, Y. (1997), *The Classical Decision Problem*, Perspectives of Mathematical Logic, Springer-Verlag. Second printing (Universitext) 2001.

Caferra, R. & Zabel, N. (1992), 'A method for simultaneous search for refutations and models by equational constraint solving', *Journal of Symbolic Computation* **13**(6), 613–641.

Davis, M., Logemann, G. & Loveland, D. (1962), 'A machine program for theorem proving', *Communications ACM* **5**(7), 394–397.

Davis, M. & Putnam, H. (1960), 'A computing procedure for quantification theory', *J. Association for Computing Machinery* **7**, 201–215.

Degtyarev, A. & Voronkov, A. (2000), Stratified resolution, *in* D. McAllester, ed., 'Automated Deduction – CADE-17, 17th International Conference on Automated Deduction', LNAI 1831, Springer-Verlag, Pittsburgh, PA, USA, pp. 365–384.

Fermüller, C. G. & Leitsch, A. (1996), 'Hyperresolution and automated model building', *Journal of Logic and Computation* **6**(2), 173–203.

Fermüller, C., Leitsch, A., Hustadt, U. & Tammet, T. (2001), Resolution decision procedures, *in* A. Robinson & A. Voronkov, eds, 'Handbook of Automated Reasoning', Elsevier, chapter 25, pp. 1791–1850.

Fermüller, C., Leitsch, A., Tammet, T. & Zamov, N. (1993), *Resolution Methods for the Decision Problem*, Vol. 679 of *Lecture Notes in Computer Science*, Springer Verlag.

Ganzinger, H., Nieuwenhuis, R. & Nivela, P. (2001), Context trees, *in* 'Proc. International Joint Conference on Automated Reasoning', Vol. 2083 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 242–256.

Giunchiglia, F. & Walsh, T. (1992), 'A theory of abstraction', *Artificial Intelligence* **57**(2-3), 323–389.

Graf, P. (1995), Substitution tree indexing, *in* J. Hsiang, ed., 'Rewriting Techniques and Applications, 6th International Conference, RTA-95', LNCS 914, Springer-Verlag, Kaiserslautern, Germany, pp. 117–131.

Hähnle, R., Murray, N. V. & Rosenthal, E. (2001), Ordered resolution vs. connection graph resolution, *in* R. Goré, A. Leitsch & T. Nipkow, eds, 'Automated Reasoning, First International Joint Conference, IJCAR 2001', LNAI 2083, Springer-Verlag, Siena, Italy, pp. 182–194.

Hooker, J., Rago, G., Chandru, V. & Shrivastava, A. (2002), 'Partial instantiation methods for inference in first order logic', *J. Automated Reasoning* **28**, 371–396.

Jackson, D. (2000), Automating first-order relational logic, *in* 'Proceedings of the 8th ACM SIGSOFT International Symposium on Foundations of Software Engineering', ACM Press, pp. 130–139.

Jackson, D. (2002), 'Alloy: a lightweight object modelling notation', *ACM Transactions on Software Engineering and Methodology (TOSEM)* **11**(2), 256–290.

Lee, S.-J. & Plaisted, D. (1992), 'Eliminating duplication with the Hyper-linking strategy', *J. Automated Reasoning* **9**, 25–42.

Letz, R. & Stenz, G. (2001), Proof and model generation with disconnection tableaux, *in* 'Proc. LPAR 2001', Vol. 2250 of *LNAI*, pp. 142–156.

Letz, R. & Stenz, G. (2002), Integration of equality reasoning into the disconnection calculus, *in* 'Tableaux 2002', Vol. 2381 of *LNAI*, pp. 176–190.

Nieuwenhuis, R. & Rubio, A. (1995), 'Theorem proving with ordering and equality constrained clauses', *J. Symbolic Computation* **19**(4), 321–352.

Plaisted, D. & Zhu, Y. (2000), 'Ordered semantic hyper-linking', *J. Automated Reasoning* **25**(3), 167–217.