Rewrite-based Deduction and Symbolic Constraints

Robert Nieuwenhuis*

Technical University of Catalonia, Dept. LSI,
Jordi Girona 1, 08034 Barcelona, Spain
roberto@lsi.upc.es

1 Introduction

Building a state-of-the-art theorem prover requires the combination of at least three main ingredients: good theory, clever heuristics, and the necessary engineering skills to implement it all in an efficient way. Progress in each of these ingredients interacts in different ways.

On the one hand, new theoretical insights replace heuristics by more precise and effective techniques. For example, the completeness proof of *basic paramodulation* [NR95,BGLS95] shows why no inferences below Skolem functions are needed, as conjectured by McCune in [McC90]. Regarding implementation techniques, ad-hoc algorithms for procedures like demodulation or subsumption are replaced by efficient, re-usable, general-purpose indexing data structures for which the time and space requirements are well-known.

But, on the other hand, theory also advances in other directions, producing new ideas for which the development of implementation techniques and heuristics that make them applicable sometimes takes several years. For example, basic paramodulation was presented in 1992, but it was not applied in a state-of-the-art prover until four years later, when it was considered a "key strategy" by McCune for finding his well-known proof of the Robbins conjecture [McC97] by basic paramodulation modulo associativity and commutativity (AC).

Provers like Spass [Wei97], based on (a still relatively small number of) such new theoretical insights, are now emerging and seem to be outperforming the "engineering-based" implementations of more standard calculi, in spite of still lacking more refined implementation techniques (as we will see later on).

One obstacle for progress in the development of such provers seems to be the scarcity of large enough research teams with knowledge in the three aspects: theory, heuristics and implementation techniques. (Another problem may be that implementation efforts seem to produce less publications on a researcher's CV than the same efforts on theory.)

McCune's successful application of AC-paramodulation also illustrates the effectiveness—and the need—of building-in more and more knowledge about the problem domain (here, equality and the AC properties of some symbols) inside the general-purpose logics (first-order clausal logic). In our opinion, deduction

* Partially supported by the ESPRIT Basic Research WG CCL-II.

with constraints is an adequate paradigm for doing this in a clean way. It uses specialized (constraint solving) techniques in the different constraint logics, supporting the reasoning process in the general-purpose logic. The interface between the two is through the variables: the constraints delimit the range of the quantifiers, and hence define the relevant instances of the expressions. For instance, in paramodulation- or resolution-based systems, a constrained clause $C \mid T$ represents the set of ground instances of the clause part C that satisfy the constraint T, and unification is replaced by equality constraint solving.

Here we will address the current theoretical and practical challenges concerning the construction of saturation-based provers by ordered paramodulation techniques and symbolic constraints. We will briefly survey the considerable amount of recent progress that has been made on the theoretical side, and mention some of the most promising ideas that are ready for use in actual provers. At the same time, we will describe a number of open problems (of both theoretical and practical nature) that still need to be solved in order to make other theoretical advances applicable. In some cases we will point to possible solutions for these problems.

A special attention will be devoted to *redundancy proving* and its very close relationship with *implicit* inductive theorem proving, that is, where the user does not need to provide explicit induction schemes.

2 Ordered Paramodulation, Redundancy and Saturation

We now very shortly survey some fundamental theoretical results in this area. For more details and further references on most of the contents of this section, see [BG98,NR99b].

The inference rule for paramodulation[1] [RW69]

$$\frac{C \vee s \simeq t \qquad D}{(C \vee D[t]_p)\sigma} \qquad \text{where } \sigma = mgu(s, D|_p)$$

has been further restricted in many ways. For example, its refutation completeness is preserved when applied only if $D|_p$ is not a variable, and if, for some ordering $\succ$ on terms and equations, the paramodulation steps involve only maximal terms of maximal equations of both premises. In this case it is called *superposition* [BG94]. *Ordered paramodulation* is the slightly less restricted version of superposition where inferences also take place *on* non-maximal sides of equations [HR91].

[1] Here $D|_p$ denotes the subterm of the clause D at position p, and $D[t]_p$ denotes the result of replacing in D that subterm by t. If $D|_p$ is in the term u of a (positive or negative) equation $u \simeq v$ in D, then we say that the paramodulation step *involves* the terms s and u; more precisely, it takes place *with s on u*.

2.1 Paramodulation with constrained clauses

By expressing the ordering and unification restrictions as inherited constraints, paramodulation remains complete [NR95]. Then it becomes:

$$\frac{C \vee s \simeq t \mid T \qquad D \mid T'}{C \vee D[t]_p \mid T \wedge T' \wedge s = D|_p \wedge OC}$$

where the part OC of the constraint represents the ordering restrictions, that is, OC is of the form $s > t \wedge \ldots$ Here the symbols $=$ and $>$ in the constraints are interpreted, respectively, as syntactic equality $\equiv$ of terms and as the given ordering $\succ$ on terms. A main advantage is that the ordering and equality restrictions of the inferences can be kept in constraints and inherited between clauses: if some inference is not compatible with the required restrictions (applied on the current inference rule *and on the previous ones*), then it produces a conclusion with an unsatisfiable constraint, i.e., a tautology, and hence the inference is not needed. The basicness restriction (no inferences computed on terms coming from unifiers of ancestor inferences) [BGLS95,NR95] in this notation is a natural consequence of the fact that inferences only take place on the clause part C of $C \mid T$, and no unifiers are ever applied to C.

2.2 Paramodulation modulo E

It is well-known that paramodulation with some axioms, like the AC axioms, generates many slightly different permuted versions of clauses. Hence for efficiency reasons it is many times better to treat all these clauses together as a single one representing the whole class. This leads to the notion of building-in equational theories E, where the axioms E are removed, and instead one uses E-paramodulation:

$$\frac{C \vee s \simeq t \qquad D}{(C \vee D[t]_p)\sigma} \qquad \text{for all } \sigma \in mgu_E(s, D|_p)$$

where $mgu_E(s,t)$ denotes a complete set of E-unifiers of s and t. For example, for the case where E consists of the AC-axioms for some function symbols, a procedure was first given in [PS81]. If f is such an AC-symbol, then by (purely equational) E-paramodulation with $f(a,x) \simeq x$ on $f(b,y) \simeq g(y)$ we can infer $b \simeq g(a)$, and also $f(b,z) \simeq g(f(a,z))$, where z is a new variable.

2.3 E-paramodulation with constraints

When dealing with constrained clauses, E-paramodulation is expressed exactly as ordinary paramodulation. *The only aspect that changes is the interpretation of the constraints:* now $=$ is interpreted as E-equality, instead of syntactic equality[2]. Apart from the basicness restriction, an additional advantage is now that only one conclusion is generated, instead of one conclusion for each E-unifier[Vig94,NR97]. This can have dramatic consequences. For example, there are more than a million unifiers in $mgu_{AC}(f(x,x,x), f(y_1,y_2,y_3,y_4))$.

[2] Although for some E *extended* inference rules are needed (see, e.g., [RV95]).

2.4 Redundancy and saturation

Roughly, a clause C is *redundant* in a set of clauses S if C is a logical consequence of smaller (with respect to the given clause ordering) clauses of S. This abstract notion covers well-known practical simplification and elimination techniques like demodulation or subsumption, as well as many other more powerful methods. A similar notion of redundancy of inferences exists as well, and a set of clauses is called *saturated* for a given inference system $\mathcal{I}$ if it is closed under $\mathcal{I}$, up to redundant inferences. Roughly, *saturation* is a procedure that adds conclusions of non-redundant inferences and removes redundant clauses. In the limit such a procedure produces a saturated set.

Saturation with respect to appropriate $\mathcal{I}$ is refutation complete: a model can be built for every saturated set not containing the empty clause [BG94]. This is not only of theoretical value: Spass successfully applies finite saturation to prove satisfiability for non-trivial problems (winning in the corresponding category of the last two CADE ATP System Competitions).

The availability of decision procedures for subproblems is well-known to be very useful in a prover. The ideas explained so far in this section have also generated a number of new results in this direction. For example, superposition with simplification can be used as a decision procedure for the monadic class with equality [BGW93b] (which is equivalent to a class of set constraints [BGW93a]). Similar very recent results have been obtained for the guarded fragment [GMV99,GdN99]. Results on the complexity and decidability of unification problems in Horn theories have been obtained by basic paramodulation [Nie98] and sorted superposition [JMW98].

3 Perspectives

We now mention some of the most promising ideas in this field that are ready for use in actual provers, and describe a number of open problems (of both theoretical and practical nature) that still need to be solved in order to make other theoretical advances applicable.

3.1 Basicness and redundancy

From several practical experiments it is known that the basic restriction in paramodulation (modulo the empty theory) saves a large amount of (in some sense, repeated) work and behaves quite well in practice. It is also easy to implement in most provers by marking *blocked* subterms, i.e., the point where the constraint starts.

Open problem 1: However, it is well-known that full simplification by demodulation is incomplete in combination with the basic strategy (see [NR95] for counter examples). Although some ideas are given in [LS98], better results are needed for practice. We conjecture (this conjecture is supported by our practical experiments) that unrestricted *forward* demodulation (and, in general, forward redundancy), which is where 90% of the time is spent in most provers, does not lead to incompleteness. For backward demodulation, *weakening* of the constraint (as explained in [NR95]) would still be needed.

Let us now look at simplification in the context of E-paramodulation with constraints. Consider again McCune's AC-paramodulation proof of the Robbins conjecture, where no constraints were used. Instead, for each paramodulation inference, complete sets of unifiers were still computed, and one new equation added for each one them (although heuristics were used to discard some of the unifiers). One possible reason for not using constraints might have been the next open problem:

Open problem 2: How can we apply a constrained equation $s \simeq t \mid T$ in a demodulation step without solving the E-unification problem in T? A quite naive solution could be the following. If the equation is small, and hence likely to be useful for demodulation, and the number of unifiers σ of T is small as well, it may pay off to keep some of the instantiated versions $s\sigma \simeq t\sigma$, along with the constrained equation, for use in demodulation. For large clauses this will probably not be useful.

3.2 Orderings

In all provers based on ordered strategies, the choice of the right ordering for a given problem turns out to be crucial. In many cases weaker (size- and weights-based) orderings like the Knuth-Bendix ordering behave well. In others, path orderings like LPO or RPO are better, although they depend heavily on the choice of the underlying *precedence* ordering on symbols.

Open problem 3: How to choose orderings and precedences in practice? The prover can of course recognise familiar algebraic structures like groups or rings, and try orderings that normally behave well for each case, but is there no more general solution?

For the case of E-paramodulation, these aspects are even less well-studied. Furthermore, until very recently, all completeness results for ordered paramodulation required the term ordering $\succ$ to be well-founded, monotonic and total(izable) on ground terms. However, in E-paramodulation, the existence of such a total *E-compatible* ordering is a very strong requirement. For example, a large amount of work has been done on the development of progressively more suitable AC-compatible orderings. But for many E such orderings cannot exist at all. This happens for instance when E contains an idempotency axiom.

Open problem 4: Very recently, the monotonicity requirement on the ordering has been dropped for ordered paramodulation [BGNR99]. However, many open questions remain concerning the completeness of full superposition and the compatibility with redundancy notions when working with non-monotonic orderings (see [BGNR99] for details).

Open problem 5: Also, more research is needed for developing suitable E-compatible orderings (monotonic as well as non-monotonic ones), and for studying their practical behaviour on different problem domains. For example, a nice and simple non-monotonic AC-compatible ordering can be obtained by using RPO on *flattened* terms. What can be done for other theories E?

3.3 Constraint solving

When working modulo the empty theory, equality constraint solving is just syntactic unification.

Regarding ordering constraint solving, many algorithms of a theoretical nature have been given for path orderings, starting with [Com90]. Although deciding the satisfiability of path ordering constraints is NP-complete for all relevant cases, very recently, a family of path ordering constraint solving algorithms of a more practical nature has been given [NR99a].

Open problem 6: Is there any useful ordering for which deciding the satisfiability of (e.g., only conjunctive) constraints is in P? (Although for size-based orderings the problem looks simple, it seems that one quickly runs into linear diophantine (in)equations...) Or, even if not in P, for which orderings can we have good practical algorithms?

Open problem 7: In practice one could use more efficient (sound, but incomplete) tests detecting most cases of unsatisfiable constraints: when a constraint T is unsatisfiable, the clause $C \mid T$ is redundant (in fact, it is a tautology) and can be removed. Are there any such tests?

Open Problem 8: In the context of a built-in theory E, equality constraint solving amounts to deciding E-unifiability problems. Although for many theories E a lot of work has been done on computing complete sets of unifiers, the decision problem has received less attention (see [BS99]). And, as in the previous open problem, are there any sound tests detecting most cases of unsatisfiability? Note that, when the empty clause with a constraint T is found, this denotes an inconsistency only if (the equality part of) T is satisfiable. Hence, unlike what happens with ordering constraints, at least then it is necessary to really prove the satisfiability (and not the unsatisfiability) of constraints. But this still does not require a decision procedure; a semi-decision procedure suffices, and always exists (e.g., by narrowing). This may be useful when the decision problem is extremely hard (associativity) or undecidable (associativity $\cup$ distributivity and extensions).

Open Problem 9: Once the adequate orderings for E-paramodulation have been found (see open problem 5), is there any reasonably efficient ordering constraint solving procedure for them?

3.4 Indexing data structures

As said, for many standard operations like many-to-one matching or unification indexing data structures have been developed that can be used in operations like inference computation, demodulation or subsumption. Such data structures are crucial in order to obtain a prover whose throughput remains stable while the number of clauses increases.

Open Problem 9: For many operations no indexing data structures have been developed yet. For example, assume we use demodulation with unorientable equations, and such an equation $s \simeq t$ is found to be applicable to a term $s\sigma$. Then after matching we have to check whether $s\sigma \succ t\sigma$, i.e., whether the corresponding rewrite step is indeed reductive. If it is not reductive, then the indexing data structure is asked to provide a new applicable equation, and so on. Of course it would be much better to have an indexing data structure that checks matching and ordering restrictions at the same time.

Open Problem 10: Apart from the AC case, indexing data structures for built-in E have received little attention. Especially for matching, at least for purely equational logic, they are really necessary. What are the perspectives for developing such data structures for other theories E?

3.5 More powerful redundancy notions

In the *Saturate* system [GNN95], a number of experiments with powerful redundancy notions has been carried out. For example, *constrained rewriting* turns out to be powerful enough for deciding the confluence of ordered rewrite systems [CNNR98]. Other techniques based on forms of *contextual* and *clausal* rewriting can be used to produce rather complex saturatedness proofs for sets of clauses. In Saturate, the use of these methods is limited, since they are expensive (they involve search and ordering constraint solving) and Saturate is just an experimental Prolog implementation. However, from our experiments it is clear that such techniques importantly reduce the number of retained clauses.

Open Problem 11: Can such refined redundancy proof methods be implemented in a sufficiently efficient way to make them useful in real-world provers? It seems that their cost can be made independent of the size of the clause database of the prover (up to the size of the indexing data structures, but this is the case as well for simple redundancy methods like demodulation). Hence, they essentially slow down the prover in a (perhaps large) linear factor, but may produce an exponential reduction of the search space, thus being effective in hard problems.

3.6 Redundancy and inductive theorem proving

Let us now consider the close relationship between redundancy and inductive proofs[3]. More precisely, we aim at (semi-)automatically proving or disproving inductive validity: given a set of Horn clauses H (the axioms), a clause C (the conjecture) is inductively valid if it is valid in the minimal (or initial) Herbrand model I of H.

Example 1. Let H consist of the 3-axiom group presentation
$\{\ \ e+x=x,\quad i(x)+x=e,\quad (x+y)+z=x+(y+z)\ \ \}$
over a signature $\mathcal{F}$ is $\{e^0, a^0, i^1, +^2\}$ (arities are written as superscripts). Then I (a group with one generator a) is isomorphic to the integers with $+$ and if the

[3] This relationship was already noted in [GS92] and used in a different setting.

conjecture C is $\forall x, y.\ x + y = y + x$ then $I \models C$. Note that this is not the case if there is another generator b, since then $a + b = b + a$ does not hold.

In order to minimise user interaction and hence the amount of needed user expertise, we do not want to require the user to provide *explicit* induction schemes. Instead, we use what has been called *implicit* induction[4], based on well-founded general-purpose term orderings $\succ$.

Our ideas are based on a well-known naive method: enumerate the set of all ground instances of the conjecture C, by progressively instantiating it.

Example 2. Let us continue with the previous example and let C be the (false) conjecture $(x + x) + y = y$. Then we can choose an arbitrary variable of C, say x, as the *induction variable,* and instantiate it in all possible ways, that is, with all members of the set $\{f(x_1 \ldots x_n) \mid f^n \in \mathcal{F}\}$ where the x_i are distinct *fresh* variables. By such an *expansion* step of the induction variable, we obtain four new conjectures, namely $(e+e)+y = y$, $(a+a)+y = y$, $(i(z)+i(z))+y = y$, and $((z+w)+(z+w))+y = y$ (note that, independently of the chosen variable, every ground instance of C is also an instance of one of the four new conjectures). If we continue with $(a + a) + y = y$, in one more step the ground counter example $(a + a) + e = e$ (among others) is obtained, thus disproving $I \models C$.

Indeed, under the assumption that $I \models D$ is decidable for ground clauses D, the method is refutation complete: if $vars(C) = \{x_1 \ldots x_n\}$ then any ground counter example $C\sigma$ can be obtained by $|x_1\sigma| + \ldots + |x_n\sigma|$ expansion steps. Hence it will eventually be found if expansion is applied with fairness. Note that if the validity of ground conjectures is undecidable, refutation completeness is impossible anyway. But of course validity (dis)proofs can then still be obtained in many cases.

Example 3. Let $\mathcal{F}$ be $\{0^0, s^1, +^2\}$ and assume H consists of the equations $0+x = x$ and $s(x)+y = s(x+y)$. Then I is isomorphic to the algebra of natural numbers with $+$. In the case of a constructor discipline, expansion can be limited to constructor instances (here, with 0 and $s(y)$). Therefore, the (false) conjecture $x+x = x$ can be expanded in two ways, into $0+0 = 0$ and $s(y)+s(y) = s(y)$. The instance $0 + 0 = 0$ is valid in I, and hence we continue with $s(y) + s(y) = s(y)$, getting $s(0)+s(0) = s(0)$ and $s(s(z))+s(s(z)) = s(s(z))$. Since $s(0)+s(0) = s(0)$ is a counterexample, the conjecture is disproved. This process can be seen as a tree:

```
          x+x=x
         /     \
     0+0=0    s(y)+s(y)=s(y)
                /            \
      s(0)+s(0)=s(0)    s(s(z))+s(s(z))=s(s(z))
```

[4] But our method does not fall under the *proof-by-consistency* or *inductionless induction* techniques either.

Now let us consider validity proving. Since the leaves of the tree cover all possible instances of the conjecture at the root, it is clear that the root is proved valid if at a certain point all leaves can be shown valid.

Our main point here is that *any standard redundancy proving method can be used for proving the validity of such leaves C*, since it simply amounts to showing that $H \cup L \cup S_{<C\sigma} \models C\sigma$ for all ground instances $C\sigma$ of C, where L is a set of *lemmas*, arbitrary first-order clauses that are already known to be valid in I, and $S_{<C\sigma}$ is the set of nodes of the tree whose complexity measure (wrt. $\succ$) is smaller than the one of $C\sigma$, and can hence be assumed to be valid in I by induction hypothesis.

Note that lemmas can come from user input (proved valid by any induction method) or can be leaves that have already been proved valid, and that both lemmas and axioms can be applied without any ordering restrictions. We also remark that valid ground leaves are always provable only from H.

Redundancy is not only needed in the final stage for proving validity of all leaves. Since nodes that are known to be valid require no further expansion, it is also crucial for efficiency to apply it *during* validity (dis)proofs.

Example 4. Continuing with the previous example, $\forall x, y.\ \ x + y = y + x$ is an inductive theorem. The tree is as follows:

```
                  x+y=y+x
                /         \
         0+y=y+0              s(z)+y=y+s(z)
        /       |              |           \
0+0=0+0    0+s(z)=s(z)+0    s(z)+0=0+s(z)  s(z)+s(w)=s(w)+s(z)
  (1)          (2)              (3)                (4)
```

All leaves are redundant, and hence the theorem is proved. For instance, for leaves (2) and (3) the following redundancy proof applies: $s(z) + 0$ is equivalent to $s(z + 0)$ by the second axiom, which by applying a smaller instance of the root node becomes $s(0 + z)$ and then, by the first axiom, becomes $s(z)$, which by the first axiom is equivalent to $0 + s(z)$.

Note that, unlike what happens in other implicit induction methods, these proofs are quite readible: they simply consist of case analysis and redundancy proofs. One can of course deal with a forest instead of a single tree in case several conjectures are given as input. This will be useful for *simultaneous induction.* Furthermore, unlike other implicit induction methods, here any well-founded ordering can be used, and there are no requirements on H or C (like saturatedness). We also remark that the method is not only applicable in the case of initial semantics, but for any semantics based on Herbrand models (like the class of *all* such models, or, when H is non-Horn, for *perfect model* semantics [GS92]).

Open Problem 12: There are many degrees of freedom in the schematic view of the induction method given here (nodes can be simplified, different notions of complexity of a node can be used, other kinds of expansion, etc.). How can we find the right settings in practice? In principle, all known redundancy methods that fit into the abstract notion of redundancy can be used. What kind of concrete methods will be useful?

Open Problem 13: Of course many well-known difficulties in inductive theorem proving appear as well here. For example, the conjecture may have to be generalized (e.g., in the previous example, $(x+x)+x = x+(x+x)$ cannot be proved directly, but associativity of $+$ is an easily provable generalization of it). Lemmas (other than generalisations of the conjecture) may be needed as well. How to find the right lemmas and generalisations? In particular, how can they be found from failed redundancy proofs?

Open Problem 14: It seems that superposition on $H \cup \{C\}$ sometimes finds good lemmas. This explains some of the successes of proof-by consistency methods. But we believe that the main drawback of the latter method is that it actually tries to prove *all* these superposition consequences. Can a hybrid method be useful, applying and proving by expansion only the useful lemmas obtained by superposition?

4 Concluding remark

We hope that the reader has become motivated for building provers that take profit of the recent theoretical results that are now available, and challenged by some of the open problems that have been described.

References

[BG94] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.

[BG98] Leo Bachmair and Harald Ganzinger. Equational reasoning in saturation-based theorem proving. In W. Bibel and P. Schmitt, editors, *Automated Deduction: A Basis for Applications.* Kluwer, 1998.

[BGLS95] L. Bachmair, H. Ganzinger, Chr. Lynch, and W. Snyder. Basic paramodulation. *Information and Computation*, 121(2):172–192, 1995.

[BGNR99] Miquel Bofill, Guillem Godoy, Robert Nieuwenhuis, and Albert Rubio. Paramodulation with non-monotonic orderings. In *14th IEEE Symposium on Logic in Computer Science (LICS)*, Trento, Italy, July 2–5, 1999.

[BGW93a] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Set constraints are the monadic class. In *Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 75–83, Montreal, canada, June 19–23, 1993. IEEE Computer Society Press.

[BGW93b] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Superposition with simplification as a decision procedure for the monadic class with equality. In *3rd Kurt Gödel Colloquium: Computational Logic and Proof Theory*, LNCS 713, pages 83–96. SpringerVerlag, 1993.

[BS99] Franz Baader and Wayne Snyder. Unification theory. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning.* Elsevier Science Publishers (to appear), 1999.

[CNNR98] Hubert Comon, Paliath Narendran, Robert Nieuwenhuis, and Michael Rusinowitch. Decision problems in ordered rewriting. In *13th IEEE Symposium on Logic in Computer Science (LICS)*, pages 410–422, Indianapolis, USA, June 27–30, 1998.

[Com90] Hubert Comon. Solving symbolic ordering constraints. *International Journal of Foundations of Computer Science*, 1(4):387–411, 1990.

[GdN99] Harald Ganzinger and Hans de Nivelle. A superposition decision procedure for the guarded fragment with equality. In *14th IEEE Symposium on Logic in Computer Science (LICS)*, Trento, Italy, July 2–5, 1999.

[GMV99] Harald Ganzinger, Christoph Meyer, and Margus Veanes. The two-variable guarded fragment with transitive relations. In *14th IEEE Symposium on Logic in Computer Science (LICS)*, Trento, Italy, July 2–5, 1999.

[GNN95] Harald Ganzinger, Robert Nieuwenhuis, and Pilar Nivela. The Saturate System, 1995. Software and documentation available at: `http://www.mpi-sb.mpg.de/SATURATE/Saturate.html`.

[GS92] Harald Ganzinger and Jürgen Stuber. Inductive theorem proving by consistency for first-order clauses (extended abstract). In M[ichaël] Rusinowitch and J[ean-]L[uc] Rémy, editors, *The Third International Workshop on Conditional Term Rewriting Systems, Extended Abstracts*, pages 130–135, Pont-à-Mousson, France, July 8–10, 1992. Centre de Recherche en Informatique de Nancy and INRIA Lorraine.

[HR91] J. Hsiang and M Rusinowitch. Proving refutational completeness of theorem proving strategies: the transfinite semantic tree method. *Journal of the ACM*, 38(3):559–587, jul 1991.

[JMW98] Florent Jacquemard, Christoph Meyer, and Christoph Weidenbach. Unification in extensions of shallow equational theories. In *Proceedings of the 9th International Conference on Rewriting Techniques and Applications, RTA-9*, volume to appear, Tsukuba, Japan, 1998. Springer.

[LS98] C. Lynch and C. Scharff. Basic completion with E-cycle simplification. In *Artificial Intelligence and Symbolic Computation*, lncs 1476, pages 121–121, 1998.

[McC90] William McCune. Skolem functions and equality in automated deduction. In William Dietterich, Tom; Swartout, editor, *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 246–251, Hynes Convention Centre?, July 29–August 3 1990. MIT Press.

[McC97] William McCune. Solution of the Robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, December 1997.

[Nie98] Robert Nieuwenhuis. Decidability and complexity analysis by basic paramodulation. *Information and Computation*, 147:1–21, 1998. Extended abstract in IEEE LICS'96.

[NR95] Robert Nieuwenhuis and Albert Rubio. Theorem Proving with Ordering and Equality Constrained Clauses. *Journal of Symbolic Computation*, 19(4):321–351, April 1995.

[NR97] Robert Nieuwenhuis and Albert Rubio. Paramodulation with Built-in AC-Theories and Symbolic Constraints. *Journal of Symbolic Computation*, 23(1):1–21, May 1997.

[NR99a] Robert Nieuwenhuis and José Miguel Rivero. Solved forms for path ordering constraints. In P. Narendran and M. Rusinowitch, editors, *Tenth International Conference on Rewriting Techniques and Applications (RTA)*, LNCS, Trento, Italy, July 2–4, 1999. Springer-Verlag.

[NR99b] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers (to appear), 1999.

[PS81] G.E. Peterson and M.E. Stickel. Complete sets of reductions for some equational theories. *Journal Assoc. Comput. Mach.*, 28(2):233–264, 1981.

[RV95] Michael Rusinowitch and Laurent Vigneron. Automated deduction with associative commutative operators. *J. of Applicable Algebra in Engineering, Communication and Computation*, 6(1):23–56, 1995.

[RW69] G. A. Robinson and L. T. Wos. Paramodulation and theorem-proving in first order theories with equality. *Machine Intelligence*, 4:135–150, 1969.

[Vig94] Laurent Vigneron. Associative Commutative Deduction with constraints. In Allan Bundy, editor, *12th International Conference on Automated Deduction*, LNAI 814, pages 530–544, Nancy, France, June 1994. Springer-Verlag.

[Wei97] Christoph Weidenbach. SPASS—version 0.49. *Journal of Automated Reasoning*, 18(2):247–252, April 1997.