A survey of some recent trends in rewrite-based and paramodulation-based deduction (invited paper)

Robert Nieuwenhuis*

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

1 Introduction

Deduction with equality is fundamental in mathematics, logics and many applications of formal methods in computer science. During the last two decades this field has importantly progressed through new Knuth-Bendix-like completion techniques and their extensions to ordered paramodulation for first-order clauses. These techniques have lead to important results on deduction in first-order logic with equality, like [HR91,BDH86,BD94,BG94,BG98,NR01], results that have been applied to state-of-the-art theorem provers like Spass [Wei97] and Vampire [RV01]. These techniques have also led to results on logic-based complexity and decidability analysis [BG01,Nie98], on deduction with *constrained clauses* [KKR90,NR95], on inductive theorem proving [CN00], and on many other applications like symbolic constraint solving, or equational-logic programming. In the handbook chapter [NR01] the fundamental techniques in this area are reviewed and presented in a uniform framework.

However, in the last two years several interesting developments have taken place that are not explained in [NR01] or that are only mentioned in one or two lines added in its last revision. In this paper, we survey some of these developments (some of the ones the author has been involved in himself, and a few others the author knows well enough; this survey is not meant to be exhaustive).

After giving a small amount of technical background in Section 2 (in order to be self-contained; all details can be found in [NR01]), in Section 3 we explain some new techniques for proving the completeness of paramodulation and Knuth-Bendix completion parameterized by orderings fulfilling less requirements than the usual ones. In Section 4 we give a short overview of new techniques for building algebraic knowledge like abelian groups into paramodulation-based provers, and in Section 5 we mention some of the symbolic constraint solving techniques needed for this purpose. Finally, in Section 6 we outline some recent trends for term indexing, which is well known to be a crucial ingredient for building high-performance provers.

* Partially supported by Spanish CICYT project HEMOSS ref. TIC98-0949-C02-01.

2 Background

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,NR01].

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].

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

[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*.

single one representing the whole class. This leads to the notion of building-in equational theories E, where no inferences with the axioms of E are needed, 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 per inference, 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 $mgu_{AC}(f(x,x,x), f(y_1,y_2,y_3,y_4))$.

2.4 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 several CADE ATP System Competitions). There is also a very close relationship between the finite computation of saturated sets and *implicit* inductive theorem proving, that is, where the user does not need to provide explicit induction schemes [CN00].

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

3 Paramodulation with weak orderings

Up to recently, all major known completeness results for Knuth-Bendix completion and ordered paramodulation require the term ordering $\succ$ to be well-founded, monotonic and total (or extendable to a total ordering) on ground terms. All main proof techniques, like the transfinite semantic tree method [HR91], the proof ordering method [BDH86,BD94], and the model generation method [BG94] rely at some point on these requirements.

But in many practical situations these requirements are too strong. A typical situation is deduction modulo built-in equational theories E, where the existence of a total *E-compatible* reduction ordering is a very strong requirement. For example, the existence of such an ordering for the case where E consists of associativity and commutativity (AC) properties for some symbols remained open for a long time, and, once it was found, it triggered quite a number of results, like the decidability of the ground AC-word and -unification problems. Unfortunately, for many E such orderings cannot exist. For instance, when E contains an idempotence axiom $f(x,x) = x$, then if $s \succ t$, by monotonicity one should have $f(s,s) \succ f(s,t)$, which by E-compatibility implies $s \succ f(s,t)$ and hence non-well-foundedness. Here we shortly survey some results of [BGNR99] for dropping the monotonicity and/or the totality requirements (all details and proofs can be found there).

3.1 Ordered Paramodulation

Bachmair and Ganzinger's model generation technique proves the refutation completeness of an inference system I by showing that, if a set of clauses S is closed under I, and the empty clause is not in S, then S has a model that can be expressed by a (possible infinite) ground term rewrite system (GTRS) R that is effectively *generated* form S. In [BGNR99] a variant of this technique is used, with the main difference that the termination of the GTRS R is not a consequence of the ordering. Instead, termination of R follows from other properties. In one possible setting it follows from the irreducibility, w.r.t. R itself, of the right hand sides of all rules of R (any GTRS R with irreducible right hand sides terminates). Here we state the main results for the purely equational case only.

Theorem 1. *Let E be a set of equations that is closed under ordered paramodulation w.r.t. a* west *ordering $\succ$ (a well-founded total ordering on ground terms satisfying the subterm property). Let R be the rewrite model generated from E w.r.t. $\succ$ as in [BG94], with the additional requirement that only rules with irreducible right-hand sides are generated. Then R is convergent and the congruence generated by R is a model of E.*

In another setting, if R is contained in a well-founded ordering $\succ$ with the subterm property, its termination follows from the irreducibility of the right hand sides at non-topmost positions only. Since each terminating TRS R induces a reduction (i.e., well-founded, monotonic) ordering $\rightarrow_R^+$, one can then use induction on (an extension of) this reduction ordering for proving the main completeness results.

Theorem 2. *Let E, $\succ$ and R be as in the previous theorem, except that now inferences at the topmost position of small sides are not needed, and rules of R need not be irreducible at the topmost positions of right hand sides. Then R is convergent and the congruence generated by R is a model of E.*

These results, among other applications, open the door to deduction modulo many more classes of equational theories. The only properties required for $\succ$ are well-foundedness and the subterm property. This solves a well-known open problem (e.g. at the RTA list of open problems [RTA01] since 1995). Current work by Bofill and Rubio aims at eliminating all requirements on the ordering except well-foundedness.

3.2 Knuth–Bendix completion with only a reduction ordering

Another well-known open problem (e.g. posed by N. Dershowitz and J-P. Jouannaud on the RTA list of open problems [RTA01] since its creation in 1991) that is solved in [BGNR99] concerns the Knuth-Bendix completion procedure. The aim of this procedure is to build a convergent rewrite system R for a given a set of equations E and an ordering $\succ$ on terms (see [DJ90] for details). In fact, all current state-of-the-art theorem provers in pure equational logic, like Waldmeister [HBVL97], are based on variations of the Knuth-Bendix completion procedure. When given an ordering $\succ$ that can be extended to a total reduction ordering on ground terms, this procedure is complete in the sense that it always finds a (possibly infinite) convergent TRS R, logically equivalent to E, contained in $\succ$. This is very useful for automatically proving that an equation $s \simeq t$ follows from a set of equations E, because after a finite number of steps of such a procedure always a TRS R is reached by which a rewrite proof for $s \simeq t$ exists.

But a rewrite system already terminates if $\succ$ is only a reduction ordering (in fact, a rewrite system terminates if, and only if, it is contained in a reduction ordering). Hence a very natural (frequently asked) question is: what happens if we apply completion with a given desirable orientation by a reduction ordering that cannot be extended to a total one, like $f(a) \to f(b)$ and $g(b) \to g(a)$, for which a and b must be incomparable in any monotonic extension? It turns out that this question can be answered affirmatively: a Knuth-Bendix completion procedure can be defined that finds a convergent TRS for a given set of equations E and a (possibly non-totalizable) reduction ordering $\succ$ whenever it exists. Note that for arbitrary reduction orderings it does not always exist: in each E-congruence class there should be a single minimal element. For example, if $E = \{a \simeq b, a \simeq c\}$ then one of a, b and c should be smaller than the other two w.r.t. the given ordering.

The following lemma strengthens the uniqueness result of [DMT88] to TRS included in a west ordering, but only for the ground case:

Lemma 1. *Let $\succ$ be a west ordering, and let R_1 and R_2 be interreduced TRS, both included in $\succ$, and such that $\leftrightarrow^*_{R_1}=\leftrightarrow^*_{R_2}$. Then $R_1 = R_2$.*

It is essential for proving the following theorem:

Theorem 3. *Let E' be the closure of E under equational ordered paramodulation w.r.t. a west ordering $\succ$ extending a give reduction ordering $\succ_r$, and let R denote the (unique) canonical TRS for E and $\succ_r$. Then $E' \supseteq R$.*

4 Building in abelian groups

Paramodulation with built-in abelian groups (AG) has been investigated by many authors [Che86,Mar94,Mar96,GW96,Wal98,Wal99,Stu98]. This is not surprising since abelian groups are of course ubiquitous in many applications of (semi-)automated reasoning. But building-in AG is also attractive for at least two more reasons.

On the one hand, due to the fact that diophantine equation solving is easier in the integers than in the natural numbers, AG unification is easier than AC and AC1 (i.e., abelian monoid) unification. If all free symbols are constants, then there is one single most general AG unifier and the decision problem is polynomial, whereas for AC and AC1 the decision problems are NP-complete, and for AC there are exponentially many unifiers. Although with arbitrary free symbols the decision problem is NP-complete in all three cases, AG unification behaves better in practice. Also the number of unifiers is usually much smaller and not doubly exponential as for AC (see [BS93,BS01] for surveys on these results).

Another aspect that makes building-in AG attractive is called *symmetrization* (e.g., by Le Chenadec in [Che86]): modulo abelian groups $(+, -, 0)$, every ground equation can be written as $u+\ldots+u \simeq t$, where the summand u is greater (w.r.t. the given term ordering $\succ$) than the summands in t. As we will see, this allows one to restrict inferences to this maximal summand and to avoid the prolific inferences with extended equations that appear in the AC case.

Symmetrization is also exploited in Marché's framework for Knuth-Bendix completion of unit equations with built-in theories (ranging from AC to commutative rings) [Mar94,Mar96]. His completion procedure decides the ground word problem modulo AG by building a finite convergent rewrite system. However, his procedure is not refutation complete for equations with variables: in many cases it fails since it cannot handle symmetrization at the non-ground level.

4.1 Recent work

Full first-order clauses are considered by Ganzinger and Waldmann in [GW96] [Wal97], where symmetrization is also central. This work focuses not on AG, but on the more general theory of cancellative abelian monoids. It applies AC1 unification and *abstraction* rules, which, roughly, turn clauses like $C \vee f(s) \simeq t$ into $C \vee x \not\simeq s \vee f(x) \simeq t$, where x is a new variable; this of course increases the number of possible inferences on f. By specializing to torsion-free divisible abelian groups, AC-unification and inferences into variables can be avoided, but abstraction remains necessary [Wal98,Wal99]. These results have in turn been extended in [Wal01] to superposition and chaining for totally ordered divisible abelian groups.

In Stuber's work on paramodulation for abelian groups represented as integer modules [Stu98], symmetrization is again crucial, but AG unification is not applied. Instead, AC unification is used, and hence paramodulation inferences with the AG axioms on the remaining clauses are needed. For example, refuting a clause like $f(-b+x+a) \not\simeq f(0)$ requires inferences with the AG axioms, instead of directly finding the contradictory instance $b-a$ for x by AG-unification. Technically, even for the ground case, his inference rules and proofs are rather involved. In Stuber's PhD. Thesis [Stu99], proofs for the ground case are given in a uniform framework for AG and several other commutative theories.

In [GN00] AG-unification is used instead of the computationally more expensive unification modulo AC. Furthermore, no inferences with the AG axioms or abstraction rules are needed; in this sense this is the first approach where AG is completely built in. For obtaining these results, the model generation technique is applied by rewriting, modulo associativity and commutativity of $+$, with the well-known convergent rewrite system R_{AG} for AG, plus a set of ground rewrite rules R that consists of symmetrized rules $u + \ldots^{n)} + u \to t$ and their *inverse* version $-u \to u + \ldots^{n-1)} + u - t$. Hence $\succ$ has to be an AC-compatible reduction ordering orienting these rules, which can be fulfilled by simple general-purpose orderings like RPO (this was already mentioned by Marché). This gives relatively simple completeness proofs for full first-order ground clauses, from which it is easy to obtain a decision procedure for the satisfiability of arbitrary sets of ground clauses modulo AG.

For completely building-in AG at the non-ground level, and hence avoiding all inferences with the AG axioms by applying AG-unification, the main problem is: how to lift, to inferences on non-ground clauses C, the rewrite steps with $R \cup R_{AG}$ on ground instances $C\sigma$? The steps with R indeed become inferences, but for the steps with R_{AG} this is precisely what we want to avoid. The key ideas to the solution are roughly as follows. One keeps non-ground clauses C fully simplified w.r.t. R_{AG} (which is a cheap and useful simplification anyway). Furthermore, in the completeness proofs one considers instances with reduced substitutions σ (extending some ideas from the *basic* superposition approach [NR95,BGLS95]). Some steps with R_{AG} may then still be needed in $C\sigma$ at the frontier between C and σ. But a careful analysis of these steps reveals that they can be covered by considering inferences with AG-unification on adequate subterms. The AG-superposition inference rules have strong ordering restrictions implying that inferences only need to involve the maximal summands of the clause. This generalises standard superposition: summands play the role of terms.

5 Constraint solving for abelian semigroups, monoids and groups

As said in Section 2, in the context of built-in algebraic theories E, ordering constraint inheritance has the advantage that in each inference only one conclusion is generated, instead of one conclusion for each E-unifier. But, probably due to the lack of adequate orderings and constraint solving algorithms, these ideas have not been put into practice yet. For example, McCune found his well-known AC-paramodulation proof of the Robbins conjecture [McC97] by still computing complete sets of AC-unifiers, and adding one new equation for each one of them (although heuristics were used to discard some of the unifiers).

Indeed, of the many, rather complex, AC-compatible reduction orderings that have been defined in the literature, until recently only for the AC-RPO ordering of [RN95] a constraint solving algorithm was defined [CNR95]. But, unfortunately, this algorithm is far from practical due to its conceptual and computational complexity, and moreover, it only deals with extended signature semantics.

However, in many practical cases one has to deal with only one single associative and commutative symbol, and then a simple version of the recursive path ordering (RPO; [Der82]) on *flattened* terms, which we will call FRPO, fulfills all requirements. The same FRPO can be used as an ingredient for an AG-compatible reduction ordering AG-RPO that satisfies all requirements of [GN00], by using it to compare AG-normal forms of ground terms. Finally, it turns out that an AC0-compatible ordering AC0-RPO is obtained in a similar way by considering normal forms w.r.t. the rule $x + 0 \rightarrow x$.

In the recent paper [GN01] a uniform technique was introduced providing the first constraint solving algorithms for fixed signature semantics for AC compatible orderings. More precisely, NP algorithms for FRPO-based orderings for abelian semigroups, abelian monoids and abelian groups. These new techniques could possibly lead to reasonably efficient practical algorithms for these orderings, and give new insights for the development of constraint solving methods over fixed signatures for other E-compatible orderings. The results of [GN01] are obtained by a sequence of satisfiability-preserving transformations. Let us consider first the AC-RPO ordering. First the constraint is turned into a disjunction of certain linear constraints. Then the so-called *segments* are defined: subconstraints between free symbols. Then a *splitting* transformation is applied with the aim that solutions exist with a single summand of each local segment only. Finally, for each split constraint S a system of diophantine (in)equations D_S is produced such that S is satisfiable iff D_S is satisfiable over $\mathbb{N}^+$, which produces the result that FRPO-constraint satisfaction is in NP. Very similar techniques for AC0-RPO. The AG-RPO case is technically quite involved, because terms can be smaller than their subterms, and because after splitting solutions may still need several summands in the same segment (although all these summands come from the local segment). The key idea there is that one can bound number of *decisive* summands and extend the splitting transformation and D_S accordingly (technically complex). Then again S is satisfiable iff D_S is satisfiable over $\mathbb{N}^+$ which gives the result that AG-RPO constraint satisfaction is in NP.

6 Term indexing

Indexing data structures have a crucial impact on the performance of theorem provers. The indexes contain a large number of terms, and support the fast retrieval, for any given *query* term t, of all terms in the index satisfying a certain relation with t, like matching, unifiability, or syntactic equality. Apart form retrieval speed, another aspect that is becoming more and more crucial is memory consumption. During the last years processor speed has been growing much faster than memory capacity and one may assume that this gap will become even wider in the coming years. At the same time memory access bandwidth is also becoming an important bottleneck. Excessive memory consumption leads to more cache faults, which become the dominant factor for time, instead of processor speed.

One important aspect makes indexing techniques in theorem proving essentially different from indexing in other contexts like functional or logic programming: the index is subject to insertions and deletions. Therefore, along the last two decades a significant number of results on new specific indexing techniques for theorem proving have been published and applied in different provers. The currently best-known and most frequently used indexing techniques for matching are *discrimination trees* [Chr93,McC92], the compiled variant of discrimination trees, called *code trees* [Vor95], and *substitution trees* [Gra95].

Discrimination trees are like tries where terms are viewed as strings and where common prefixes are shared. A substitution tree has, in each node, a substitution, a list of pairs $x_i \mapsto t$ where each x_i is an *internal* variable and t is a term that may contain other internal variables as well as *external* variables which are the variables in the terms to be stored.

Example 1. The two terms $f(a, g(x), h(y))$ and $f(h(b), g(y), h(y))$ will be stored in a substitution tree and discrimination tree, respectively, as shown:

substitution tree:

$$\begin{array}{ccc} & x_0 = f(x_1, g(x_2), h(y)) & \\ \swarrow & & \searrow \\ x_1 = a,\ x_2 = x & & x_1 = h(b),\ x_2 = y \end{array}$$

discrimination tree:

$$\begin{tikzcd}[column sep=small] & f \arrow[dl] \arrow[dr] & \\ a \arrow[d] & & h \arrow[d] \\ g \arrow[d] & & b \arrow[d] \\ \vdots & & \vdots \end{tikzcd}$$

In a substitution tree all terms $x_0\sigma$ are stored such that σ is the composition of the substitutions on some path from the root to a leaf of the tree. Inserting the first term in an empty substitution tree we obtain the single node $x_0 = f(a, g(x), h(y))$. When inserting the second term, internal variables are placed at the points of disagreement, and children are created with the "remaining" substitutions of both. Therefore all common contexts can be shared. □

Example 2. It clear that the additional sharing in substitution trees avoids repeated work (which is the key idea of all indexing techniques). Assume one has two terms $f(c, x, t)$ and $f(x, c, t)$ in the index, and a query $f(c, c, s)$, where s and

t are terms such that s is not an instance of t. Then two attempts to match s against t will be made in a discrimination tree, and only one in a substitution tree. But, on the other hand, in substitution trees the basic traversal algorithms are significantly more costly. □

In [GNN01] a new indexing data structure is described, called *context trees*, where, by means of a limited kind of context variables, certain common subterms can be shared, even if they occur below different function symbols. Roughly, the idea is that $f(s)$ and $g(s,t)$ can be represented as $F(s,t)$, with children $F = f$ and $F = g$. Function variables such as F stand for single function symbols only (although extensions to allow for more complex forms of second-order terms are possible).

Example 3. Assume one has three terms $h(x, f(t))$, $h(x, g(t))$, and $h(b, f(t))$ in the index. Then, in a discrimination tree, t will occur three times. In a substitution tree, we will have:

$$
\begin{array}{ccccc}
 & & x_0 = h(x_1, x_2) & & \\
 & / & & \backslash & \\
 & x_1 = x & & & x_1 = b, x_2 = f(t) \\
 / & & \backslash & & \\
x_2 = f(t) & & x_2 = g(t) & &
\end{array}
$$

and with a query $h(b, f(s))$, the terms s and t will be matched twice against each other (at the leftmost and rightmost leaves). In a context tree, the term t occurs only once:

$$
\begin{array}{ccccc}
 & & x_0 = h(x_1, F(t)) & & \\
 & / & & \backslash & \\
 & x_1 = x & & & x_1 = b, F = f \\
 / & & \backslash & & \\
F = f & & F = g & &
\end{array}
$$

and if s does not match t, the failure with the query $h(b, f(s))$ will be found at the root. □

In [NHRV01] some experimental results concerning time and space consumption are given, comparing three of the currently best implementations of term indexing techniques, as well as a discussion about how to carry out such indexing experiments. According to these experiments, in spite of the fact that the current implementation of context trees is very preliminary, it is already quite competitive in time. Moreover, context trees are, as expected, best in space. A substantial further space improvement can be expected from a compiled implementation.

Current work in this area is on the intelligent computation of backtracking points at update time of the index, as well as on the compilation of indexing data structures.

References

[BD94] Leo Bachmair and Nachum Dershowitz. Equational inference, canonical proofs, and proof orderings. *J. of the Association for Computing Machinery*, 41(2):236–276, February 1994.

[BDH86] Leo Bachmair, Nachum Dershowitz, and Jieh Hsiang. Orderings for equational proofs. In *First IEEE Symposium on Logic in Computer Science (LICS)*, pages 346–357, Cambridge, Massachusetts, USA, June 16–18, 1986. IEEE Computer Society Press.

[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.

[BG01] Basin and Ganzinger. Automated complexity analysis based on ordered resolution. *JACM: Journal of the ACM*, 48, 2001.

[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)*, pages 225–233, Trento, Italy, July 2–5, 1999.

[BS93] Franz Baader and Jörg Siekmann. Unification theory. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming.* Oxford University Press, 1993.

[BS01] Franz Baader and Wayne Snyder. Unification theory. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning.* Elsevier Science Publishers and MIT Press(to appear), 2001.

[Che86] Philippe Le Chenadec. *Canonical Forms in Finitely Presented Algebras.* Research Notes in Theoretical Computer Science. Pitman, London; Wiley, New York, Toronto, 1986.

[Chr93] Jim Christian. Flatterms, Discrimination Nets, and Fast Term rewriting. *Journal of Automated Reasoning*, 10:95–113, 1993.

[CN00] H. Comon and R. Nieuwenhuis. Induction = I-axiomatization + first-order consistency. *Information and Computation*, 2000. To appear.

[CNR95] Hubert Comon, Robert Nieuwenhuis, and Albert Rubio. Orderings, AC-Theories and Symbolic Constraint Solving. In *10th IEEE Symposium on Logic in Computer Science (LICS)*, pages 375–385, San Diego, USA, June 26–29, 1995.

[Der82] Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.

[DJ90] Nachum Dershowitz and Jean-Pierre Jouannaud. Rewrite systems. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 6, pages 244–320. Elsevier Science Publishers B.V., Amsterdam, New York, Oxford, Tokyo, 1990.

[DMT88] Nachum Dershowitz, Leo Marcus, and Andrzej Tarlecki. Existence, uniqueness, and construction of rewrite systems. *SIAM Journal on Computing*, 17(4):629–639, 1988.

[GN00] Guillem Godoy and Robert Nieuwenhuis. Paramodulation with built-in abelian groups. In *15th IEEE Symposium on Logic in Computer Science (LICS)*, pages 413–424, Santa Barbara, USA, June 26–29, 2000. IEEE Computer Society Press.

[GN01] Guillem Godoy and Robert Nieuwenhuis. Ordering Constraints for Deduction with Built-in Abelian Semigroups, Monoids and Groups. In *16th IEEE Symposium on Logic in Computer Science (LICS)*, pages xx–xx, Boston, USA, June 16–20, 2001. IEEE Computer Society Press.

[GNN01] Harald Ganzinger, Robert Nieuwenhuis, and Pilar Nivela. Context trees. In Leitsch Goré and Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2038 of *LNAI*, pages 242–256, Siena, Italy, 2001. Springer-Verlag.

[Gra95] Peter Graf. Substitution Tree Indexing. In J. Hsiang, editor, *6th RTA*, LNCS 914, pages 117–131, Kaiserslautern, Germany, April 4–7, 1995. Springer-Verlag.

[GW96] Harald Ganzinger and Uwe Waldmann. Theorem proving in cancellative abelian monoids. In M. A. McRobbie and J. K. Slaney, editors, *Thirteenth International Conference on Automated Deduction (CADE)*, volume 1104 of *LNAI*, pages 388–402, New Brunswick, 1996. Springer.

[HBVL97] Thomas Hillenbrand, Arnim Buch, Roland Vogt, and Bernd Löchner. WALDMEISTER—high-performance equational deduction. *Journal of Automated Reasoning*, 18(2):265–270, April 1997.

[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, July 1991.

[KKR90] Claude Kirchner, Hélène Kirchner, and Michaël Rusinowitch. Deduction with symbolic constraints. *Revue Française d'Intelligence Artificielle*, 4(3):9–52, 1990.

[Mar94] Claude Marché. Normalised Rewriting and Normalised Completion. In *Ninth Annual IEEE Symposium on Logic in Computer Science*, Paris, France, July 1994. IEEE Computer Society Press, Los Alamitos, CA, USA.

[Mar96] Claude Marché. Normalized rewriting: An alternative to rewriting Modulo a set of equations. *Journal of Symbolic Computation*, 21(3):253–288, March 1996.

[McC92] William McCune. Experiments with discrimination tree indexing and path indexing for term retrieval. *Journal of Automated Reasoning*, 9(2):147–167, October 1992.

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

[NHRV01] Robert Nieuwenhuis, Thomas HillenBrand, Alexandre Riazanov, and Andrei Voronkov. On the evaluation of indexing techniques for theorem proving. In Leitsch Goré and Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2038 of *LNAI*, pages 257–271, Siena, Italy, 2001. Springer-Verlag.

[Nie98] Robert Nieuwenhuis. Decidability and complexity analysis by basic paramodulation. *Information and Computation*, 147:1–21, 1998.

[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.

[NR01] Robert Nieuwenhuis and Albert Rubio. Paramodulation-based theorem proving. In J.A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers and MIT Press, 2001.

[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.

[RN95] Albert Rubio and Robert Nieuwenhuis. A total AC-compatible ordering based on RPO. *Theoretical Computer Science*, 142(2):209–227, May 15, 1995.

[RTA01] RTA-LOOP. Int. Conf. on Rewriting Techniques and Applications, The list of open problems, 2001. Maintained at `http://www.lri.fr/~rtaloop/` (by R. Treinen).

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

[RV01] Alexandre Riazanov and Andrei Voronkov. Vampire 1.1. In Leitsch Goré and Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2038 of *LNAI*, pages 376–380, Siena, Italy, 2001. Springer-Verlag.

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

[Stu98] Jürgen Stuber. Superposition theorem proving for abelian groups represented as integer modules. *Theoretical Computer Science*, 208(1–2):149–177, 28 November 1998.

[Stu99] Jürgen Stuber. *Superposition Theorem Proving for Commutative Algebraic Theories*. Dissertation, Technische Fakultät, Universität des Saarlandes, Saarbrücken, Germany, 1999. Submitted.

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

[Vor95] Andrei Voronkov. The Anatomy of Vampire. Implementing Bottom-up Procedures with Code Trees. *Journal of Automated Reasoning*, 15(2):237–265, October 1995.

[Wal97] Uwe Waldmann. *Cancellative Abelian Monoids in Refutational Theorem Proving*. Dissertation, Universität des Saarlandes, Saarbrücken, Germany, 1997.

[Wal98] Uwe Waldmann. Superposition for divisible torsion-free abelian monoids. In Claude Kirchner and Hélène Kirchner, editors, *Proceedings of the Fifteenth International Conference on Automated Deduction (CADE-98)*, volume 1421 of *LNAI*, pages 144–159, Lindau, Germany, 1998. Springer.

[Wal99] Uwe Waldmann. Cancellative superposition decides the theory of divisible torsion-free abelian groups. In *Logic Programming and Automated Reasoning, Int. Conf.*, LNAI 1705, pages 131–147, Tbilisi, Georgia, September 1999. Springer-Verlag.

[Wal01] Uwe Waldmann. Superposition and chaining for totally ordered divisible abelian groups. In Leitsch Goré and Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2001)*, volume 2038 of *LNAI*, pages 226–241, Siena, Italy, 2001. Springer-Verlag.

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