CHAPTER 1

Paramodulation-based theorem proving

Robert Nieuwenhuis and Albert Rubio

Technical University of Catalonia
Barcelona, Spain

Contents

HANDBOOK OF AUTOMATED REASONING
Edited by Alan Robinson and Andrei Voronkov

1. About this chapter

The aim of this chapter is to review the fundamental techniques in paramodulation-based theorem proving, presenting them in a uniform framework. We start with easier subcases and progressively include the different extensions. Since the objective is to obtain a concise overview of the current state of the art, some of the historical developments that are not essential for the current results are omitted (further historical remarks on paramodulation are given in Chapter [chapter with id sequent] of this handbook).

In this first section, the main concepts are introduced in an informal way, with emphasis on their intuitive background. This is done to facilitate the reading of subsequent sections, where all these notions are formally defined and explained in detail, and some of the main results are proved.

1.1. Paramodulation

Paramodulation originated as a development of resolution [Robinson 1965], one of the main computational methods in first-order logic (see Chapter [chapter with id resolution] of this handbook). For improving resolution-based methods, the study of the equality predicate has been particularly important, since reasoning with equality is well-known to be of great importance in mathematics, logic, and computer science. Robinson [1965] showed that resolution together with factoring is *refutation complete*, that is, the empty clause will eventually be inferred by systematically enumerating all consequences of an unsatisfiable set of clauses by *(binary) resolution:*

$$\frac{C \vee A \qquad D \vee \neg B}{(C \vee D)\sigma} \qquad \text{if } \sigma = mgu(A, B)$$

where $mgu(A, B)$ denotes a most general unifier of A and B, and *factoring*:

$$\frac{C \vee A \vee B}{(C \vee A)\sigma} \qquad \text{if } \sigma = mgu(A, B)$$

For dealing with the equality predicate $\simeq$ by resolution, one can specify it by means of the following *congruence axioms* $\mathcal{E}$:

$$\begin{array}{rll}
 & \rightarrow x \simeq x & \textit{(reflexivity)} \\
x \simeq y & \rightarrow y \simeq x & \textit{(symmetry)} \\
x \simeq y \wedge y \simeq z & \rightarrow x \simeq z & \textit{(transitivity)} \\
x_1 \simeq y_1 \wedge \ldots \wedge x_n \simeq y_n & \rightarrow f(x_1, \ldots, x_n) \simeq f(y_1, \ldots, y_n) & \textit{(monotonicity-I)} \\
x_1 \simeq y_1 \wedge \ldots \wedge x_n \simeq y_n & & \\
\wedge P(x_1, \ldots, x_n) & \rightarrow P(y_1, \ldots, y_n) & \textit{(monotonicity-II)}
\end{array}$$

In fact the monotonicity axioms are axiom *schemes*: one *monotonicity-I* axiom is required for each non-constant n-ary function symbol f, and, similarly, one *monotonicity-II* axiom for each predicate symbol P. A set S of clauses is satisfiable in first-order logic with equality if, and only if, $S \cup \mathcal{E}$ is satisfiable in first-order logic without equality[1].

However, it is easy to see that resolution and factoring with $\mathcal{E}$ tend to cause the generation of too many (mostly unnecessary) new clauses. Therefore, Robinson and Wos explored another possibility. They tried to avoid the need for specifying equality by treating it as part of the logical language, i.e., directly considering first-order logic with equality. This requires the design of dedicated inference rules, like *paramodulation* [Robinson and Wos 1969]:

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

where $D|_p$ is the subterm of D at position p, and $D[t]_p$ denotes the result of replacing in D this subterm by t. Paramodulation, together with resolution and factoring, was proved refutation complete, under the presence of the reflexivity axiom and certain tautologies called the *functional reflexivity* axioms

$$f(x_1, \ldots, x_n) \simeq f(x_1, \ldots, x_n)$$

for every n-ary function symbol f of the alphabet. Later on, Brand [Brand 1975] proved that the functional reflexivity axioms were unnecessary, as well as paramodulation *into variables*, that is, paramodulations where $D|_p$ is a variable. However, even under these restrictions, paramodulation is difficult to control: unless additional refinements are considered, it quickly produces a large amount of unnecessary clauses, expanding the search space excessively.

The strengths and weaknesses of paramodulation have led to fruitful theoretical and practical research on paramodulation-based theorem proving. Concerning the practical research, a large number of experiments with paramodulation have been performed at the Argonne group by Wos, Overbeek, Henschen and others (see, e.g., [Wos 1988, Wos 1996] for references), and especially by McCune with his provers Otter [McCune 1994] and EQP [McCune 1997*a*] and his recent automated proof of the Robbins conjecture [McCune 1997*b*, McCune 1997*c*]. Concerning the theoretical research, the main techniques are reviewed in this chapter, and some aspects of their implementation in practical provers is discussed.

1.2. Extending the unit equality case: ordered paramodulation

An important tool in paramodulation is the use of *term orderings* for restricting the number of inferences. Paramodulation is in fact based on Leibniz' law for replacement of equals by equals. Now the basic idea of *ordered* paramodulation is to

[1] Note that there is no logical equivalence. First-order logic (FOL) with equality has more expressive power: for instance, in FOL with equality the clause $x \simeq a \vee x \simeq b$ expresses that the cardinality of models is at most two, which cannot be expressed in FOL without equality.

only perform replacements of *big* terms by *smaller* ones, with respect to the given ordering $\succ$.

This is precisely the idea of (ordered) *rewriting*. Let us consider now unit equations: we address *word problems* of the form $E \models u \simeq v$, where E is a set of equations and $u \simeq v$ is another equation. Assume that $\succ$ is a *reduction ordering* on terms (see Section 2 for the precise definitions). A term t is rewritten in one step with an equation $l \simeq r$ (or, equivalently, $r \simeq l$) of E by replacing a subterm $l\sigma$ of t by $r\sigma$, for some substitution σ such that $l\sigma \succ r\sigma$. For example, let E consist of the equations $plus(0,x) \simeq x$ and $plus(s(x),y) \simeq s(plus(x,y))$. Denoting each step by $\rightarrow_E$ (and assuming the steps agree with $\succ$), we have

$$plus(s(s(0)),s(0)) \rightarrow_E s(plus(s(0),s(0))) \rightarrow_E s(s(plus(0,s(0)))) \rightarrow_E s(s(s(0)))$$

This (ordered) rewrite relation terminates: starting from some finite term t, after a finite number of steps a *normal form* (i.e., a term that cannot be rewritten any further) is obtained.

Now let $\rightarrow_E^*$ denote zero or more of these steps (i.e., $\rightarrow_E^*$ is the reflexive-transitive closure of the relation $\rightarrow_E$). A set of equations E is called *confluent* w.r.t. the given $\succ$ if, whenever $s \rightarrow_E^* u$ and $s \rightarrow_E^* v$, there is some t such that $u \rightarrow_E^* t$ and $v \rightarrow_E^* t$. It is not difficult to see that then every term has a unique normal form. Furthermore, rewriting is then a decision procedure for deduction in the theory of E, since $E \models s = t$ if, and only if, s and t have the same normal form[2].

The first instances of ordered paramodulation appeared in *Knuth-Bendix completion* [Knuth and Bendix 1970]. Roughly, a completion procedure attempts to transform a given set of equations into an equivalent confluent one. A crucial step of the transformation process is the computation of *critical pairs* between equations. A critical pair is an equation obtained by *superposition*, the restricted version of paramodulation in which inferences only involve left hand sides of possible rewrite steps, i.e., only the big terms (w.r.t. $\succ$) are considered. During the completion process equations are simplified by rewriting, and tautologies, i.e., equations of the form $s \simeq s$, are removed.

Note that, since the word problem is not decidable in general, a finite confluent E cannot always be obtained. In Knuth and Bendix' original procedure this could be due to *failure*[3] or to non-termination of completion. For completely avoiding failure, *ordered* or *unfailing* completion was introduced [Lankford 1975, Hsiang and Rusinowitch 1987, Bachmair, Dershowitz and Plaisted 1989].

This leads to complete theorem provers for equational theories E, since for every valid equation a rewrite proof will be found after a finite number of steps of the (possibly infinite) completion procedure. Moreover, if the process terminates, it pro-

[2]More precisely, one rewrites the ground Skolemisations of s and t, and $\succ$ is required to be total on such ground terms.

[3]Failure could occur because Knuth and Bendix considered rewriting with a terminating set of uni-directional rules, instead of ordered rewriting (applying equations in whatever direction agrees with the given reduction ordering, as explained here). Hence in their view equations had to be *oriented* into terminating rules, which fails if an equation like the commutativity axiom $f(x,y) \simeq f(y,x)$ appears.

duces a confluent system for ordered rewriting. For improving the efficiency and for reducing the number of cases of non-termination of completion, numerous additional simplification methods and *critical pair criteria* for detecting redundant inferences have been developed [Bachmair, Dershowitz and Hsiang 1986, Peterson 1990, Martin and Nipkow 1990, Bachmair 1991, Bachmair and Dershowitz 1994, Comon, Narendran, Nieuwenhuis and Rusinowitch 1998]. Indeed, nowadays completion has become the method of choice for most state-of-the-art equality reasoning systems. Since the main results for completion-based theorem proving with unit equations are particular cases of the ones given for general clauses with equality, in this chapter no further specific attention will be devoted to completion; instead, in Subsection 4.6, it will be shortly treated as an instance of saturation for general clauses.

Extending the notion of critical pair, completion procedures were developed for going beyond unit equations. For instance, for obtaining confluent sets for rewrite relations like *conditional* and *clausal* rewriting, completion procedures were designed for transforming sets of *conditional equations* (definite Horn clauses with equality, i.e., of the form $s_1 \simeq t_1 \wedge \ldots \wedge s_n \simeq t_n \rightarrow s \simeq t$) [Kaplan 1984, Jouannaud and Waldmann 1986, Kounalis and Rusinowitch 1991, Ganzinger 1991], or *restricted equality* clauses [Nieuwenhuis and Orejas 1990].

The generalization of this kind of completion procedure to full first-order clauses with equality required the development of more powerful proof techniques for establishing completeness. Using the *transfinite semantic tree* method Hsiang and Rusinowitch [1991] proved the refutation completeness of *ordered paramodulation*, while Bachmair [1989] applied an extension of the so-called *proof ordering* technique for obtaining similar results.

By means of their *model generation* proof method, similar to other *forcing* techniques developed by Zhang [1988] and Pais and Peterson [1991], Bachmair and Ganzinger [1990, 1994*b*] proved the completeness of an inference system for full first-order clauses with equality, based on *strict superposition*: paramodulation involving only maximal (w.r.t. the ordering $\succ$) terms of maximal equations of clauses. Such superposition-based inference systems, as well as the model generation method, are explained in detail in Section 3 of this chapter.

1.3. Redundancy and saturation

Knuth-Bendix completion transforms sets of equations into *complete* or *saturated* ones: sets that are closed under the addition of non-*joinable* critical pairs, where a critical pair is joinable if it can be rewritten into a tautology $s \simeq s$.

This idea of *saturation* can be generalized: a set of formulae S is saturated for a given inference system $\mathcal{I}$ if S is closed under $\mathcal{I}$, up to *redundant inferences*. Roughly, a *saturation procedure* adds conclusions of non-redundant inferences and removes *redundant formulae*. In the limit such a procedure produces a saturated set. Therefore, in the setting of first-order clauses, proving the refutation completeness of saturation amounts to showing that the empty clause $\square$ is in S for every unsatisfiable saturated set of clauses S.

Concrete simplification methods in the context of paramodulation were discussed already in [Wos, Robinson, Carson and Shalla 1967, Slagle 1974, Loveland 1978, Peterson 1983]. Bachmair and Ganzinger [1994*b*] define abstract notions of redundancy for inferences and for clauses. For example, a ground clause C is redundant with respect to a set of ground clauses S if C is a logical consequence of smaller (with respect to the given clause ordering) clauses of S. These redundancy notions cover well-known practical simplification and elimination techniques, like *demodulation* (that is, simplification by rewriting with unit equations) or *subsumption* (removing a clause of the form $C\sigma \vee D$ in the presence of a more general clause C), as well as many other more powerful methods. For establishing the refutation completeness of saturation, a model is built for every saturated set not containing the empty clause. Several of the calculi and redundancy techniques explained in this chapter are available in the *Saturate* system [Nivela and Nieuwenhuis 1993, Ganzinger, Nieuwenhuis and Nivela 1999]. In Section 4 of this chapter, saturation procedures are introduced.

1.4. Computing with finite saturated sets

Due to the refined inference rules and redundancy notions, it is sometimes possible to compute a *finite* saturated set (not containing the empty clause) for a given input. In this case its satisfiability has been proved. This kind of satisfiability proving has of course many applications and is also closely related to inductive theorem proving (see [Comon and Nieuwenhuis 2000] and Chapter [chapter with id comon] of this handbook). The Spass system [Weidenbach 1997] has successfully applied saturation to prove satisfiability for all problems in the corresponding category of the 1997 CADE theorem proving competition [Sutcliffe and Suttner 1998].

Theorem proving in theories expressed by saturated sets of axioms is also interesting because more efficient proof strategies become (refutation) complete. For instance, the set-of-support strategy, which is incomplete in general for ordered inference systems and also for equality clauses, becomes complete for saturated sets S: no inferences between clauses in S are needed. Another well-known example is the completeness of rewriting with saturated sets of unit equations: saturated sets are confluent. For sets of conditional equations E or, equivalently, of Horn clauses, similar completeness results exist for *conditional rewriting* if E fulfills some syntactic requirements (e.g., in certain clauses the maximal terms must contain all variables). In general, the more such requirements are fulfilled by the saturated sets, the more restrictive proof strategies become complete. This sometimes leads to decision procedures, like the ones by rewriting for saturated sets of (conditional) equations. Computation with saturated sets is covered in Section 4.5 of this chapter. Some decision procedures are described in Section 8.2.

1.5. Paramodulation with constrained clauses

The advantages of *constrained formulae* are nowadays widely recognized in the context of logic programming. The first ideas for specific applications to paramodulation-based theorem proving were given in [Peterson 1990, Kirchner, Kirchner and Rusinowitch 1990]. The semantics of a clause C with a constraint T, written $C \mid T$, is simply the set of all ground instances $C\sigma$ of C such that σ is a solution of T. For example, if $=$ denotes syntactic equality of terms, the constrained clause $P(x) \mid x = f(y) \ \wedge \ y > a$ denotes[4] all ground atoms $P(f(t))$ such that t is greater than a in the given term ordering $\succ$. Hence if T is unsatisfiable then $C \mid T$ is a tautology.

In [Kirchner et al. 1990] ordered paramodulation inference rules were expressed for the first time by explicitly formulating the ordering and equality restrictions of the inferences by constraints at the formula level. This gives:

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

where T and T' are the constraints inherited from the premises, the *equality constraint* $s = D|_p$ stores the unification restriction, and OC is an *ordering constraint* of the form $s > t \ \wedge \ldots$ encoding the ordering restrictions imposed by this inference. However, the completeness results of [Kirchner et al. 1990] were limited since they required to enumerate the solutions of the constraints and *propagate* (i.e., apply) these solutions to the clause part.

Constraints are closely related to the so-called *basic strategies*, where no inferences need to be computed on subterms generated in unifiers of ancestor inference steps (like its counterpart in E-unification, called *basic narrowing* [Hullot 1980*a*]). It is clear that if such an inference system with inherited constraints is applied without propagation, then it is basic: the inferences only take place on the clause part C of a formula $C \mid T$, and no unifiers are ever applied to C, since the unification restrictions are simply stored in the constraint part T.

Nieuwenhuis and Rubio [1992*a*, 1995] showed that, in the context of superposition, indeed propagation of the equality constraints is not needed, thus proving the completeness of *basic superposition*. By using *closure substitutions*, which play the role of equality constraints, the same results were obtained independently by Bachmair and others [1992, 1995], giving additional refinements based on *term selection rules* and *redex orderings*. These developments took place independently of much earlier work in Russia by Degtyarev [1979], who used *conditional clauses* (which can in fact be seen as clauses with syntactic equality constraints) for describing a form of basic paramodulation without ordering restrictions (see also [Degtyarev and Voronkov 1986]).

[4] Note that $>$ and $=$ are used as syntax in the constraint language. Their semantics will be a given term ordering $\succ$ and a given congruence (usually syntactic equality of terms) that depend on the context.

In [Nieuwenhuis and Rubio 1992*b*] it is shown that by inheriting as well the *ordering constraints* one can restrict the search space even further without losing completeness. In [Lynch and Snyder 1993] equality, disequality and irreducibility constraints are applied for obtaining more powerful redundancy methods in basic equational completion. Finally, in [Nieuwenhuis and Rubio 1995] the use of constraints in theorem proving procedures is put in a more general framework based on the notion of *constraint inheritance strategies*.

The main idea in all these strategies 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 to the current inference rule *and to the previous ones*, then the inference can be blocked. Therefore, for taking advantage of the constraints, algorithms for constraint satisfiability checking are required. In Section 7 of this chapter a short survey of the state of the art on such algorithms is given. Paramodulation with constrained clauses, the basic strategy and the corresponding completeness results are explained in detail in Sections 5.1 and 5.2.

1.6. Paramodulation with built-in equational theories

In principle, the aforementioned paramodulation methods apply to any set of clauses with equality, but in some cases special treatments for specific equational subsets of the axioms are preferable. On the one hand, some axioms generate many slightly different permuted versions of clauses, and for efficiency reasons it is many times better to treat all these clauses together as a single one representing the whole class. On the other hand, special treatments can avoid non-termination of completion procedures, like with $f(a,b) \simeq c$ in the presence of associativity and commutativity axioms for f. Also, some equations like the commutativity axiom are more naturally viewed as "structural" axioms (defining a congruence relation on terms) rather than as "simplifiers" (defining a reduction relation). This allows one to extend completion procedures in order to deal with congruence classes of terms instead of single terms, i.e., working with a *built-in* equational theory E, and performing rewriting and completion with special E-matching and E-unification algorithms.

Early results on paramodulation and rewriting *modulo* E were given by Plotkin [1972], Slagle [1974] and Lankford and Ballantine [1977] and *extended* E*-rewriting* was defined by Peterson and Stickel [1981]. Several E-completion procedures for the equational case were developed e.g. in [Lankford and Ballantyne 1977, Huet 1980, Peterson and Stickel 1981, Jouannaud 1983, Jouannaud and Kirchner 1986, Bachmair and Dershowitz 1989]. Special attention has always been devoted to the case where E includes axioms of associativity and commutativity (AC), which occur very frequently in practical applications, and are well-suited for being built in due to their permutative nature.

The generalization of these E-completion techniques to full first-order clauses with equality has been studied in e.g. [Paul 1992, Wertz 1992, Rusinowitch and

Vigneron 1995, Bachmair and Ganzinger 1994*a*], usually with particular treatments for the AC case. Paramodulation modulo E then becomes roughly the following rule, which has one conclusion for each σ in $U_E(s, D|_p)$, a *minimal complete* set of E-unifiers of $D|_p$ and s:

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

Note that in general there is no unique most general E-unifier for a given E-unification problem, and that new variables may appear: for example, if f is an AC-symbol, then $f(x, a)$ and $f(y, b)$ have the two AC-unifiers $\sigma_1 = \{x \mapsto b, y \mapsto a\}$ and $\sigma_2 = \{x \mapsto f(b, z), y \mapsto f(a, z)\}$. In Section 6 of this chapter we introduce some of the main techniques on paramodulation modulo equational theories.

1.7. Basic paramodulation with built-in equational theories

For an equational theory E, the number of E-unifiers of two terms may be large. For instance, the cardinality of a minimal complete set of AC-unifiers is doubly exponential in general [Domenjoud 1992] (in a sense, this is also an upper bound [Kapur and Narendran 1992]). Hence a single E-paramodulation inference can generate a large number of new clauses.

Therefore, equality constraints become extremely useful in this context. In constrained E-paramodulation, instead of E-unifying the terms, the unification problem is stored in the constraint. Hence in the constrained superposition inference rule given in Section 1.5, the semantics of the symbol '=' in the equality constraint $s = D|_p$ becomes E-equality. Dealing with a constrained clause $C \mid s = t$ can be much more efficient than having n clauses $C_1, \ldots, C_n$, one for each E-unifier of s and t, since many inferences are computed at once, and each inference generates one single conclusion. Furthermore, computing E-unifiers is not needed. A clause C with an E-equality constraint T can be proved redundant by means of efficient (sound, but possibly incomplete) methods for detecting unsatisfiable T. If C is the empty clause, a contradiction has been derived if, and only if, the constraint part T is satisfiable, and hence in this case refutation completeness requires a semi-decision procedure for detecting these contradictions. Such a procedure exists for every finite E.

The completeness of such a fully basic strategy for the AC-case (combined with ordering constraints) was first proved in [Nieuwenhuis and Rubio 1994, Nieuwenhuis and Rubio 1997], although the first results on (almost basic) constrained deduction methods modulo AC were reported in [Vigneron 1994]. The basicness restriction is considered to "have been a key strategy" by McCune [1997*b*] in his celebrated AC-paramodulation-based proof of the Robbins problem. In Section 6.3 of this chapter basic paramodulation modulo AC is explained.

2. Preliminaries

In order to keep this chapter self-contained, here we introduce the main basic tools used: terms, rewriting, term orderings, first-order equality clauses and equality Herbrand interpretations. Most (if not all) of our definitions are consistent with Chapter [chapter with id rewriting] of this handbook.

2.1. *Terms and (rewrite) relations*

Let $\mathcal{F}$ be a *signature*, a (finite) set of function symbols with an arity function $arity: \mathcal{F} \to I\!N$ and let $\mathcal{X}$ be a set of variable symbols. Function symbols f with $arity(f) = n$ are called *n-ary* symbols (when $n = 1$, one says *unary* and when $n = 2$, *binary*). If $arity(f) = 0$, then f is a *constant symbol*. The set of *first-order terms* over $\mathcal{F}$ and $\mathcal{X}$, denoted by $\mathcal{T}(\mathcal{F}, \mathcal{X})$, is the smallest set containing $\mathcal{X}$ such that $f(t_1, \ldots, t_n)$ is in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ whenever $f \in \mathcal{F}$, $arity(f) = n$, and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Similarly, $\mathcal{T}(\mathcal{F})$ is the set of variable free or *ground* terms. Note that $\mathcal{T}(\mathcal{F}) = \emptyset$ if there are no constant symbols in $\mathcal{F}$. As usual, along this chapter it is therefore assumed that there is at least one constant symbol in $\mathcal{F}$.

A *position* is a sequence of positive integers. If p is a position and t is a term, then by $t|_p$ we denote the *subterm of t at position p*: we have $t|_\lambda = t$ (where λ denotes the empty sequence) and $f(t_1, \ldots, t_n)|_{i.p} = t_i|_p$ if $1 \leq i \leq n$ (and is undefined if $i > n$). We also write $t[s]_p$ to denote the term obtained by replacing in t the subterm at position p by the term s. For example, if t is $f(a, g(b, h(c)), d)$, then $t|_{2.2.1} = c$, and $t[d]_{2.2} = f(a, g(b, d), d)$. We say that a a variable (or function symbol) x *occurs* (at position p) in a term t if $t|_p$ is (rooted by) x. By $vars(t)$ we denote the set of all variables occurring in t. If t is a term of the form $f(t_1, \ldots, t_n)$, then we define $top(t)$ to be the function symbol f. The syntactic equality of two terms s and t will be denoted by $s \equiv t$.

A *substitution* σ is a mapping from variables to terms. It can be extended to a function from terms to terms in the usual way: using a postfix notation, $t\sigma$ denotes the result of simultaneously replacing in t every $x \in Dom(\sigma)$ by $x\sigma$. Here substitutions are sometimes written as sets of pairs $x \mapsto t$, where x is a variable and t is a term. For example, if σ is $\{x \mapsto f(b, y), y \mapsto a\}$, then $g(x, y)\sigma$ is $g(f(b, y), a)$ (this example illustrates the *simultaneous* replacement: applying σ "from left to right" yields $g(f(b, a), a)$, which is not the intended meaning).

A substitution σ is *ground* if its range is $\mathcal{T}(\mathcal{F})$. Unless stated otherwise, we will assume that ground substitutions σ applied to a term t are also *grounding*, that is, $vars(t) \subseteq Dom(\sigma)$, and hence $t\sigma$ is ground. A term t *matches* a term s if $s\sigma \equiv t$ for some σ. Then t is called an *instance* of s.

A term t is *unifiable* with a term s if $s\sigma \equiv t\sigma$ for some substitution σ. Then σ is called a *unifier* of s and t. Furthermore, a substitution σ is called a *most general unifier* of s and t, denoted $mgu(s, t)$, if $s\sigma \equiv t\sigma$, and for every other unifier θ of s and t, it holds that $s\theta \equiv s\sigma\sigma' \equiv t\theta \equiv t\sigma\sigma'$ for some σ', that is, roughly, if every other unifier θ is a particular instance of σ. We sometimes speak about *the mgu* of

s and t because it is unique up to variable renaming (see Chapter [chapter with id unification] of this handbook for details and for unification algorithms computing mgu's).

A *multiset* over a set S is a function $M: S \to \mathbb{N}$. The union and intersection of multisets are defined as usual by $M_1 \cup M_2(x) = M_1(x) + M_2(x)$, and $M_1 \cap M_2(x) = min(M_1(x), M_2(x))$. We also use a set-like notation: $M = \{a, a, b\}$ denotes $M(a) = 2$, $M(b) = 1$, and $M(x) = 0$ for $x \not\equiv a$ and $x \not\equiv b$. A multiset M is *empty* if $M(x) = 0$ for all $x \in S$.

If $\to$ is a binary relation, then $\leftarrow$ is its inverse, $\leftrightarrow$ is its symmetric closure, $\to^+$ is its transitive closure and $\to^*$ is its reflexive-transitive closure. We write $s \to^! t$ if $s \to^* t$ and there is no t' such that $t \to t'$. Then t is called *irreducible* and a *normal form* of s (w.r.t. $\to$). The relation $\to$ is *well-founded* or *terminating* if there exists no infinite sequence $s_1 \to s_2 \to \ldots$ and it is *confluent* or *Church-Rosser* if the relation $\leftarrow^* \circ \to^*$ is contained in $\to^* \circ \leftarrow^*$. It is *locally confluent* if $\leftarrow \circ \to \;\subseteq\; \to^* \circ \leftarrow^*$. By Newman's lemma, terminating locally-confluent relations are confluent. A relation $\to$ on terms is *monotonic* if $s \to t$ implies $u[s]_p \to u[t]_p$ for all terms s, t and u and positions p. A *congruence* is a reflexive, symmetric, transitive and monotonic relation on terms.

An *equation* is a multiset $\{s, t\}$, denoted $s \simeq t$ or, equivalently, $t \simeq s$. A *rewrite rule* is an ordered pair (s, t), written $s \Rightarrow t$, and a set of rewrite rules R is a *rewrite system*. The rewrite relation with R on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, denoted $\to_R$, is the smallest monotonic relation such that $l\sigma \to_R r\sigma$ for all $l \to r \in R$ and all σ. If $s \to_R t$ then we say that *s rewrites into t* with R. We say that R is terminating, confluent, etc. if $\to_R$ is. A rewrite system R is called *convergent* if it is confluent and terminating. It is not difficult to see that then every term t has a unique normal form w.r.t. $\to_R$, denoted by $nf_R(t)$, and $s = t$ is a logical consequence of R (where R is seen as a set of equations) if and only if $nf_R(s) = nf_R(t)$. Sometimes the congruence relations (on $\mathcal{T}(\mathcal{F})$) $\leftrightarrow^*_R$ (or $\leftrightarrow^*_E$) are denoted by R^* (E^*) or $=_R$ ($=_E$).

2.2. Term orderings

A (strict partial) ordering $\succ$ is a transitive and irreflexive binary relation. An ordering $\succ$ on terms is *stable* (or *closed*) under substitutions if $s \succ t$ implies $s\sigma \succ t\sigma$ for all s, t and σ; it fulfills the *subterm property* if $u[s]_p \succ s$ for all s, u and $p \neq \lambda$. It is *total* on $\mathcal{T}(\mathcal{F})$ if for all s and t in $\mathcal{T}(\mathcal{F})$, either $s = t$ or $s \succ t$ or $t \succ s$; if $=$ is a congruence different from syntactic equality, we speak about totality *up to* $=$.

A *rewrite ordering* is a monotonic ordering stable under substitutions; a *reduction ordering* is a well-founded rewrite ordering, and a *simplification ordering* is a rewrite ordering with the subterm property.

The following properties are not difficult to check: a reduction ordering total on $\mathcal{T}(\mathcal{F})$ is necessarily a simplification ordering on $\mathcal{T}(\mathcal{F})$; by Kruskal's theorem, simplification orderings are well-founded (for finite, fixed-arity signatures); and a rewrite system R is terminating if and only if all its rules are contained in a reduction ordering $\succ$, i.e., $l \succ r$ for every $l \to r \in R$ (in fact, then $\to^+_R$ is itself a reduction ordering).

Let $\succ$ be an ordering on terms and let $=$ be a congruence relation. Then $\succ$ is called *compatible* with $=$ if $s' = s \succ t = t'$ implies $s' \succ t'$ for all s,s',t and t'. If E is a set of equations, then $\succ$ is called *E-compatible* if it is compatible with $=_E$. Note that if $\succ$ is E-compatible, $s =_E t$ implies $s \not\succ t$ and $t \not\succ s$.

Let $\succ$ be an ordering on terms and let $=$ be a congruence relation such that $\succ$ is compatible with $=$. Then these relations induce relations on tuples and multisets of terms as follows.

The *lexicographic (left to right) extension of* $\succ$ *with respect to* $=$ is the relation $\succ^{lex}$ on n-tuples of terms defined by:

$$\langle s_1, \ldots, s_n \rangle \succ^{lex} \langle t_1, \ldots, t_n \rangle \quad \text{if} \quad s_1 = t_1, \ldots, s_{k-1} = t_{k-1} \text{ and } s_k \succ t_k$$

for some k in $1 \ldots n$. It is well-known that, if $\succ$ is well founded, so is $\succ^{lex}$.

The *multiset extension of* $=$ is defined as the smallest relation $==$ on multisets of terms such that $\emptyset == \emptyset$ and

$$S \cup \{s\} == S' \cup \{t\} \text{ if } s = t \wedge S == S'$$

The *multiset extension of* $\succ$ *with respect to* $=$ is defined as the smallest ordering $\succ\!\!\succ$ (or $\succ_{mul}$) on multisets of terms such that

$$M \cup \{s\} \succ\!\!\succ N \cup \{t_1, \ldots, t_n\} \text{ if } M == N \text{ and } s \succ t_i \text{ for all } i \in 1 \ldots n$$

Sometimes the notation $\succ\!\!\succ$ is used without explicitly indicating which is the congruence $=$. In these cases $=$ is assumed to be the syntactic equality relation $\equiv$ on terms. If $\succ$ is well founded on S, so is $\succ\!\!\succ$ on finite multisets over S [Dershowitz and Manna 1979].

A way to define suitable orderings for practical purposes (like termination proving or automated deduction) is to construct them directly from a well-founded *precedence*, an ordering $\succ_{\mathcal{F}}$ on $\mathcal{F}$. This is done in the so-called *path orderings*, like the *lexicographic path ordering* (LPO) or the *recursive path ordering (with status)* (RPO) [Kamin and Levy 1980, Dershowitz 1982].

Let $\succ_{\mathcal{F}}$ be a precedence and let $\mathcal{F}$ be the disjoint union of two sets lex and mul, the symbols with lexicographic and multiset *status*, respectively. By $=_{mul}$ we denote the equality of ground terms up to the permutation of direct arguments of symbols with multiset status: $f(s_1, \ldots, s_m) =_{mul} g(t_1, \ldots, t_n)$ if $f = g$ and hence $m = n$, and $s_{\pi(i)} =_{mul} t_i$ for $1 \leq i \leq n$ and where π is a permutation of $1 \ldots n$ which is the identity if $f \in lex$.

In this setting, RPO is defined as follows: $s \succ_{rpo} x$ if x is a variable that is a proper subterm of s or else $s \equiv f(s_1 \ldots s_n) \succ_{rpo} t \equiv g(t_1 \ldots t_m)$ if at least one of the following conditions holds:

- $s_i \succ_{rpo} t$ or $s_i =_{mul} t$, for some $i \in \{1 \ldots n\}$
- $f \succ_{\mathcal{F}} g$, and $s \succ_{rpo} t_j$, for all j in $\{1 \ldots m\}$
- $f \equiv g$ (and hence n=m) and $f \in mul$ and $\{s_1, \ldots, s_n\} \succ\!\!\succ_{rpo} \{t_1, \ldots, t_n\}$
- $f \equiv g$ (and hence n=m) and $f \in lex$, $\langle s_1, \ldots, s_n \rangle \succ^{lex}_{rpo} \langle t_1, \ldots, t_n \rangle$, and $s \succ_{rpo} t_j$, for all j in $\{1 \ldots n\}$

where $\succ_{rpo}^{lex}$ and $\ggcurly_{rpo}$ are, respectively, the lexicographic and multiset extensions of $\succ_{rpo}$ with respect to $=_{mul}$.

The *lexicographic path ordering* (LPO) is defined as the particular case of an RPO where $\mathcal{F} = lex$, i.e., where all symbols have a lexicographic status.

It is known that RPO is a reduction ordering on $\mathcal{T}(\mathcal{F}, \mathcal{X})$, which is moreover total on $\mathcal{T}(\mathcal{F})$ up to $=_{mul}$ (and hence in case of LPO, total up to $\equiv$) if $\succ_{\mathcal{F}}$ is total on $\mathcal{F}$ [Kamin and Levy 1980, Dershowitz 1982].

LPO's are useful for extending reduction orderings $\succ$ that are total up to a congruence $=$ (like RPO is total up to $=_{mul}$), to reduction orderings total up to $\equiv$. This extension is obtained by a lexicographic combination $\succ_t$ whose first component is $\succ$, and whose second component is a total LPO $\succ_{lpo}$, that is, $s \succ_t t$ if either $s \succ t$ or $s = t$ and $s \succ_{lpo} t$.

It is not difficult to see that RPO is C-compatible (C for commutativity) if commutative symbols have multiset status, but it is not AC-compatible.

2.3. Equality clauses and Herbrand interpretations

A clause is a pair of finite multisets of equations Γ (the *antecedent*) and Δ (the *succedent*), denoted by $\Gamma \rightarrow \Delta$. We sometimes use a comma in clauses to denote the union of multisets or the inclusion of equations in multisets; for example, we write $s \simeq t, \Gamma, \Gamma' \rightarrow \Delta$ instead of $\{s \simeq t\} \cup \Gamma \cup \Gamma' \rightarrow \Delta$. Clauses $e_1, \ldots, e_n \rightarrow e'_1, \ldots, e'_m$ are sometimes (equivalently) written as a disjunction of equations and negated equations $\neg e_1 \vee \ldots \vee \neg e_n \vee e'_1 \vee \ldots \vee e'_m$. Hence, the e_i are called the *negative* equations, and the e'_j the *positive* equations, respectively, of the clause.

A clause $\Gamma \rightarrow \Delta$ is called a *Horn clause* if Δ contains at most one equation. The *empty clause* $\square$ is a clause $\Gamma \rightarrow \Delta$ where both Γ and Δ are empty. A *positive* (resp. *negative*) clause is a clause $\Gamma \rightarrow \Delta$ where Γ (resp. Δ) is empty, and a *unit* clause is a clause with exactly one literal.

We will use all aforementioned notions and notations defined for terms t, like $t|_p$, $t[s]_p$, $vars(t)$, $t\sigma$, etc., as well for equations and clauses in the expected way. For example, a term u *occurs* in a clause $\Gamma \rightarrow \Delta$ if $t \simeq s \in \Gamma \cup \Delta$ and $t|_p \simeq u$ for some position p.

Let R be a set of ground equations (or rewrite rules). Then the congruence $\leftrightarrow_R^*$ defines an equality Herbrand *interpretation* I: the domain of I is $\mathcal{T}(\mathcal{F})$, each n-ary function symbol f of $\mathcal{F}$ is interpreted as the function f_I where $f_I(t_1, \ldots, t_n)$ is the term $f(t_1, \ldots, t_n)$, and where the only predicate $\simeq$ is interpreted by $s \simeq t$ if $s \leftrightarrow_R^* t$. The interpretation I defined by R in this way will be denoted by R^*. We write $s = t \in I$ if $s \leftrightarrow_R^* t$. I *satisfies* (is a model of) a ground clause $\Gamma \rightarrow \Delta$, denoted $I \models \Gamma \rightarrow \Delta$, if $I \not\supseteq \Gamma$ or $I \cap \Delta \neq \emptyset$. The empty clause $\square$ is hence satisfied by no interpretation. I satisfies a non-ground clause C if I satisfies all ground instances of C. I satisfies a set of clauses S, denoted by $I \models S$, if it satisfies every clause in S. A clause C is a logical consequence of (or C *follows from*) a set of clauses S, denoted by $S \models C$, if C is satisfied by every model of S.

2.4. Constraints and constrained clauses

An *(ordering and equality) constraint* is a quantifier-free first-order formula built over the binary predicate symbols $>$ and $=$ relating terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Regarding semantics, the constraints are interpreted in $\mathcal{T}(\mathcal{F})$, and $=$ is interpreted as some congruence $=_c$ on $\mathcal{T}(\mathcal{F})$ (like syntactic equality or AC-equality) and $>$ is interpreted as a given reduction ordering $\succ$ on ground terms that is total up to $=_c$. Hence a *solution* of a constraint T is a ground substitution σ with domain $vars(T)$ and such that $T\sigma$ evaluates to true for the given $=_c$ and $\succ$. If a solution for T exists, then T is called *satisfiable*. If every ground substitution with domain $vars(T)$ is a solution of T then T is a *tautology*.

A *constrained clause* is a pair $C \mid T$ where C is a clause and T is a constraint. A *ground instance* of $C \mid T$ is a ground clause $C\sigma$ where σ is a solution of T. The semantics of $C \mid T$ is the set of all its ground instances. Hence, by definition, an interpretation I satisfies $C \mid T$ if $I \models C\sigma$ for every ground instance $C\sigma$ of $C \mid T$. Therefore, clauses with unsatisfiable constraints are tautologies. A clause $C \mid T$ is the *constrained empty clause*, denoted as well by $\square$, if C is empty and T is satisfiable. Constrained clauses $C \mid T$ where T is a tautology are sometimes denoted by C, omitting the constraint part T.

3. Paramodulation calculi

A logical *inference* is a step by which from a multiset of zero or more constrained clauses (the *premises*) a new constrained clause (the *conclusion*) is obtained. An *inference rule* $\mathcal{R}$

$$\frac{C_1 \mid T_1 \ \ldots \ C_n \mid T_n}{D \mid T} \quad \text{if} \quad condition$$

is (a finite representation of) the set of inferences where from the multiset of clauses of the form $\{C_1 \mid T_1 \ \ldots \ C_n \mid T_n\}$ one can infer $D \mid T$ if *condition* holds. One such an inference is called an *inference by* $\mathcal{R}$. An *inference system* $\mathcal{I}$ is a set of inference rules. An *inference by* $\mathcal{I}$ is an inference by one of the rules of $\mathcal{I}$. We will frequently consider inference rules where premises or conclusions have constraints that are tautologies and hence these constraints are omitted.

An inference rule $\mathcal{R}$ is *correct* if, for all inferences by $\mathcal{R}$, the conclusion is a logical consequence of the premises, and an inference system is correct if all its rules are correct. A set of constrained clauses S is *closed* under $\mathcal{I}$ if for every inference by $\mathcal{I}$ with premises in S, the corresponding conclusion is in S. $\mathcal{I}$ is *refutation complete* if $\square \in S$ for every unsatisfiable set of constrained clauses S closed under $\mathcal{I}$. All inference systems in the remainder of this chapter are easily proved correct, and we will focus on completeness.

3.1. The model generation method

We start with a simple example on ground Horn clauses in order to introduce the *model generation* method, the standard technique for establishing the completeness of ordered paramodulation calculi that will be used throughout this chapter. Note that if $C \mid T$ is a constrained clause where C is ground and T is satisfiable, then it is equivalent to $C \mid \top$ where $\top$ denotes a tautological constraint. Hence in the remainder of this section constraints will be omitted.

In the following, let $\succ$ be a given total reduction ordering on $\mathcal{T}(\mathcal{F})$, and let $s \succeq t$ denote $s \succ t \vee s \equiv t$. The inference system $\mathcal{G}$ for ground Horn clauses with equality is the following:

superposition right:

$$\frac{\Gamma' \to l \simeq r \qquad \Gamma \to s \simeq t}{\Gamma', \Gamma \to s[r]_p \simeq t} \quad \text{if} \quad \begin{array}{l} s|_p \equiv l,\ l \succ r,\ s \succ t, \text{ and} \\ l \succ u \text{ for all } u \text{ occurring in } \Gamma', \text{ and} \\ s \succ v \text{ for all } v \text{ occurring in } \Gamma \end{array}$$

superposition left:

$$\frac{\Gamma' \to l \simeq r \qquad \Gamma, s \simeq t \to \Delta}{\Gamma', \Gamma, s[r]_p \simeq t \to \Delta} \quad \text{if} \quad \begin{array}{l} s|_p \equiv l,\ l \succ r,\ s \succ t, \text{ and} \\ l \succ u \text{ for all } u \text{ occurring in } \Gamma', \text{ and} \\ s \succeq v \text{ for all } v \text{ occurring in } \Gamma, \Delta \end{array}$$

equality resolution:

$$\frac{\Gamma, s \simeq s \to \Delta}{\Gamma \to \Delta} \quad \text{if} \quad s \succeq v \text{ for all } v \text{ occurring in } \Gamma, \Delta$$

Let us remark that the equality resolution rule is named after the fact that it encodes a resolution inference with the reflexivity axiom of equality $x \simeq x$,

It is sometimes said that in the superposition rules the inferences take place *with* the term l *on* the term s, and that the inference *involves* s and l. Note that in $\mathcal{G}$, superposition right inferences involve only terms s and l that are *strictly maximal* in their respective premises, that is, they are bigger w.r.t. $\succ$ than all other occurrences of terms in these premises. Superposition left takes place also with strictly maximal terms, but on (possibly non-strictly) *maximal* terms (that is, they are larger than or equal to all terms in their premise).

In order to prove the refutation completeness of $\mathcal{G}$ we first define the following total ordering $\succ_c$ on ground clauses. If C is a clause

$$s_1 = s'_1, \ldots, s_n = s'_n \to t_1 = t'_1, \ldots, t_m = t'_m$$

then we define $ms(C)$ as the multiset:

$$\{\{s_1, s_1, s'_1, s'_1\}, \ldots, \{s_n, s_n, s'_n, s'_n\}, \{t_1, t'_1\}, \ldots, \{t_m, t'_m\}\}$$

Finally, let $\succ_c$ be the ordering on clauses defined by comparing these expressions by the two-fold multiset extension of $\succ$, that is, $C \succ_c D$ if $ms(C)(\succ_{mul})_{mul}ms(D)$. The result is a total ordering on ground clauses[5].

Now we come to the key to the model generation method. Our aim is to prove the completeness of $\mathcal{G}$. We do this by showing that, if S is a set of ground Horn clauses closed under $\mathcal{G}$ and $\Box \notin S$, then S is satisfiable. The satisfiability proof of S is of a constructive nature: first, an equality Herbrand interpretation will be built, and second, it will be shown that this interpretation is a model of S.

We now informally explain the first part. The interpretation we build will be the congruence R^* induced by a set of ground rewrite rules R, where each rule in R has been *generated* by some clause of S (hence the name "model generation"). The generation process of R is defined by induction on $\succ_c$. Each clause C in S generates a rule or not, depending on the set R_C of rules generated by clauses D of S with $C \succ_c D$ (and on the congruence R^*_C induced by R_C). These ideas are formalised as follows:

3.1. DEFINITION. (Model generation) Let C be a clause in S. Then $Gen(C) = \{l \Rightarrow r\}$, and C is said to *generate* the rule $l \Rightarrow r$, if, and only if, C is of the form $\Gamma \rightarrow l \simeq r$ and the three following conditions hold:

1. $R^*_C \not\models C$,
2. $l \succ r$ and $l \succ u$ for all u occurring in Γ
3. l is irreducible by R_C

where $R_C = \bigcup_{C \succ_c D} Gen(D)$. In all other cases $Gen(C) = \emptyset$. Finally, R denotes the set of all rules generated by clauses of S, that is, $R = \bigcup_{D \in S} Gen(D)$.

Let us analyse the three conditions. The first one states that a clause only contributes to the model if it does not hold in the partial model built so far and hence we are *forced* to extend this partial model. The second one states that a clause can only generate a rule $l \Rightarrow r$ if l is the strictly maximal term of the clause. The third condition, stating that l is irreducible by the rules generated so far, is, together with the second one, the key for showing that R is convergent, from which the completeness result quite easily follows:

3.2. LEMMA. *For every set of ground clauses S, the set of rules R generated for S is convergent (i.e., confluent and terminating). Furthermore, if $R^*_C \models C$ then $R^* \models C$ for all ground C.*

Proof. Evidently, R is terminating since $l \succ r$ for all its rules $l \Rightarrow r$. To prove confluence, it suffices to show *local* confluence, which in the ground case is well-known (and easily shown) to hold if there are no two different rules $l \Rightarrow r$ and $l' \Rightarrow r'$ where l' is a subterm of l. This property is fulfilled: clearly when a clause

[5]Roughly, $\succ_c$ compares the multisets of all equations occurring in the clauses, but where in addition terms occurring negatively have slightly more weight than the ones occurring positively; in fact, in order to make $\succ_c$ total on ground clauses, the information of which equations are positive and which ones are negative has to be present anyway.

C in S generates $l \Rightarrow r$, no such $l' \Rightarrow r'$ is in R_C; but if $l' \Rightarrow r'$ is generated by a clause D with $D \succ_c C$ then, by definition of $\succ_c$, we must have $l' \succ l$ and hence l' cannot be a subterm of l either.

To show $R_C^* \models C$ implies $R^* \models C$, let C be $\Gamma \rightarrow \Delta$, and assume $R_C^* \models C$. If $R_C^* \models \Delta$ then $R^* \models \Delta$ since $R \supseteq R_C$. Otherwise, $R_C^* \not\models \Gamma$. Then $R^* \not\models \Gamma$ follows from the fact a term t occurring negatively in a clause is bigger than the same t occurring positively: all rules in $R \setminus R_C$ are generated by clauses bigger than C, and hence have left hand sides that are too big to reduce any term occurring in Γ. Since R is convergent this implies $R^* \not\models \Gamma$. □

3.3. THEOREM. *The inference system $\mathcal{G}$ is refutation complete for ground Horn clauses.*

Proof. Let S be a set of ground Horn clauses that is closed under $\mathcal{G}$ and such that $\square \notin S$. We prove that then S is satisfiable by showing that R^* is a model for S. We proceed by induction on $\succ_c$, that is, we derive a contradiction from the existence of a minimal (w.r.t. $\succ_c$) clause C in S such that $R^* \not\models C$. There are a number of cases to be considered, depending on the occurrences in C of its maximal term s, i.e., the term s such that $s \succeq u$ for all terms u in C (s is unique since $\succ$ is total on $\mathcal{T}(\mathcal{F})$):

1. s occurs only in the succedent and C is $\Gamma \rightarrow s \simeq s$. This is not possible since $R^* \not\models C$.
2. s occurs only in the succedent and C is $\Gamma \rightarrow s \simeq t$ with $s \not\equiv t$. Since $R^* \not\models C$, we have $R^* \supseteq \Gamma$ and $s \simeq t \notin R^*$, i.e., C has not generated the rule $s \Rightarrow t$. This must be because s is reducible by some rule $l \Rightarrow r \in R_C$. Assume $l \Rightarrow r$ has been generated by a clause C' of the form $\Gamma' \rightarrow l \simeq r$. Then there exists an inference by superposition right:
$$\frac{\Gamma' \rightarrow l \simeq r \qquad \Gamma \rightarrow s \simeq t}{\Gamma', \Gamma \rightarrow s[r]_p \simeq t}$$
whose conclusion D has only terms u with $s \succ u$, and hence $C \succ_c D$. Moreover, D is in S and $R^* \not\models D$, since $R^* \supseteq \Gamma \cup \Gamma'$ and $s[r]_p \simeq t \notin R^*$ (since otherwise $s[l]_p \simeq t \in R^*$). This contradicts the minimality of C.
3. s occurs in the antecedent and C is $\Gamma, s \simeq s \rightarrow \Delta$. Then there exists an inference by equality resolution:
$$\frac{\Gamma, s \simeq s \rightarrow \Delta}{\Gamma \rightarrow \Delta}$$
for whose conclusion D it holds that $C \succ_c D$. Moreover, D is in S and $R^* \not\models D$, which is a contradiction as in the previous case.
4. s occurs in the antecedent and C is $\Gamma, s \simeq t \rightarrow \Delta$ with $s \succ t$. Since $R^* \not\models C$, we have $s \simeq t \in R^*$ and since R is convergent, s and t must have the same normal forms w.r.t. R, so s must be reducible by some rule $l \Rightarrow r \in R$. Assume $l \Rightarrow r$

has been generated by a clause C' of the form $\Gamma' \to l \simeq r$. Then there exists an inference by superposition left:

$$\frac{\Gamma' \to l \simeq r \qquad \Gamma, s[l]_p \simeq t \to \Delta}{\Gamma', \Gamma, s[r]_p \simeq t \to \Delta}$$

for whose conclusion D it holds that $C \succ_c D$. Moreover, D is in S and $R^* \not\models D$, which again contradicts the minimality of C. $\square$

The following example shows how the rewrite system R changes during a closure of a set of ground clauses and that, although for the intermediate sets the obtained R^* is not a model, the R^* obtained for the closed set is a model.

3.4. EXAMPLE. Consider the lexicographic path ordering generated by the precedence $f \succ_{\mathcal{F}} a \succ_{\mathcal{F}} b \succ_{\mathcal{F}} c \succ_{\mathcal{F}} d$. The following table shows in the left column the ground Horn clauses (sorted with respect to the ordering) at each closure step, in which the first one is the initial set, and in the right column the set R corresponding to each intermediate set. The maximal term of every clause is underlined and the subterms of the clauses involved in the inference are framed.

S	R
$\to \boxed{\underline{c}} \simeq d$ $\underline{f(d)} \simeq d \to a \simeq b$ $\to \underline{f(\boxed{c})} \simeq d$	$c \Rightarrow d$
$\to \underline{c} \simeq d$ $\to \boxed{\underline{f(d)}} \simeq d$ $\boxed{\underline{f(d)}} \simeq d \to a \simeq b$ $\to \underline{f(c)} \simeq d$	$c \Rightarrow d$ $f(d) \Rightarrow d$
$\to \underline{c} \simeq d$ $d \simeq d \to \underline{a} \simeq b$ $\to \underline{f(d)} \simeq d$ $\underline{f(d)} \simeq d \to a \simeq b$ $\to \underline{f(c)} \simeq d$	$c \Rightarrow d$ $a \Rightarrow b$ $f(d) \Rightarrow d$

Let us conclude this section with a remark on additional ordering restrictions. In superposition left as well as in equality resolution, it is possible to strengthen the conditions in such a way that only one negative literal becomes eligible for inferences. For example, in superposition left on an equation $s \simeq t$, one can require that $t \succ t'$ for all equations $s \simeq t'$ in Γ, that is, we use the maximal equation rather than just the maximal term; if two equations have the same maximal terms, we compare the other terms. Similarly, in equality resolution we can require $s \succ t'$ for all equations $s \simeq t'$ in Γ. In the inference system for general clauses (see Subsection 3.5) we have included these restrictions, since such comparisons between equations are needed there anyway. We did not consider them for $\mathcal{G}$ for simplicity reasons, and also because by means of *selection of negative equations* we will be able to obtain stronger results in a simpler way (see Subsection 3.6).

3.2. Non-equality predicates

In this framework, equality can be considered to be the only predicate, since for every other predicate symbol p, (positive or negative) atoms $p(t_1 \dots t_n)$ can be expressed as (positive or negative) equations $p(t_1 \dots t_n) \simeq \mathit{true}$, where true is a new special symbol, and where p is considered as a function symbol rather than as a predicate symbol. Note however that, in order to avoid meaningless expressions in which predicate symbols occur at proper subterms one should adopt a two-sorted type discipline on terms in the encoding.

It is easy to see that this transformation preserves satisfiability. Very roughly: one can "translate" the interpretations such that a ground atom is true in a Herbrand interpretation I if and only if in the equality Herbrand interpretation I' over the modified signature the term $p(t_1 \dots t_n)$ is congruent to true. Be we remark that I and I' are not isomorphic since two ground atoms that are false in I need not be in the same congruence class of I'.

After this satisfiability preserving transformation, ordered resolution (ground) inferences of the form:

$$\frac{\Gamma' \rightarrow A \qquad \Gamma, A \rightarrow \Delta}{\Gamma', \Gamma \rightarrow \Delta} \qquad \text{if} \quad A \succ \Gamma' \text{ and } A \succeq \Gamma, \Delta.$$

become a special case of superposition left:

$$\frac{\Gamma' \rightarrow A \simeq \mathit{true} \qquad \Gamma, A \simeq \mathit{true} \rightarrow \Delta}{\Gamma', \Gamma, \mathit{true} \simeq \mathit{true} \rightarrow \Delta}$$

combined with equality resolution (or simplification, as we will see) for eventually eliminating the trivial equation $\mathit{true} \simeq \mathit{true}$.

For efficiency reasons it is convenient to make true small in the ordering. Sometimes it is also useful to take into account that p is a predicate symbol when handling the ordering restrictions. For example, in orderings like RPO, if the predicate symbols p are bigger in the precedence than function symbols then $p \succ_{\mathcal{F}} q$ implies $p(t_1, \dots, t_n)\sigma \succ_{lpo} q(s_1, \dots, s_m)\sigma$ for all ground σ.

3.3. Clauses with variables

Up to now, in this section we have only dealt with ground clauses. If we consider that a non-ground clause represents the set of all its ground instances[6], a refutation complete method for the non-ground case would be to systematically enumerate all ground instances of the clauses, and to perform inferences by $\mathcal{G}$ between those instances. But fortunately it is possible to perform inferences between non-ground clauses, covering in one step a possibly large number of ground inferences. We now adapt $\mathcal{G}$ according to this view.

For example, at the ground level, in the superposition right inference

$$\frac{\Gamma' \rightarrow l \simeq r \qquad \Gamma \rightarrow s \simeq t}{\Gamma', \Gamma \rightarrow s[r]_p \simeq t} \quad \text{if} \quad \begin{array}{l} s|_p \equiv l,\ l \succ r,\ s \succ t, \text{ and} \\ l \succ u \text{ for all } u \text{ occurring in } \Gamma', \text{ and} \\ s \succ v \text{ for all } v \text{ occurring in } \Gamma \end{array}$$

we required $s|_p$ and l to be the same term. At the non-ground level, this becomes a constraint $s|_p = l$ on the possible instances of the conclusion, that is, the conclusion is a constrained clause $D \mid T$. Hence if the conclusion is $D \ \mid s|_p = l \wedge \ldots$, the instances $D\sigma$ for which $s|_p\sigma \not\equiv l\sigma$ are not created. The same is done for the ordering restrictions. For instance, instead of requiring $l \succ r$ as a condition of the inference, it becomes part of the constraint of the conclusion, excluding those instances $D\sigma$ of the conclusion that correspond to ground inferences between instances of the premises for which $l\sigma \succ r\sigma$ does not hold:

$$\frac{\Gamma' \rightarrow l \simeq r \qquad\qquad \Gamma \rightarrow s \simeq t}{\Gamma', \Gamma \rightarrow s[r]_p \simeq t \ \mid s|_p{=}l \ \wedge \ l{>}r \ \wedge \ s{>}t \ \wedge \ldots}$$

Note that here we have written the inference rule without constraints in its premises, since at this point we are only interested in the constraints that are generated in this concrete inference. In Section 5 paramodulation with constraints inherited from the premises with be considered in detail.

This inference rule can be further restricted with the additional condition stating that the inference is not necessary if $s|_p$ is a variable. This shows that, by working on the non-ground level, certain inferences between ground instances of the premises turn out to be redundant: at the non-ground level we do not perform, for an instance with σ, the inferences *inside* σ (also called inferences *below variables*), that is, on positions $s\sigma|_p$ where $s|_{p'}$ is a variable for some prefix p' of p.

Note that, as usual, it may be necessary to rename variables in the premises in order to avoid name clashes: the premises C and D are assumed to fulfill $vars(C) \cap vars(D) = \emptyset$.

Now we define the inference system $\mathcal{H}$ for non-ground Horn clauses, writing $s > \Gamma$ as a shorthand for the constraint $\ s{>}u_1 \ \wedge \ s{>}v_1 \ \wedge \ldots \wedge \ s{>}u_n \ \wedge \ s{>}v_n$ if

[6] By Herbrand's theorem, considering only the ground instances preserves satisfiability; in fact, this is a consequence of (the proof of) Theorem 3.10.

Γ is a multiset of equations $\{u_1 \simeq v_1, \ldots, u_n \simeq v_n\}$ (and similarly, we write $s \geq \Gamma$ for $s \geq u_1 \ \wedge \ s \geq v_1 \ \wedge \ldots \wedge \ s \geq u_n \ \wedge \ s \geq v_n$):

superposition right:

$$\frac{\Gamma' \rightarrow l \simeq r \qquad \Gamma \rightarrow s \simeq t}{\Gamma', \Gamma \rightarrow s[r]_p \simeq t \ \mid s|_p = l \ \wedge \ l > r \ \wedge \ l > \Gamma' \ \wedge \ s > t \ \wedge \ s > \Gamma}$$

superposition left:

$$\frac{\Gamma' \rightarrow l \simeq r \qquad \Gamma, s \simeq t \rightarrow \Delta}{\Gamma', \Gamma, s[r]_p \simeq t \rightarrow \Delta \ \mid s|_p = l \ \wedge \ l > r \ \wedge \ l > \Gamma' \ \wedge \ s > t \ \wedge \ s \geq \Gamma, \Delta}$$

equality resolution:

$$\frac{\Gamma, s \simeq t \rightarrow \Delta}{\Gamma \rightarrow \Delta \ \mid s = t \ \wedge \ s \geq \Gamma, \Delta}$$

where in both superposition rules $s|_p$ is required not to be a variable.

3.5. EXAMPLE. Consider the lexicographic path ordering generated by the precedence $h \succ_{\mathcal{F}} a \succ_{\mathcal{F}} f \succ_{\mathcal{F}} g \succ_{\mathcal{F}} b$. In the following inference

$$\frac{g(x) \simeq x \rightarrow f(a,x) \simeq f(x,x) \qquad\qquad \rightarrow h(f(a,g(y)) \simeq h(y)}{\begin{array}{l} g(x) \simeq x \rightarrow h(f(x,x)) \simeq h(y) \mid f(a,x) = f(a,g(y)) \ \wedge \ h(f(a,g(y))) > h(y) \ \wedge \\ \qquad\qquad\qquad f(a,x) > f(x,x) \ \wedge \ f(a,x) > g(x) \ \wedge \ f(a,x) > x \end{array}}$$

the constraint of the conclusion is satisfiable: using the properties of the ordering and solving the unification problem, the constraint can be simplified into

$$x = g(y) \ \wedge \ a > x$$

which has, for instance, the solution $\{y \mapsto b, x \mapsto g(b)\}$.

On the other hand, the following inference is not needed

$$\frac{\rightarrow f(x,x) \simeq f(a,x) \qquad\qquad \rightarrow f(g(y),z) \simeq h(z)}{\begin{array}{l} \rightarrow f(a,x) \simeq h(z) \ \mid f(x,x) = f(g(y),z) \ \wedge \ f(g(y),z) > h(z) \ \wedge \\ \qquad\qquad f(x,x) > f(a,x) \end{array}}$$

since the constraint of the conclusion has no solution; it can be simplified to

$$x = g(y) \ \wedge \ x = z \ \wedge \ y \geq h(z) \ \wedge \ x > a$$

which implies $y \geq h(g(y))$. Note that the equality constraint and the ordering constraint considered separately are both satisfiable but their conjunction is not. □

Let us also remark that, at the non-ground level, several terms in a premise C may be involved in paramodulation inferences; for a term t it may be the case that for some ground instances $C\sigma$ the term $t\sigma$ is the maximal one, and for other instances it is not.

3.4. Completeness without constraint inheritance

There are several possible treatments for the constrained clauses generated by the inference system $\mathcal{H}$. The classical view is to deal only with unconstrained clauses. Conclusions of the form $C \mid s = t \wedge OC$, for some ordering constraint OC, are then immediately converted into $C\sigma$ where $\sigma = mgu(s,t)$. This strategy will be called here $\mathcal{H}$ *without constraint inheritance*, in contrast with other possibilities which will be introduced later on.

Of course, the clause $C\sigma$ has to be generated only if the constraint $s = t \wedge OC$ is satisfiable in $\mathcal{T}(\mathcal{F})$, where $=$ is interpreted as the syntactic equality relation $\equiv$, and $>$ as the given reduction ordering $\succ$. If $\succ$ is the lexicographic path ordering (LPO) the satisfiability of such constraints is decidable [Comon 1990, Nieuwenhuis 1993] (see Section 7 of this chapter). But traditionally in the literature weaker approximations by non-global tests are used; for example, inference systems are sometimes expressed with local conditions of the form $r \not\succeq l$ when in our framework we have $l > r$ as a part of the global constraint OC. Note that such weaker approximations do not lead to unsoundness, but only to the generation of unnecessary (for completeness) clauses.

In the following, we call a set of (unconstrained) Horn clauses S *closed under* $\mathcal{H}$ *without constraint inheritance* if $D\sigma \in S$ for all inferences by $\mathcal{H}$ with premises in S and conclusion $D \mid s = t \wedge OC$ such that $s = t \wedge OC$ is satisfiable and $\sigma = mgu(s,t)$.

3.6. THEOREM. *The inference system* $\mathcal{H}$ *is refutation complete without constraint inheritance for Horn clauses.*

Proof. Let S be a set of Horn clauses closed under $\mathcal{H}$ without constraint inheritance such that $\square \notin S$. The proof is very similar to the one for $\mathcal{G}$: we exhibit a model R^* for S. We proceed again by induction on $\succ_c$, but now the role of the ground clauses in the proof for $\mathcal{G}$ is played by all *ground instances* of clauses in S, and the generation of rules in R from these ground instances is the same as for $\mathcal{G}$. Now we derive a contradiction from the existence of a minimal (w.r.t. $\succ_c$) *ground instance* $C\sigma$ *of a* clause C in S such that $R^* \not\models C\sigma$. The cases considered are the same ones as well, again depending on the occurrences in $C\sigma$ of its maximal term $s\sigma$.

The only difference lies in the *lifting* argument, which is the same in all cases and is hence analyzed here for only one of them: C is $\Gamma, s \simeq t \rightarrow \Delta$ and $s\sigma \succ t\sigma$. Since $R^* \not\models C\sigma$, we have $s\sigma \simeq t\sigma \in R^*$ and since R is convergent, $s\sigma$ must be reducible by some rule $l\sigma \Rightarrow r\sigma \in R$, generated by a clause C' of the form $\Gamma' \rightarrow l \simeq r$. (Note that, since we assume that there are no name clashes between the variables of C and C', we can consider that the instances of C and of C' under consideration are both by the same ground σ.) Now we have $s\sigma|_p = l\sigma$, and there are two possibilities:

An inference. $s|_p$ is a non-variable position of s.

Then there exists an inference by superposition left:

$$\frac{\Gamma' \to l \simeq r \qquad \Gamma, s \simeq t \to \Delta}{\Gamma', \Gamma, s[r]_p \simeq t \to \Delta \ \ | \, s|_p = l \ \wedge \ l > r \ \wedge \ l > \Gamma' \ \wedge \ s > t \ \wedge \ s \geq \Gamma, \Delta}$$

whose conclusion $D \mid T$ has an instance $D\sigma$ (i.e., σ is a solution of T) such that $C\sigma \succ_c D\sigma$, where $R^* \not\models D\sigma$, contradicting the minimality of $C\sigma$.

Lifting. $s|_{p'}$ is a variable x for some prefix p' of p.
Then $p = p' \cdot p''$ for some p'', and $x\sigma|_{p''}$ is $l\sigma$. Now let σ' be the ground substitution with the same domain as σ but where $x\sigma' = x\sigma[r\sigma]_{p''}$ and $y\sigma' = y\sigma$ for all other variables y. Then $R^* \not\models C\sigma'$ and $C\sigma \succ_c C\sigma'$, contradicting the minimality of $C\sigma$.
□

3.5. General clauses

In this section general clauses are considered, i.e., clauses that may have several equations in their succedents. For this purpose, the inference system $\mathcal{H}$ is adapted. In order to restrict the amount of inferences to be performed, it is desirable to preserve the property of $\mathcal{H}$ that for each ground clause (or instance) C, only one literal of C is involved in superposition inferences with C. Since now the maximal term of C may occur in more than one equation in the succedent, it is decided that among these equations the one whose other side is maximal will be used. This leads to the notion of maximal and strictly maximal equations in C. In order to express maximality and strict maximality of equations as constraints, we use the following notation. The constraint $gr(s \simeq t, \Delta)$ expresses that the equation $s \simeq t$, i.e., the multiset $\{s, t\}$, is strictly greater, w.r.t. the multiset extension of $\succ$, than all equations $u \simeq v$ in Δ. For each $u \simeq v$ this condition $s \simeq t \succ\!\!\succ u \simeq v$ can be expressed for instance by the constraint:

$s > u \ \wedge \ (s \geq v \ \vee \ t \geq v) \ \ \vee \ \ s > v \ \wedge \ (s \geq u \ \vee \ t \geq u) \ \vee$
$t > u \ \wedge \ (s \geq v \ \vee \ t \geq v) \ \ \vee \ \ t > v \ \wedge \ (s \geq u \ \vee \ t \geq u)$

Similarly, the constraint $greq(s \simeq t, \Delta)$ expresses that $s \simeq t \succeq\!\!\succeq u \simeq v$ for all $u \simeq v$ in Δ. The full inference system $\mathcal{I}$ for general clauses is *superposition right:*

$$\frac{\Gamma' \to l \simeq r, \Delta' \qquad\qquad \Gamma \to s \simeq t, \Delta}{\Gamma', \Gamma \to s[r]_p \simeq t, \Delta', \Delta} \;\Big|\; \begin{array}{l} s|_p = l \;\wedge \\ l > r \;\wedge\; l > \Gamma' \;\wedge\; gr(l \simeq r, \Delta') \;\wedge \\ s > t \;\wedge\; s > \Gamma \;\wedge\; gr(s \simeq t, \Delta) \end{array}$$

superposition left:

$$\frac{\Gamma' \to l \simeq r, \Delta' \qquad\qquad \Gamma, s \simeq t \to \Delta}{\Gamma', \Gamma, s[r]_p \simeq t \to \Delta', \Delta} \;\Big|\; \begin{array}{l} s|_p = l \;\wedge \\ l > r \;\wedge\; l > \Gamma' \;\wedge\; gr(l \simeq r, \Delta') \;\wedge \\ s > t \;\wedge\; greq(s \simeq t, \Gamma \cup \Delta) \end{array}$$

equality resolution:

$$\frac{\Gamma, s \simeq t \to \Delta}{\Gamma \to \Delta \;|\; s = t \;\wedge\; greq(s \simeq t, \Gamma \cup \Delta)}$$

equality factoring:

$$\frac{\Gamma \to s \simeq t, s' \simeq t', \Delta}{\Gamma, t \simeq t' \to s \simeq t', \Delta \;|\; s = s' \;\wedge\; s > t \;\wedge\; s > \Gamma \;\wedge\; greq(s \simeq t, \Delta \cup \{s' \simeq t'\})}$$

where as in the Horn case in both superposition rules $s|_p$ is not a variable.

Here the superposition rules and the equality resolution rule play the same role as their counterparts in the inference system $\mathcal{H}$. The equality factoring rule is new. Intuitively, it expresses that, if s and s' are syntactically equal, and t and t' are semantically equal, then the two equations in the succedent express the same information, and one of them can be omitted.

3.7. EXAMPLE. Consider the lexicographic path ordering generated by the precedence $f \succ_{\mathcal{F}} g \succ_{\mathcal{F}} h$ and the following inference by superposition right

$$\frac{\to g(z) \simeq h(z) \qquad\qquad \to f(g(x), y) \simeq g(x), f(g(x), y) \simeq y}{\to f(h(z), y) \simeq g(x), f(g(x), y) \simeq y} \;\Big|\; \begin{array}{l} g(x) = g(z) \;\wedge\; g(z) > h(z) \;\wedge \\ f(g(x), y) > g(x) \;\wedge \\ gr(f(g(x), y) \simeq g(x), \{f(g(x), y) \simeq y\}) \end{array}$$

where $gr(f(g(x), y) \simeq g(x), \{f(g(x), y) \simeq y\})$ can be simplified into $g(x) > y$. Now, simplifying the rest of the constraint, the conclusion of the inference can be written as

$$\to f(h(z), y) \simeq g(x), f(g(x), y) \simeq y \;\mid\; x = z \;\wedge\; g(x) > y$$

Below an overview of the new aspects for the completeness proof of $\mathcal{I}$ with respect to $\mathcal{H}$ is given. For simplicity, only the ground case is considered; lifting to clauses with variables is analogous to what was done for $\mathcal{H}$. First, a new condition is added in the generation of the rewrite system R for a set of clauses S (see Section 3.1) and the second condition is adapted in order to select the strictly maximal positive equation that produces the rule:

3.8. DEFINITION. Let S be a set of ground clauses and let C be a clause in S. Then $Gen(C) = \{l \Rightarrow r\}$, and C is said to *generate* the rule $l \Rightarrow r$, if, and only if, C is of the form $\Gamma \rightarrow l \simeq r, \Delta$ and

1. $R^*_C \not\models C$
2. $l \succ r$, $l \succ \Gamma$, and $l \simeq r \succ\!\!\succ u \simeq v$ for all $u \simeq v$ in Δ
3. l is irreducible by R_C
4. $R^*_C \not\models r \simeq t'$ for every $l \simeq t' \in \Delta$

where $R_C = \bigcup_{C \succ_c D} Gen(D)$. In all other cases $Gen(C) = \emptyset$. Finally, R denotes the set of all rules generated by clauses of S, that is, $R = \bigcup_{D \in S} Gen(D)$.

The proof of Lemma 3.2 can be easily adapted to show that here again R is convergent and that if $R^*_C \models C$ then $R^* \models C$. In a very similar way, it can be shown that the new conditions force clauses generating rules to have only one positive literal satisfied by the interpretation:

3.9. LEMMA. *If a clause C of the form $\Gamma \rightarrow l \simeq r, \Delta$ generates the rule $l \Rightarrow r$ then $R^* \models \Gamma$ and $R^* \not\models \Delta$.*

3.10. THEOREM. *The inference system $\mathcal{I}$ is refutation complete for general clauses.*

Proof. Since lifting is done as for $\mathcal{H}$, here we only extend the proof for the ground case $\mathcal{G}$. There is one additional case due to the new conditions for generating rules in R. The other cases of the proof for $\mathcal{G}$ are straightforwardly adapted by using lemma 3.9 to show that the conclusion of the required inference is not satisfied by the model.

The new case is: C is of the form $\Gamma \rightarrow s \simeq t, \Delta$, with $s \succ t, \Gamma$ and $s \simeq t$ is maximal in Δ, and it has not generated a rule because there is an equation $s \simeq t'$ in Δ such that $R^*_C \models t \simeq t'$ (note that this case includes also the case in which $s \simeq t$ is maximal in Δ, but not strictly maximal).

Then, with $\Delta = s \simeq t', \Delta'$, there exists an inference by equality factoring

$$\frac{\Gamma \rightarrow s \simeq t, s \simeq t', \Delta}{\Gamma, t \simeq t' \rightarrow s \simeq t', \Delta}$$

whose conclusion D is such that $C \succ_c D$ and $R^* \not\models D$, contradicting the minimality of C. □

3.6. Selection of negative equations

The inference system $\mathcal{I}$ includes strong ordering restrictions: roughly, a superposition inference is needed only if the terms involved are maximal sides of maximal equations in their respective premises, and even strictly maximal in case they occur in positive equations. But more constraints can be imposed. If a clause C has a non-empty antecedent, it is possible to arbitrarily *select* exactly one of its negative equations. Then completeness is preserved even if C is not used as left premise of any superposition inference and the only inferences involving C are equality resolution or superposition left on its selected equation.

The inference system $\mathcal{S}$ (for selection) for general clauses is defined to consist of the four rules of inference system I where for all premises of the inference rules no negative equation has been selected, plus the following two additional rules, where the selected equations have been underlined:

superposition left on a selected equation:

$$\frac{\Gamma' \to l \simeq r, \Delta' \qquad\qquad \Gamma, \underline{s \simeq t} \to \Delta}{\Gamma', \Gamma, s[r]_p \simeq t \to \Delta', \Delta \quad \mid \quad s|_p = l \ \wedge} \\ \qquad\qquad l > r \ \wedge \ l > \Gamma' \ \wedge \ gr(l \simeq r, \Delta') \ \wedge \ s > t$$

equality resolution on a selected equation:

$$\frac{\Gamma, \underline{s \simeq t} \to \Delta}{\Gamma \to \Delta \ \mid \ s = t}$$

where, as usual in superposition rules, $s|_p$ is not a variable.

Note that an adequate selection strategy gives us a strictly more restrictive inference system: among the set of maximal negative equations, just select one of them, and select no equation if this set is empty. It is clear that in the inference system $\mathcal{I}$ *all* maximal equations of the antecedent are eligible for superposition left or equality resolution, whereas in $\mathcal{S}$ only the selected one is eligible.

The intuition behind selection is, roughly, that a clause with negative equations does not need to contribute to the deduction process until its whole antecedent has been proved from other clauses, and in particular one can require the selected equation to be proved first.

In practice one can select for example always a maximal equation (under some arbitrary ordering) of the antecedent. Selecting always a negative equation, whenever there is one, leads in the Horn case to the so-called *positive unit literal strategies*, that is, the left premise of superposition inferences is always a positive unit clause [Dershowitz 1991, Nieuwenhuis and Nivela 1991]. For general clauses eager selection leads to *positive* strategies, where the left premise is always a positive clause, i.e., it has only positive literals. Adapting the proof of completeness of Theorem 3.10 to this framework with selection is an easy exercise: it suffices to consider that clauses with selected equations generate no rules.

3.7. Merging paramodulation and perfect models

There is an alternative to the equality factoring inference rule, which is *merging paramodulation* rule plus *ordered factoring* [Bachmair and Ganzinger 1994*b*]. The inference system $\mathcal{M}$ consists of the paramodulation rules and the equality resolution rule of $\mathcal{I}$, plus the following two rules:

merging paramodulation:

$$\frac{\Gamma' \rightarrow l \simeq r, \Delta' \qquad\qquad \Gamma \rightarrow s \simeq t, s' \simeq t', \Delta}{\Gamma', \Gamma \rightarrow s \simeq t[r]_p, s' \simeq t', \Delta', \Delta} \;\; \begin{array}{l} \mid t|_p = l \;\wedge\; s = s' \;\wedge \\ l > r \;\wedge\; l > \Gamma' \;\wedge\; gr(l \simeq r, \Delta') \;\wedge \\ s > t \;\wedge\; s > \Gamma \;\wedge\; greq(s \simeq t, \Delta \cup \{s' \simeq t'\}) \end{array}$$

ordered factoring:

$$\frac{\Gamma \rightarrow s \simeq t, s' \simeq t', \Delta}{\Gamma \rightarrow s \simeq t, \Delta} \;\; \mid s = s' \;\wedge\; t = t' \;\wedge\; s > t \;\wedge\; s > \Gamma \;\wedge\; greq(s \simeq t, \Delta)$$

where in the merging paramodulation rule $t|_p$ is not a variable.

The completeness proof for the resulting inference system $\mathcal{M}$ can be obtained by exactly the same construction for the rewrite system R and the same cases as for $\mathcal{I}$, except that where before equality factoring was needed, now either merging paramodulation or ordered factoring apply.

An important property of the inference system $\mathcal{M}$ is related to the following. For $\mathcal{G}$ and $\mathcal{H}$, it is not difficult to see that the model R^* constructed from S is (isomorphic to) the unique minimal Herbrand model of S: it is a Herbrand model, as we have shown, and it is minimal, since all rules of R are logical consequences of S. This turns out to be very useful in applications to inductive theorem proving (see Chapter [chapter with id comon] of this handbook). It is well-known that if $\mathcal{S}$ contains some non-Horn axiom, then in general a unique minimal Herbrand model of S no longer exists. For example, if $\mathcal{S} \equiv \{p \vee q\}$ then both the models $\{p\}$ and $\{q\}$ are minimal. The total reduction ordering $\succ$ on ground literals provides a way to single out one of the minimal models, the so-called *perfect model* (of $\mathcal{S}$ and $\succ$). The perfect model is the minimal one with respect to the (multi)set extension $\succ_{mul}^{-1}$ of $\succ^{-1}$. If $\mathcal{S} \equiv \{p \vee q\}$ where $p \succ q$ then $\{q\} \succ_{mul}^{-1} \{p\}$ and hence $\{p\}$ is the perfect model (see [Bachmair and Ganzinger 1991] for details).

Now it turns out that the model R^* obtained for sets of clauses closed under $\mathcal{M}$ is indeed the perfect one, which is not the case for the inference system $\mathcal{I}$ as shown in the following example[7]:

[7] By Leo Bachmair, private communication.

3.11. EXAMPLE. Assume we have $a \succ b \succ c \succ d$ and the following two clauses

$$\to\ b \simeq d$$
$$\to\ a \simeq b, a \simeq c$$

Then the closure w.r.t. $\mathcal{I}$ only introduces the new clause

$$b \simeq c\ \to\ a \simeq c$$

and a number of tautologies (that are not involved in the model construction). Therefore $R = \{b \Rightarrow d, a \Rightarrow b\}$, and the model $R^* = I$ is $\{b \simeq d, a \simeq b, a \simeq d\}$.

On the other hand, the closure by $\mathcal{M}$ produces the clause

$$\to\ a \simeq d, a \simeq c$$

apart from other clauses that are not relevant for the generation of R. In this case $R = \{b \Rightarrow d, a \Rightarrow c\}$, and the model $R^* = M$ is $\{b \simeq d, a \simeq c\}$.

Now we have that $I \succ_{mul}^{-1} M$, since after removing $b \simeq d$ in both sets $a \simeq d \succ^{-1} a \simeq c$, i.e., I is not minimal. □

In logic programming, perfect models give semantics for programs with negation (as failure), and the ordering $\succ$ is usually induced from the way non-Horn clauses are written: one positive atom is written in the head of the clause, and the other ones are written negatively in the tail. For instance, $p \vee q$ can be written $p :-\ \neg q$ or $q :-\ \neg p$. Heads are made big in the ordering. If the resulting ordering is well-founded then the program has a perfect model. Roughly, a logic program with negation is called (locally) *stratified* if there is some well-founded ordering on ground atoms such that for all ground instances of clauses the head is bigger than every negative atom in the tail, and bigger than or equal to every positive atom in the tail [Przymusiński 1988]. Local stratification is too strong a condition for the existence of a perfect model, and it has been relaxed into *weak stratification*, where only ground instances contributing to the model need to fulfill the requirements [Przymusińska and Przymusiński 1990]. These ideas are generalized and extended to arbitrary programs with equality in [Bachmair and Ganzinger 1991].

4. Saturation procedures

The completeness results presented until now only apply to closure procedures, that is, deduction procedures which compute the closure of an initial set of clauses under a given inference system, without considering simplification or deletion techniques. However, such techniques are well-known to be crucial for efficiency in paramodulation-based theorem proving. In this section we study their compatibility with refutation completeness.

4.1. Redundancy in practice

Let us first give some examples of practical simplification and deletion methods. Most provers apply these methods in two possible contexts. The first one, usually called *forward* redundancy elimination, is applied to new clauses immediately after they are obtained by an inference. For example, the conclusion of an inference can be simplified by rewriting it using other clauses before storing it. On the other hand, *backward* techniques are the ones applied to existing clauses, using newer ones that have been generated later on.

4.1. Example. Consider the lexicographic path ordering generated by the precedence $f \succ_{\mathcal{F}} a \succ_{\mathcal{F}} b$ and the following two equations whose maximal side is written underlined:

$$\begin{array}{ll} 1. & \underline{f(a,x)} \simeq x \\ 2. & \underline{f(x,a)} \simeq f(x,b) \end{array}$$

There is a superposition inference with conclusion

$$f(a,b) \simeq a$$

to which forward simplification can be applied by rewriting it with equation 1 into

$$b \simeq \underline{a}$$

Adding the result to the set, we obtain:

$$\begin{array}{lr} 1. & \underline{f(a,x)} \simeq x \\ 2. & \underline{f(x,a)} \simeq f(x,b) \\ 3. & \underline{a} \simeq b \end{array}$$

Now, by backward simplification using the new equation 3, equation 2 can be simplified into the tautology $f(x,b) \simeq f(x,b)$. The elimination of this tautology is another backward redundancy step. Furthermore, equation 1 can be simplified using 3 into

$$\underline{f(b,x)} \simeq x$$

Hence the final set will only contain the equations

$$\begin{array}{lr} 3. & \underline{a} \simeq b \\ 4. & \underline{f(b,x)} \simeq x \end{array}$$

□

In this section it is explained how redundancy elimination methods like the ones used in this example can be treated uniformly in the context of *saturation* procedures. Let us first give some informal intuition. The notion of saturation w.r.t. a given inference system $\mathcal{I}$ generalises the one of closure w.r.t. $\mathcal{I}$: roughly, a set of clauses S is *saturated* if S is closed under $\mathcal{I}$ *up to redundant inferences*. Refutation completeness then means that the empty clause $\Box$ is in S for every unsatisfiable saturated set of clauses S.

A procedure like the one of the previous example can be seen as a procedure computing a saturated set. Such a *saturation* procedure will be modelled by a *derivation*, a possibly infinite sequence of sets of clauses where each set can be obtained from the previous one in two possible ways: either by adding a clause or by removing a clause.

Two abstract notions of redundancy will play an essential role in saturation: one for clauses and one for inferences. Here we first explain them informally.

Roughly, a clause C is redundant w.r.t. a set S if C follows from clauses in S that are smaller than C. Redundant clauses correspond to the ones that are removed in a derivation. This abstract notion provides a useful means for proving the completeness of the inference system in combination with concrete practical (e.g., backward) redundancy elimination techniques.

Similarly, an inference will be redundant if its conclusion follows from clauses that are smaller than its maximal premise. In a derivation, conclusions of redundant inferences need not to be added. This abstract notion of redundant inference covers several powerful concrete forward redundancy techniques.

In the remainder of this section these ideas are formally developed and explained.

4.2. Redundancy and saturation in the ground case

In this section, as a simple example, saturation is described in detail for the case of ground clauses. Hence in this section all clauses (denoted by $C, D, \ldots$) and sets of clauses (denoted by S) are assumed to be ground. A two-stage approach is followed. First a "static" point of view is considered: it is proved that unsatisfiable saturated sets contain the empty clause. This involves the notion of redundant inference. After this, the "dynamic" problem of how to compute such saturated sets is considered, which is where the concepts of derivation and of redundant clauses are needed.

4.2.1. the static view

In the previous section we built a model R^* for any set S closed under $\mathcal{I}$ and such that $\Box \notin S$. Now our aim is to weaken the closedness requirement as much as possible into some notion of saturatedness. This could of course be done by defining a set S to be saturated whenever $R^* \models S$, but this would not be very useful in practice, since this (global) semantic property can almost never be checked. In order to find a useful practical notion of saturatedness, one can analyse the kind of inferences that are really needed in the proof showing that $R^* \models S$ for closed sets: the ones in which the rightmost premise is the minimal clause that is false in R^*. This leads us to the following.

We denote by $S^{\prec C}$ the set of all D in S such that $C \succ_c D$. An *inference* with maximal premise C and conclusion D is *is redundant with respect to a set* S if

$S^{\prec C} \models D$. A set of clauses S is *saturated* with respect to an inference system $\mathcal{I}nf$ if every inference of $\mathcal{I}nf$ with premises in S is redundant with respect to S.

4.2. THEOREM. *Let S be a set of ground clauses that is saturated with respect to $\mathcal{I}$. Then $R^* \models S$ if, and only if, $\square \notin S$, and hence S is unsatisfiable if, and only if, $\square \in S$.*

Proof. The only difference with Theorem 3.10 is that now a contradiction has to be obtained from the fact that the inferences between clauses in S are redundant instead of considering that the conclusion is in S. Let us show it for an inference by superposition right. In this case the minimal counterexample C has not generated a rule because its maximal term is reducible by a rule generated by a clause C'. Then the following inference by superposition right is considered where C' is the left premise, C is the right one and D is the conclusion.

$$\frac{\Gamma' \rightarrow l \simeq r, \Delta' \qquad \Gamma \rightarrow s \simeq t, \Delta}{\Gamma', \Gamma \rightarrow s[r]_p \simeq t, \Delta', \Delta}$$

As in Theorem 3.10, by using Lemma 3.9 from $R^* \not\models C$ we can infer that $R^* \not\models D$. But on the other hand, since the inference is redundant we have $S^{\prec C} \models D$, and by minimality of C we have $R^* \models D$ which is a contradiction. $\square$

4.2.2. the dynamic view: computing saturated sets

The previous theorem states that, instead of computing sets closed under the inference system, it suffices to saturate them. Therefore, now practical methods for computing saturated sets are defined. These methods are formalized by the notion of *derivation*, a sequence of sets of clauses where each time the next set is obtained by either adding some logical consequence or removing some *redundant* clause.

A ground *clause C is redundant* with respect to a set of ground clauses S if $S^{\prec C} \models C$. A *derivation* is a (possibly infinite) sequence of sets of clauses $S_0, S_1, \ldots$ where each S_{i+1} is either $S_i \cup \{C\}$, for some C such that $S_i \models C$, or $S_i \setminus \{C\}$, for some C that is redundant with respect to S_i. Clauses belonging, from some i on, to all S_k with $k > i$, are called *persistent*. The set S_∞ is the set of persistent clauses, defined $S_\infty = \cup_i \cap_{k>i} S_k$.

A nice property of the general notion of redundancy presented above is given by the following lemma. It states that all non-persistent clauses occurring in the derivation are redundant w.r.t. the set of persistent ones.

4.3. LEMMA. *Let $S_0, S_1, \ldots$ be a derivation and let C be a clause in $(\cup_i S_i) \setminus S_\infty$. Then C is redundant w.r.t. S_∞.*

Proof. We proceed by induction on C w.r.t. $\succ_c$. Since $C \notin S_\infty$ there is some S_j, s.t. C is redundant w.r.t. S_j, which implies that $S_j^{\prec C} \models C$, and hence by induction hypothesis $S_\infty^{\prec C} \models C$. $\square$

It is easy to see that simplification by demodulation fits into the notion of derivation. For example, simplifying $P(f(a))$ into $P(a)$ with the equation $f(a) \simeq a$ is modeled by first adding $P(a)$, and then removing $P(f(a))$ which has become redundant in the presence of $P(a)$ and $f(a) \simeq a$.

4.4. EXAMPLE. Consider the lexicographic path ordering generated by the precedence $P \succ_{\mathcal{F}} Q \succ_{\mathcal{F}} f \succ_{\mathcal{F}} a$. The table:

Refutation S	Comments	Derivation $S_0, S_1, \ldots$
1. $\rightarrow \underline{Q(a)}$ 2. $\underline{Q(a)} \rightarrow f(a) \simeq a$ 3. $P(a) \rightarrow$ 4. $\rightarrow P(f(a))$	Initial set of clauses	S_0
1. $\rightarrow Q(a)$ 2. $Q(a) \rightarrow f(a) \simeq a$ 3. $P(a) \rightarrow$ 4. $\rightarrow \underline{P(f(a))}$ 5. $\rightarrow \underline{f(a)} \simeq a$	Inf. 5 from 1 and 2	$S_0 \models (\rightarrow f(a) \simeq a)$ $S_1 = S_0 \cup \{\rightarrow f(a) \simeq a\}$
1. $\rightarrow Q(a)$ 2. $Q(a) \rightarrow f(a) \simeq a$ 3. $\underline{P(a)} \rightarrow$ 5. $\rightarrow f(a) \simeq a$ 6. $\rightarrow \underline{P(a)}$	Simplif. 4 to 6 with 5	$S_1 \models (\rightarrow P(a))$ $S_2 = S_1 \cup \{\rightarrow P(a)\}$ $\rightarrow P(f(a))$ is redundant w.r.t. S_2 $S_3 = S_2 \setminus \{\rightarrow P(f(a)\}$
1. $\rightarrow Q(a)$ 2. $Q(a) \rightarrow f(a) \simeq a$ 3. $P(a) \rightarrow$ 5. $\rightarrow f(a) \simeq a$ 6. $\rightarrow P(a)$ 7. $\square$	Inf. 7 from 6 and 3	$S_3 \models \square$ $S_4 = S_3 \cup \{\square\}$

represents a derivation performed by a theorem prover that is based on the strict superposition calculus and applies simplification by demodulation. The first column contains the set of ground clauses at every step of the refutation and the second one contains some explanations about the current step. In the third column the sets of the derivation are given and the changes justified. In the first column, the underlined subterms are the ones involved in the next inference. □

If our aim is to obtain refutation complete theorem proving procedures by computing derivations, and in the limit, to obtain a saturated set, then a notion of *fairness* is required. It roughly says that no inference π should be postponed forever: either some premise of π disappears at some point of the derivation, or else π has to become redundant at some point. One way of forcing π to become redundant is by adding its conclusion: for every ground inference π by our inference systems, always its conclusion is smaller than its maximal premise[8] and hence π is trivially redundant w.r.t. a set S containing its conclusion D (since D follows from a clause of S that is smaller than the maximal premise, namely D itself).

4.5. DEFINITION. A derivation $S_0, S_1, \ldots$ is *fair* with respect to an inference system $\mathcal{I}nf$ if for every inference π of $\mathcal{I}nf$ with premises in S_∞ there is some $j \geq 0$ s.t. π is redundant with respect to S_j.

Now we can prove that fair derivations compute (in the limit) saturated sets and generate the empty clause if and only if the initial set is unsatisfiable.

4.6. THEOREM. *If $S_0, S_1, \ldots$ is a fair derivation with respect to $\mathcal{I}nf$, then S_∞ is saturated with respect to $\mathcal{I}nf$, and hence if $\mathcal{I}nf$ is $\mathcal{I}$, then S_0 is unsatisfiable if, and only if, $\Box \in S_j$ for some j. Furthermore, if $S_0, S_1, \ldots, S_n$ is a fair derivation then S_n is saturated and logically equivalent to S_0.*

Proof. First we prove that S_∞ is saturated with respect to $\mathcal{I}nf$. By fairness, all inferences with premises in S_∞ are redundant in some S_j and hence, by lemma 4.3, redundant in S_∞, which implies that S_∞ is saturated.

Second, if $\mathcal{I}nf$ is $\mathcal{I}$, since S_∞ is saturated with respect to $\mathcal{I}$, by theorem 4.2 S_∞ is unsatisfiable if, and only if, $\Box \in S_\infty$. Since, definition of derivation, S_0 is logically equivalent to all S_i with $i \geq 0$ and, by lemma 4.3, to S_∞ as well, S_0 is unsatisfiable if, and only if, $\Box \in S_\infty$ and hence $\Box \in S_j$ for some j.

Finally if $S_0, S_1, \ldots, S_n$ then $S_\infty = S_n$ and hence S_n is saturated. □

Now, what can be done in practice to ensure fairness? On the one hand, it is needed that after adding the conclusion of an inference, the inference becomes redundant. As said, for the inference systems presented in this chapter, this is indeed

[8] For inference systems not satisfying this property, an inference should be redundant as well w.r.t. a set S if its conclusion is in S (and not only if its conclusion follows from clauses in S smaller than its maximal premise). Another possibility is to relax the notion of *fairness* (that will be introduced in a moment), requiring that either the conclusion of π belongs to some S_j or else π is redundant in S_j.

the case. In order to capture forward simplification we also want the inference to become redundant if we add the simplification of the conclusion. Indeed, with this notion of redundancy of inferences any clause smaller than the maximal premise can be used to simplify the conclusion into a smaller clause.

On the other hand, this has to be done for all inferences with persistent premises. But since it is not possible to know, at a certain point S_i, whether a given premise is going to be persistent or not, some means should be provided ensuring that no inference with persistent premises is postponed infinitely many times. In most implementations this is achieved by periodically considering all inferences with the clause whose *size* (in number of symbols) is smallest. If a certain clause persists then it will eventually be considered, since there are only finitely many clauses with smaller size.

4.3. Non-ground saturation procedures

For the non-ground case, the definitions for redundancy of inferences and clauses of the previous section are straightforwardly extended (roughly, C or π is redundant if all its ground instances are) and the notions of derivation and saturatedness do not change. Here we consider saturation without constraint inheritance, that is, the S_i occurring in derivations are sets of clauses without constraints, and if fairness requires a non-ground inference π with conclusion $D \mid s = t \wedge OC$ to become redundant in some S_j, then this is done by adding $D\sigma$ to S_j, where $\sigma = mgu(s,t)$.

In the following, $gnd(C)$ denotes the set of all ground instances of a clause C, and if S is a set of clauses then $gnd(S)$ denotes $\cup_{C \in S}\; gnd(C)$.

Let π be an inference with premises $C_1, \ldots, C_n$ and conclusion $D \mid T$. Then a *ground instance* $\pi\sigma$ *of the inference* π is an inference with premises $C_1\sigma, \ldots, C_n\sigma$ and conclusion $D\sigma$ for some ground σ such that $\sigma \models T$. An inference π is *redundant* with respect to a set S if all its ground instances are redundant with respect to $gnd(S)$. Note that an inference whose conclusion has an unsatisfiable constraint is redundant since it has no ground instances.

4.7. EXAMPLE. Consider the lexicographic path ordering generated by the precedence $P \succ_{\mathcal{F}} f \succ_{\mathcal{F}} h \succ_{\mathcal{F}} g \succ_{\mathcal{F}} a$ and the following set S of equations whose maximal sides are written underlined:

$$\begin{array}{rr}
1. & \underline{g(x)} \simeq x \\
2. & \underline{h(a,z)} \simeq z \\
3. & \underline{f(x,h(x,y))} \simeq g(y) \\
4. & \underline{f(a,z)} \simeq z
\end{array}$$

The inference between 2 and 3 can be shown redundant w.r.t. S using rewriting as follows. It has the conclusion

$$f(x,z) \;\simeq\; g(y) \;\mid h(a,z) = h(x,y) \;\wedge h(a,z) > z \wedge f(x,h(x,y)) > g(y)$$

Once it is checked that the constraint is satisfiable, the most general unifier $\{x \mapsto a, z \mapsto y\}$ of the unification problem in the constraint is applied to the conclusion, obtaining:

$$f(a,y) \simeq g(y)$$

It has to be shown that all its ground instances, which are of the form $f(a,t) \simeq g(t)$, follow from instances of S smaller than the corresponding instance of the maximal premise, which is $f(a,h(a,t)) \simeq g(t)$. This can be done by rewriting: both sides of $f(a,y) \simeq g(y)$ rewrite into y using equations 4 and 1.

As another example of forward simplification, assume the set consists only of equations 1, 2 and 3. The inference between 2 and 3 we have seen generates $f(a,y) \simeq g(y)$, which by forward simplification with 1 produces equation 4. $\square$

It is easy to show, by a similar lifting argument as the one used in Theorem 3.6, that the non-ground version of Theorem 4.2 holds.

4.8. THEOREM. *Let S be a set of clauses that is saturated with respect to $\mathcal{I}$. Then, S is unsatisfiable if, and only if, $\square \in S$.*

Now we can again focus on the problem of how to compute (non-ground, this time) saturated sets. For this purpose, in this context a clause C is *redundant* with respect to a set S if all its ground instances are redundant with respect to $gnd(S)$.

The notions of non-ground derivation, persistence and fairness are defined exactly as in the ground case. The non-ground versions of Lemma 4.3 and Theorem 4.6 can be proved in a similar way.

4.9. THEOREM. *If $S_0, S_1, \ldots$ is a fair derivation with respect to $\mathcal{I}nf$, then S_∞ is saturated with respect to $\mathcal{I}nf$, and hence, if $\mathcal{I}nf$ is $\mathcal{I}$, then S_0 is unsatisfiable if, and only if, $\square \in S_j$ for some j. Furthermore, if $S_0, S_1, \ldots, S_n$ is a fair derivation then S_n is saturated and logically equivalent to S_0.*

4.10. EXAMPLE. Let us now consider a more complicated example showing the power of the notion of redundancy for inferences, where moreover the generated ordering constraints are not ignored like in Example 4.7, but play a crucial role.

Consider the transitivity axiom for a predicate p:

$$p(x,y) \wedge p(y,z) \rightarrow p(x,z)$$

Consequences by superposition left (or resolution) of this clause are:

$$p(x,y) \wedge p(y,z) \wedge p(z,u) \rightarrow p(x,u)$$
$$p(x,y) \wedge p(y,z) \wedge p(z,u) \wedge p(u,w) \rightarrow p(x,w)$$
$$\ldots$$

We first show that these consequences are not redundant clauses in the presence of the transitivity axiom. An instance of $p(x,y) \wedge p(y,z) \wedge p(z,u) \rightarrow p(x,u)$ of the form

$$p(a,b) \wedge p(b,c) \wedge p(c,d) \rightarrow p(a,d)$$

only follows from instances of the transitivity axiom

$$p(b,c) \wedge p(c,d) \rightarrow p(b,d) \tag{4.1}$$

$$p(a,b) \wedge p(b,d) \rightarrow p(a,d) \tag{4.2}$$

$$p(a,b) \wedge p(b,c) \rightarrow p(a,c) \tag{4.3}$$

$$p(a,c) \wedge p(c,d) \rightarrow p(a,d) \tag{4.4}$$

in two possible ways: from (4.1,4.2) or from (4.3,4.4). However, if $b \succ_{\mathcal{F}} a \succ_{\mathcal{F}} c \succ_{\mathcal{F}} d$ then in both cases the instances used are not smaller than the instance that has to be proved redundant. Therefore, the clause $p(x,y) \wedge p(y,z) \wedge p(z,u) \rightarrow p(x,u)$ is not redundant. But we can prove that the two possible resolution inferences producing it from the transitivity axiom are indeed redundant.

One of the two inferences is

$$\frac{p(x,y) \wedge p(y,u) \rightarrow p(x,u) \qquad\qquad p(y,z) \wedge p(z,u) \rightarrow p(y,u)}{p(x,y) \wedge p(y,z) \wedge p(z,u) \rightarrow p(x,u) \ \mid\ \begin{array}{l} p(y,u) > p(x,u) \ \wedge\ p(y,u) > p(x,y) \ \wedge \\ p(y,u) > p(y,z) \ \wedge\ p(y,u) > p(z,u) \end{array}}$$

in which the unifier has already been applied. The constraint of the conclusion can be simplified into $y > x \ \wedge\ u > z \ \wedge\ y > z$. Now the ground instances of the conclusion that indeed satisfy this constraint follow from smaller instances of the set 4.1–4.4, i.e., the inference is redundant.

Another difficult question is: how to find in practice, and automatically, the appropriate clauses and their instances that allow us to prove such redundancies? In the Saturate system [Nivela and Nieuwenhuis 1993, Ganzinger, Nieuwenhuis and Nivela 1999], several such concrete techniques are implemented. For this concrete example, Saturate proves the redundancies automatically by *clausal rewriting* combined with LPO constraint solving. □

4.4. More general notions of redundancy for clauses

As said, the notion of redundancy of clauses given in the previous section together with the notion of derivation can capture simplification methods like demodulation by rewriting. However, it cannot capture useful methods like *subsumption* in its full generality. For instance, a clause $P(a)$ (for some constant a) cannot be proved redundant w.r.t. a set containing $P(x)$, since only strictly smaller instances can be used in the redundancy proof.

To overcome this problem, the notion of redundancy of clauses can be made more powerful by applying not only smaller instances but also equal instances in the redundancy proof.

We denote by $S^{\preceq C}$ the set of all D in S such that $C \succeq_c D$. Then a clause C is *non-strictly redundant* w.r.t. a set of clauses S if $gnd(S)^{\preceq D} \models D$ for all D in $gnd(C)$. Note that it is equivalent to the requirement that for all D in $gnd(C)$

either $gnd(S)^{\prec D} \models D$ or $D \in gnd(S)$. This means that one only needs to care about all those ground instances of C that do not belong to S. It is easy to see that this notion of redundancy covers subsumption.

4.11. EXAMPLE. The clause $P(a)$ is redundant w.r.t. $\{P(x)\}$, since $P(a) \in gnd(P(x))$. But if the signature under consideration is fixed, then it is possible to go beyond.

Assume $\mathcal{F}$ is $\{a, f, Q, P\}$, let C be the clause $Q(x) \vee P(x)$, and let S be the set of clauses $\{\ P(f(y)), \quad Q(a) \vee P(a)\ \}$.

Then C can be proved non-strictly redundant w.r.t. the set of clauses S as follows. The ground instance $Q(a) \vee P(a)$ of C is in S and is hence redundant w.r.t. S. The remaining ground instances of C are of the form $Q(f(t)) \vee P(f(t))$ for some ground term t, which are redundant since $P(f(t)) \models Q(f(t)) \vee P(f(t))$ and $Q(f(t)) \vee P(f(t)) \succ_c P(f(t))$.

Note that some instances of C have been proved strictly redundant, i.e., using smaller instances w.r.t. $\succ_c$, and others have been proved non-strictly redundant, that is using equal instances in $gnd(S)$. □

In this new setting with non-strict redundancy, the notion of derivation has to be slightly modified. If S_{i+1} is $S_i \setminus \{C\}$, now we require C to be (non-strictly) redundant w.r.t. $S_i \setminus \{C\}$, instead of w.r.t. S_i as before (otherwise all clauses C in S_i could be removed!).

Unfortunately, this stronger notion of redundancy for clauses has some side effects on the notion of fairness, mainly because there might be persistent ground instances that do not correspond to any persistent clause.

4.12. EXAMPLE. Assume $\mathcal{F} = \{a, P\}$ and the following derivation:
$S_0 = \{\neg P(x), P(x)\}$,
$S_1 = \{\neg P(x), P(x), P(a)\}$,
$S_2 = \{\neg P(x), P(a)\}$,
$S_3 = \{\neg P(x), P(x), P(a)\}$,
$S_4 = \{\neg P(x), P(x)\}$,
...
The only instance of $P(x)$ is $P(a)$. Therefore $P(x)$ is redundant in the presence of $P(a)$, and vice versa, and hence the sequence $S_0, S_1, \ldots$ is indeed a derivation. The only persistent clause is $\neg P(x)$, and no inference between persistent clauses exists. Hence the empty clause will not be generated in this derivation. This problem is clearly due to the fact that fairness, as it was stated in the previous subsection for the weaker notion of redundancy of clauses, only requires inferences between persistent clauses to be considered. □

In the previous example there is a clause $P(a)$, which is a *persistent ground instance,* i.e. a ground clause C such that from some k on, C belongs to all $gnd(S_i)$ with $i > k$, which is not an instance of any persistent clause. Therefore a simple way to overcome this problem is to modify the notion of fairness by requiring in addition that, roughly, the set of persistent ground instances is covered by the set of persistent clauses. This idea is formalised as follows.

4.13. DEFINITION. The set $G_\infty = \cup_i \cap_{k>i}\, gnd(S_i)$ is the set of *persistent ground instances*. A derivation $S_0, S_1, \ldots$ is *ground fair* with respect to an inference system $\mathcal{I}nf$ if $gnd(S_\infty) \supseteq G_\infty$ and all inferences of $\mathcal{I}nf$ with premises in S_∞ are redundant w.r.t. S_j for some j.

Ground fairness can be achieved in practical theorem provers by associating to each clause a counter indicating its "level" of non-strict redundancy steps, and forbidding such non-strict redundancy steps beyond a certain level. In most implementations the problem of the previous example is avoided automatically, since, as said, fairness is achieved by periodically considering all inferences with the smallest clause with respect to size. If a certain ground instance persists, then at any point it is an instance of some clause with smaller or equal size, and hence it will eventually be considered, because there are only finitely many such clauses with smaller size.

The non-ground version of Lemma 4.3 can be proved for non-strict redundancy in the same way as before, using the fact that $gnd(S_\infty) \supseteq G_\infty$. Theorem 4.9 holds as well.

4.5. Computing with saturated sets

In practice, it is sometimes possible to obtain a finite saturated set S_n (not containing the empty clause) for a given input S_0. In this case its satisfiability has been proved. Let us give an example.

4.14. EXAMPLE. Consider the lexicographic path ordering generated by the precedence $P \succ_{\mathcal{F}} Q \succ_{\mathcal{F}} f \succ_{\mathcal{F}} g \succ_{\mathcal{F}} a$. The following table represents a finite derivation that terminates with a saturated set. The concrete redundancy method applied is simplification by demodulation. The first column contains the set of clauses at each step of the derivation. In the second column the sets of the derivation are given and the changes justified. In the first column, the underlined terms are the ones involved in the next inference.

The final set S_4 is saturated since all inferences with premises in S_4 are redundant:

1. the inferences between 3 and 4 and between 2 and 6 are redundant since their conclusions are in S_4 (and hence they follow from clauses smaller than the maximal premise).
2. the inference between 5 and 6 is also redundant, although its conclusion is not in S_4. Let us show it. The inference has premises $\underline{g(g(y))} \simeq g(y)$ (note that we have renamed the variables of 5 to avoid name clashes) and $Q(g(x)) \rightarrow \underline{P(g(x))}$, with the unifier $\sigma = \{x \mapsto g(y)\}$. The conclusion is $Q(g(g(y))) \rightarrow \overline{P(g(y))}$, which can be rewritten by 5 into $Q(g(y)) \rightarrow P(g(y))$, which is smaller than 6σ and belongs (up to renaming of variables) to S_4, and hence the inference is redundant. □

Derivation $S_0, S_1, \ldots$	Comments
1. $Q(g(g(x))) \rightarrow P(g(x))$ 2. $P(g(a)) \rightarrow$ 3. $\rightarrow \underline{f(x,y)} \simeq g(x)$ 4. $\rightarrow \underline{f(x,a)} \simeq g(g(x))$	Initial set of clauses S_0
1. $\underline{Q(g(g(x)))} \rightarrow P(g(x))$ 2. $P(g(a)) \rightarrow$ 3. $\rightarrow f(x,y) \simeq g(x)$ 4. $\rightarrow f(x,a) \simeq g(g(x))$ 5. $\rightarrow \underline{g(g(x))} \simeq g(x)$	Infer 5 from 3 and 4 $S_1 = S_0 \cup \{5\}$
2. $\underline{P(g(a))} \rightarrow$ 3. $\rightarrow f(x,y) \simeq g(x)$ 4. $\rightarrow f(x,a) \simeq g(g(x))$ 5. $\rightarrow g(g(x)) \simeq g(x)$ 6. $Q(g(x)) \rightarrow \underline{P(g(x))}$	Simplify 1 into 6 with 5 $S_1 \models 6$ $S_2 = S_1 \cup \{6\}$ 1 is redundant w.r.t. $S_2 \setminus \{1\}$ $S_3 = S_2 \setminus \{1\}$
2. $\underline{P(g(a))} \rightarrow$ 3. $\rightarrow f(x,y) \simeq g(x)$ 4. $\rightarrow f(x,a) \simeq g(g(x))$ 5. $\rightarrow g(g(x)) \simeq g(x)$ 6. $Q(g(x)) \rightarrow \underline{P(g(x))}$ 7. $Q(g(a)) \rightarrow$	Infer 7 from 2 and 6 $S_3 \models 7$ $S_4 = S_3 \cup \{7\}$

The remainder of this section is on the applications of such finite saturation derivations. On the one hand, the existence of a finite saturated set S not containing the empty clause implies that its consistency has been proved. Consistency proving has many applications and is also closely related to inductive theorem proving, as shown in [chapter with id comon] of this handbook.

But on the other hand, theorem proving in theories expressed by saturated sets S of axioms is also interesting because more efficient proof strategies become (refutationally) complete. For example, it is clear from the previous section that when saturating $S \cup S'$, for some S', inferences all whose premises belong to S are redundant in $S \cup S'$, for the following reason. Since S is saturated, these inferences are redundant in S, and hence as well in any set containing S, because the redundancy notions are easily shown to be monotonic in this sense.

This leads to the completeness of the the set-of-support strategy, where S' is the set of support. This strategy is complete for standard binary resolution, but is incomplete in general for ordered inference systems and also for paramodulation ([Snyder and Lynch 1991] describe a lazy paramodulation calculus that is complete with set of support).

4.6. Completion as an instance of saturation

In some cases, depending on the syntactic properties of the given finite saturated set S, decision procedures for the given theory are obtained. This is the case for instance for saturated sets of unit equations E, where the saturation process behaves like unfailing Knuth-Bendix completion. Clearly, simplification by rewriting and removing tautologies $s \simeq s$ (and other more refined techniques) fit into the redundancy notions. Furthermore, indeed rewriting with the final saturated set provides a decision procedure for the word problem.

Let $\succ$ be a total reduction ordering, and let E be a set of unit equations. Furthermore, let $\rightarrow_E$ be the ordered rewrite relation (remember that $\rightarrow_E$ is the smallest monotonic relation on terms such that $s\sigma \rightarrow_E t\sigma$ whenever $s \simeq t$ is in E and $s\sigma \succ t\sigma$).

4.15. THEOREM. *If E is a set of unit equations that is saturated w.r.t. $\mathcal{H}$ and $\succ$, then every ground term s has a unique normal form $nf(s)$ w.r.t. $\rightarrow_E$. Furthermore, for every pair of ground terms s and t, $E \models s \simeq t$ if, and only if, $nf(s) \equiv nf(t)$.*

Proof. It is shown that every ground term s (possibly with new Skolem constants for its variables) can be rewritten into the unique minimal (w.r.t. $\succ$) representative of its E-congruence class. By induction w.r.t. $\succ$, it suffices to prove the reducibility w.r.t. $\rightarrow_E$ of non-minimal s. Let u be this minimal representative of the congruence class of s. Since $s \succ u$, the only inference rule that can be applied in a refutation of $s \simeq u \rightarrow$ are strict superposition left steps on s with some positive equation $l \simeq r$ of E. But then s is reducible by rewriting with $l \simeq r$, since $s|_p = l\sigma$ for some p. $\Box$

In fact, similar results apply as well to the other forms of saturatedness that will be introduced later on in this chapter (modulo equational theories, with constraint inheritance).

Decision procedures are also obtained for the ground case. For the ground Horn inference system $\mathcal{G}$ applied with eager selection, clearly each inference of $l \simeq r$ on a clause D produces a smaller clause D'. Furthermore, D is a logical consequence of the smaller clauses $l \simeq r$ and D', i.e., D has become redundant and can be removed. Hence after each inference, the clause set decreases w.r.t. the well-founded multiset extension of the clause ordering and hence the process terminates, thus deciding satisfiability.

4.16. THEOREM. *Superposition with selection decides the satisfiability of sets of ground Horn clauses.*

Furthermore, a decision procedure for the satisfiability of sets of arbitrary ground clauses is obtained by first transforming into Horn clauses (where $S \cup C \vee A_1 \vee \ldots \vee A_n$ is split into the disjunction of sets S_i of the form $S \cup C \vee A_i$; then S is satisfiable if some of the S_i is).

4.17. EXAMPLE. Note, however, that ground saturation procedures without redundancy do not always terminate, in spite of the fact that only smaller ground clauses are created in a well-founded ordering. Consider an LPO with $a \succ_{\mathcal{F}} f \succ_{\mathcal{F}} b$ and the initial two ground equations

$$\begin{array}{l} 1.\ f(\underline{a}) \simeq a \\ 2.\ f(b) \simeq \underline{a} \end{array}$$

Then infinitely many equations i, for $i > 2$ are generated by superposition at the underlined subterm between equation 1 and equation $i - 1$

$$\begin{array}{lr} 3. & f(f(b)) \simeq \underline{a} \\ 4. & f(f(f(b))) \simeq \underline{a} \\ 5. & f(f(f(f(b)))) \simeq \underline{a} \\ \vdots & \end{array}$$

□

Other syntactic restrictions on non-equational saturated sets S that are quite easily shown to produce decision procedures include *reductive* Horn clauses (also called conditional equations) or *universally reductive* general clauses [Bachmair and Ganzinger 1994*b*]. In such cases, the non-redundant inferences in a refutation of $S \cup G$ for certain classes of ground clauses G only produce new smaller ground clauses and saturation terminates by a similar argument as in the ground case. This kind of ideas provide several directions in which the previous two theorems can be generalized (see also Section 8.2 of this chapter).

4.7. Extended signatures

When applying the results we have seen so far for computing with sets $S \cup S'$ where S is saturated, one aspect has to be considered carefully: what happens if new (e.g. Skolem) symbols appear in S'?

4.18. EXAMPLE. Suppose S is the following set of unit equations:

$$\{\rightarrow f(x) \simeq a, \qquad \rightarrow g(a) \simeq a\}$$

under a lexicographic path ordering with the precedence $g \succ_{\mathcal{F}} a \succ_{\mathcal{F}} f$.

This set is saturated with respect to *the given signature*: the only possible inference with the two equations of S not needed, since its conclusion, $g(f(x)) \simeq a \mid a > f(x)$, has an unsatisfiable constraint $a > f(x)$ because a is the smallest constant symbol. From the rewriting and Knuth-Bendix completion point of view, S being saturated with respect to a given signature means that $\rightarrow_S$ is confluent for rewriting terms built over this signature, i.e., it is *ground confluent*, which is a weaker property than general confluence.

Now let us try to prove, for instance, that $S \models \forall y \;\; g(f(y)) \simeq a$. After negating and Skolemizing the goal $G = g(f(b)) \simeq a \rightarrow$ is obtained, which has to be refuted, where b is a new Skolem constant. Then, under the new extension of the precedence $g \succ_{\mathcal{F}} a \succ_{\mathcal{F}} f \succ_{\mathcal{F}} b$, the set-of-support strategy fails: no inferences can be computed between equations in S and G, but $S \cup \{G\}$ is inconsistent. Equivalently, from the rewriting point of view, $S \models G$ but G is in normal form with respect to $\rightarrow_S$.

This incompleteness is due to the fact that S is not saturated with respect to the new signature (note that a is no longer the smallest constant symbol). If S is instead saturated with respect to *extended* signatures then the inference with conclusion $g(f(x)) \simeq a \mid a > f(x)$ should be performed, since the constraint $a > f(x)$ is satisfiable in some extension of the signature. Then this incompleteness problem does not appear. □

From the previous example we learn that for some applications it is necessary to solve the ordering constraints under extended signatures (see Section 7 for details on ordering constraint solving). Similar incompleteness problems appear if we apply redundancy methods that rely on the given signature like, for instance, the one used in example 4.11.2.

Let us now consider the combination of two finite sets of clauses S_1 and S_2 (built over $\mathcal{F}_1$ and $\mathcal{F}_2$ resp.) that are saturated with respect to $\succ_1$ and $\succ_2$ respectively. Then an extension of the set-of-support-strategy applies: no inferences have to be considered in which all premises are in S_1 or all premises in S_2. Therefore, if $\mathcal{F}_1 \cap \mathcal{F}_2 = \emptyset$ then $S_1 \cup S_2$ is saturated.

Again here it is needed that S_1 and S_2 are saturated under the semantics in which the satisfiability of the constraints has been considered with respect to extended signatures, or at least with respect to a signature containing $\mathcal{F}_1 \cup \mathcal{F}_2$. Otherwise, situations similar to the example above can again appear.

Furthermore, it is necessary to find an ordering $\succ$ with all the required properties containing both $\succ_1$ and $\succ_2$. If $\succ_1$ and $\succ_2$ are two orderings of the same family of path orderings and this family is stable under extensions of the precedence (e.g. RPO is such a family) then one can define a precedence $\succ_{\mathcal{F}_1 \cup \mathcal{F}_2}$ extending $\succ_{\mathcal{F}_1}$ and $\succ_{\mathcal{F}_2}$ (whenever $\succ_{\mathcal{F}_1}$ and $\succ_{\mathcal{F}_2}$ are not contradictory, that is, $f, g \in \mathcal{F}_1 \cap \mathcal{F}_2$ and $f \succ_{\mathcal{F}_1} g$ implies $f \succ_{\mathcal{F}_2} g$). This produces a total extension. See [Rubio 1995] for related results on combining arbitrary orderings.

5. Paramodulation with constrained clauses

In this section we develop strategies where the ordering and/or equality restrictions of the inferences are kept in constraints and inherited between clauses. As explained in Section 1, this produces a further pruning of the search space. For simplicity reasons, first only Horn clauses are considered and at the end of the section the extension to general constrained clauses is outlined.

5.1. Equality constraint inheritance: basic strategies

We now analyse the first constraint-based restriction of the search space: the so-called *basicness restriction*, where superposition inferences on subterms created by unifiers on ancestor clauses are not performed. This restriction is conveniently expressed by inheriting the equality constraints without applying (or even computing) any unifiers. Hence from now on we consider sets of *constrained* clauses, rather than unconstrained ones, as in the previous sections.

5.1. DEFINITION. In the following, we call a set of constrained Horn clauses S *closed under* $\mathcal{H}$ *with equality constraint inheritance* if $D \mid T_1 \wedge \ldots \wedge T_n \wedge s = t$ is in S whenever $C_1 \mid T_1\ ,\ \ \ldots,\ \ C_n \mid T_n$ are clauses in S and there is an inference by $\mathcal{H}$ with premises $C_1, \ldots, C_n$ and conclusion $D \mid s = t \wedge OC$ such that the constraint $T_1 \wedge \ldots \wedge T_n \wedge s = t \wedge OC$ is satisfiable.

This strategy is incomplete in general: the closure under $\mathcal{H}$ with equality constraint inheritance of an unsatisfiable set of constrained Horn clauses needs not contain the empty clause.

5.2. EXAMPLE. Let $\succ$ be the lexicographic path ordering where $a \succ_{\mathcal{F}} b$. Consider the following unsatisfiable clause set S:

$$\begin{array}{ll} 1. & \quad \rightarrow a \simeq b \\ 2. & \quad \rightarrow P(x) \mid x = a \\ 3. & P(b) \rightarrow \end{array}$$

S is clearly closed under $\mathcal{H}$ with equality constraint inheritance, since no inferences by $\mathcal{H}$ that are compatible with the constraint of the second clause can be made. We have $a \succ_{\mathcal{F}} b$ and hence the first clause could only be used by superposing a on some non-variable subterm, while superposition left (i.e., resolution) between 2 and 3 leads to a clause with an unsatisfiable constraint $x = a \wedge b = x$. However, S does not contain the empty clause. This incompleteness is due to the fact that the usual lifting arguments, like the ones in Theorem 3.6, do not work here, since they are based on the existence of *all* ground instances of the clauses. Note that this is not the case here: the only instance of the second clause is $P(a)$, whereas the lifting argument in Theorem 3.6 requires the existence of the instance $P(b)$. □

Fortunately, the strategy is complete for what we will call *well-constrained* sets of clauses, which turn out to be adequate for many practical situations. A key idea in this context is the following (quite intuitive) notion of *irreducible ground substitution*. Let R be a ground rewrite system contained in the given ordering $\succ$ (that is, $l \succ r$ for all rules $l \Rightarrow r$ of R). A ground substitution σ is *reducible* by R if $x\sigma$ is reducible by R for some $x \in Dom(\sigma)$; if there is no such x then σ is *irreducible*. Furthermore, if S is a set of constrained clauses, then $irred_R(S)$ is its set of *irreducible instances*, that is, the set of ground instances $C\sigma$ of clauses $C \mid T$ in S such that σ is a solution of T and $x\sigma$ is irreducible by R for all $x \in vars(C)$.

5.3. DEFINITION. A set of constrained clauses S is *well-constrained* if either there are no clauses with equality predicates in S or else for all R contained in $\succ$ we have $irred_R(S) \cup R \models S$.

5.4. EXAMPLE. (Example 5.2 continued) The clause set S of the previous example is not well-constrained: if R is $\{a \Rightarrow b\}$ then the instance $P(a)$ of the second clause is not a logical consequence of $irred_R(S) \cup R$ (in fact, the second clause has no irreducible instances for this R). □

Let us give some more intuition behind the definition of well-constrained sets. For clauses without equality predicates, the situation is clear: all such sets are well-constrained (this is why logic programming without equality is compatible with arbitrary constraint systems).

Now let us consider clause sets S including equality predicates. First, note that if S is a well-constrained set, so is its closure w.r.t. any sound inference system, since the property of well-constrainedness is preserved under the addition of logical consequences. Second, it is not difficult to see that if all clauses in S have only tautologic constraints then S is well-constrained: every instance $C\sigma$ is either in $irred_R(S)$, or else σ is reducible by R. Then σ can be reduced into a "normal form" σ', where $C\sigma'$ is in $irred_R(S)$, and we have $irred_R(S) \cup R \models C\sigma$.

But there are other non-trivial examples of well-constrained sets.

5.5. EXAMPLE. Let $\succ$ be the lexicographic path ordering where $g \succ_{\mathcal{F}} a \succ_{\mathcal{F}} f \succ_{\mathcal{F}} b$. Then, constrained clauses like $g(x,x) \simeq b \mid a > x$ may appear in well-constrained sets, since the variable x is not "lower bounded": as for unconstrained clauses, for all σ the term $x\sigma$ can be reduced into a "normal form" $x\sigma'$, where $g(x\sigma', x\sigma') \simeq b$ is in $irred_R(S)$, and hence we have $irred_R(S) \cup R \models g(x\sigma, x\sigma)$. Here $g(x,x) \simeq b \mid a > x$ denotes the infinite set of clauses of the form $g(f^n(b), f^n(b)) \simeq b$ for $n \geq 0$, that is, $g(b,b) \simeq b$, $g(f(b), f(b)) \simeq b$, $g(f(f(b)), f(f(b))) \simeq b \ldots$ Note that such (in this case even non-regular) tree languages cannot be captured by standard first-order clauses. □

Furthermore, it will become clear from the completeness proof below that the notion of well-constrained clause could be modified in order to capture more cases by not considering *all* R contained in $\succ$, but only those R whose rules could be generated in the model generation technique applied to the given clause set. Then, one can know in advance that certain (e.g., constructor) terms will be irreducible w.r.t. such R. Here we have not done this in order to keep the definition of well-constrainedness simple.

The refutation completeness of $\mathcal{H}$ for well-constrained clause sets S can now be established by applying a simple variant of the model generation technique. Before we give the formal proof, let us explain the main ideas. Let S be a set of well-constrained clauses that is closed under $\mathcal{H}$ with equality constraint inheritance, and assume $\square \notin S$. As always, we show that then S is satisfiable by generating a rewrite system R for S (in a similar way as before) and then proving that $R^* \models S$.

For this purpose, we first show that $R^* \models irred_R(S)$ like in Theorem 3.6, but where the lifting case never needs to be applied (since we only consider the set of irreducible instances of S). Once we have $R^* \models irred_R(S)$, then also $R^* \models S$, since of course $R^* \models R$ and by well-constrainedness of S (where well-constrainedness is required only with respect to the particular R that has been generated) we have $irred_R(S) \cup R \models S$ (note that if there are no equality literals in S then $irred_R(S)$ coincides with S).

5.6. THEOREM. *The inference system $\mathcal{H}$ is refutation complete with equality constraint inheritance for well-constrained sets S of Horn clauses.*

Proof. Let S be closed under $\mathcal{H}$ with equality constraint inheritance. Again we build a model R^* for S whenever $\square \notin S$. As said, we prove that $R^* \models irred_R(S)$, which implies $R^* \models S$ by well-constrainedness.

We build R as for Theorem 3.6, but now only the irreducible (w.r.t. R_C) instances of S contribute to its construction: a ground instance C of the form $\Gamma \rightarrow l \Rightarrow r$ in $irred_{R_C}(S)$ generates the rule $l \Rightarrow r$ of R if the usual conditions (i), (ii) and (iii) apply.

Now again we derive a contradiction from the existence of a minimal (w.r.t. $\succ_c$) ground instance $C\sigma \in irred_R(S)$ for some $C \mid T \in S$, where σ is a solution of T, such that $R^* \not\models C\sigma$. Again we consider several cases, depending on the occurrences in $C\sigma$ of its maximal term $s\sigma$. Let us analyse only the case where C is $\Gamma, s \simeq t \rightarrow \Delta$ and $s\sigma \succ t\sigma$. Since $R^* \not\models C\sigma$, we have $R^* \models s\sigma \simeq t\sigma$, and hence the term $s\sigma$ is reducible by some rule $l\sigma \Rightarrow r\sigma \in R$, generated by an instance $C'\sigma$ of some $C' \mid T'$, where C' is of the form $\Gamma' \rightarrow l \simeq r$.

Now we have $s\sigma|_p = l\sigma$, and, since σ is irreducible by R, the only possibility is now that $s|_p$ is a non-variable position of s. Then there exists an inference by superposition left:

$$\frac{\Gamma' \rightarrow l \simeq r \qquad \Gamma, s \simeq t \rightarrow \Delta}{\Gamma', \Gamma, s[r]_p \simeq t \rightarrow \Delta \quad | \, s|_p = l \;\wedge\; l > r \;\wedge\; l > \Gamma' \;\wedge\; s > t \;\wedge\; s \geq \Gamma, \Delta}$$

whose conclusion has an instance $D\sigma$ where σ is a solution of $T \;\wedge\; T' \;\wedge\; s|_p = l \;\wedge\; l > r \;\wedge\; l > \Gamma' \;\wedge\; s > t \;\wedge\; s \geq \Gamma, \Delta$ such that $C\sigma \succ_c D\sigma$ and where $R^* \not\models D\sigma$. Furthermore, $D\sigma \in irred_R(S)$: indeed $x\sigma$ is irreducible by R for all variables $x \in vars(D)$. This is clearly the case if $x \in vars(C)$. For $x \in C'$, there are two cases: if $x \equiv l$ then $x \notin vars(D)$ since $l\sigma \succ r\sigma, \Gamma'\sigma$; if $x \not\equiv l$ then $x\sigma$ is irreducible w.r.t. $R_{C'}$ by construction of R, and hence also w.r.t. R, since for all rules $l' \Rightarrow r' \in R \setminus R_{C'}$ we have $l' \succeq l\sigma \succ x\sigma$ and hence such rules cannot reduce $x\sigma$. Altogether, this contradicts the minimality of $C\sigma$. □

5.2. Ordering constraint inheritance

The proof ideas used for equality constraint inheritance apply as well to ordering constraint inheritance or to a combination of both.

A set of constrained Horn clauses S is *closed under $\mathcal{H}$ with ordering constraint inheritance* if $(D \mid T_1 \wedge \ldots \wedge T_n \wedge OC\ )\sigma$ is in S whenever $C_1 \mid T_1\ ,\ \ \ldots,\ \ C_n \mid T_n$ are clauses in S and there is an inference by $\mathcal{H}$ with premises $C_1, \ldots, C_n$ and conclusion $D \mid s = t \wedge OC$ such that $\sigma = mgu(s,t)$ and the constraint $T_1 \wedge \ldots \wedge T_n \wedge s = t \wedge OC$ is satisfiable.

5.7. THEOREM. *The inference system $\mathcal{H}$ is refutation complete with ordering constraint inheritance for well-constrained sets S of Horn clauses.*

A set of constrained Horn clauses S is *closed under $\mathcal{H}$ with equality and ordering constraint inheritance* if $D \mid T_1 \wedge \ldots \wedge T_n \wedge s = t \wedge OC$ is in S whenever $C_1 \mid T_1\ ,\ \ \ldots,\ \ C_n \mid T_n$ are clauses in S and there is an inference by $\mathcal{H}$ with premises $C_1, \ldots, C_n$ and conclusion $D \mid s = t \wedge OC$ such that the constraint $T_1 \wedge \ldots \wedge T_n \wedge s = t \wedge OC$ is satisfiable.

5.8. THEOREM. *The inference system $\mathcal{H}$ is refutation complete with equality and ordering constraint inheritance for well-constrained sets S of Horn clauses.*

5.3. Basic paramodulation

It is possible to further restrict the inference system $\mathcal{H}$ with constraint inheritance, at the expense of weakening the ordering restrictions. Roughly, the improvement comes from the possibility of moving the inserted right hand side (denoted by r in our superposition rules) in conclusions to the constraint part, thus *blocking* this term for further inferences. On the other hand, paramodulations take place only *with* the maximal term, like in superposition, but *on* both sides of the equation containing the maximal term. More precisely, the inference rule of (ordered, basic) paramodulation right then becomes:

ordered paramodulation right:

$$\frac{\Gamma' \rightarrow l \simeq r \qquad \Gamma \rightarrow s \simeq t}{\Gamma', \Gamma \rightarrow s[x]_p \simeq t \ \ \mid x = r \ \wedge \ s|_p = l \ \wedge \ l > r \ \wedge \ l > \Gamma' \ \wedge \ (s > \Gamma \vee t > \Gamma)}$$

where $s|_p$ is not a variable, and x is a new variable. The inference rule for paramodulation left is defined analogously. It is clear that these inference rules are an improvement upon superposition only when applied (at least) with inheritance of the part $x = r$ of the equality constraint, since otherwise the advantage of blocking r is lost.

The completeness proof is an easy extension of the previous results by the model generation method. It suffices to modify the rule generation by requiring, when a rule $l \Rightarrow r$ is generated, that both l and r are irreducible by R_C, instead of only l as before, and to adapt the proof of Theorem 5.6 accordingly, which is straightforward. We refer to [Bachmair, Ganzinger, Lynch and Snyder 1995] for a deeper discussion of this form of basic paramodulation.

5.4. *Saturation for constrained clauses*

In this section the redundancy notions for constrained clauses and inferences are defined. The idea is similar to how it was done for unconstrained clauses with variables, except that here, as in the proofs of refutation completeness of $\mathcal{H}$ with constraint inheritance, the ground instances are replaced by the set of irreducible (w.r.t. some R) ground instances. These definitions are of a rather theoretical nature. Practical sufficient conditions for them are given in [Nieuwenhuis and Rubio 1995].

In the following, $C \mid T$ and $D \mid T$ (or sometimes simply C, D) denote constrained clauses, S denotes a set of constrained clauses, and R denotes sets of ground rewrite rules included in $\succ$.

First, to get some intuition, let us look at an example showing that the usual simplification techniques are not compatible with constraint inheritance, even if the initial set has no constraints at all. For simplicity, we consider here only equality constraint inheritance, and a simplification step in which $f(g(a))$ is simplified into $f(b)$ by demodulation with the instance $g(a) \simeq b$ of $g(x) \simeq b \mid x = a$. Note that this is the natural extension of the standard method of simplification by rewriting with unconstrained equations, which, as we have seen, does not lead to incompleteness when no constraints are inherited.

5.9. EXAMPLE. Consider an LPO with $f \succ_{\mathcal{F}} g \succ_{\mathcal{F}} a \succ_{\mathcal{F}} b$ and the inconsistent set of four initial clauses:

1. $\rightarrow a \simeq b$
2. $\rightarrow f(g(x)) \simeq g(x)$
3. $\rightarrow f(g(a)) \simeq b$
4. $g(b) \simeq b \rightarrow$

Now we could generate a saturation process as follows:

5.	$\rightarrow g(x) \simeq b$	$\mid x = a$	(by superposition of 3 on 2)
6.	$\rightarrow f(b) \simeq g(x)$	$\mid x = a$	(by superposition of 5 on 2)
$3'$.	$\rightarrow f(b) \simeq b$		(simplifying 3 by 5)

Finally, the set $\{1, 2, 3', 4, 5, 6\}$ is closed under the inference rules, but the empty clause has not been generated. □

Note that the problem is caused by the fact that, although the initial set is well-constrained, after applying the simplification step well-constrainedness is lost, and, as a consequence, refutation completeness. Therefore the redundancy methods should consider only irreducible instances, in order to be consistent with the techniques applied in the previous sections for constraint inheritance.

We denote by $irred_R(\pi)$ the set of ground instances $\pi\sigma$ of an inference π with constraint inheritance such that $C\sigma \in irred_R(C \mid T)$ for each $C \mid T$ that is premise or conclusion of π.

Then, an inference π is redundant in S if for every R compatible with $\succ$ and for every $\pi\sigma \in irred_R(\pi)$ with premises $C_1, \ldots, C_n$, maximal premise C and conclusion D, either $R \cup irred_R(S)^{\prec C_i} \models C_i$ for some $i \in \{1 \ldots n\}$ or $R \cup irred_R(S)^{\prec C} \models D$.

Similarly, a constrained clause $C \mid T$ is redundant in S if, for every R compatible with $\succ$, $R \cup irred_R(S)^{\preceq C\sigma} \models C\sigma$ for every $C\sigma \in irred_R(C \mid T)$. Note that non-strict redundancy of clauses (see Section 4.4 for details) is considered, which is crucial for showing that some powerful simplification methods based on constraints fit in this framework.

It is not difficult to see that in redundancy proofs one can use equations with tautologic constraints or constrained equations like $f(x) \simeq x \mid a > x$ for simplification by rewriting. But by means of constraints one can go beyond:

5.10. EXAMPLE. Let $\succ$ be the lexicographic path ordering where $f \succ_{\mathcal{F}} a \succ_{\mathcal{F}} b$, and consider the set of equations

$$1.\ f(x) \simeq a$$
$$2.\ f(b) \simeq b$$

By analyzing its possible ground instances, equation 1 can be *split* into the one where x is b and the remaining instances. In the former case, 1 can be simplified with

2, and in the latter case the constraint $x \neq b$ can be added, obtaining, respectively, equations 3 and 4:

$$\begin{array}{ll} 2. & f(b) \simeq b \\ 3. & \quad b \simeq a \\ 4. & f(x) \simeq a \mid x \neq b \end{array}$$

By simplifying 4 with 3 we obtain

$$\begin{array}{ll} 2. & f(b) \simeq b \\ 3. & \quad b \simeq a \\ 5. & f(x) \simeq b \mid x \neq b \end{array}$$

Finally, 2 and 5 can be removed by adding 6

$$\begin{array}{ll} 3. & \quad b \simeq a \\ 6. & f(x) \simeq b \end{array}$$

□

Related techniques are applied for paramodulation without any ordering restrictions (plus a certain kind of inferences inside constraints) in [Bourely, Caferra and Peltier 1994].

We can now state refutation completeness, which is proved by combining the techniques of Theorems 5.6 and 4.8.

5.11. THEOREM. *Let S be a well-constrained set of clauses that is saturated w.r.t. $\mathcal{I}$ with constraint inheritance. Then S is satisfiable if, and only if, $\square \notin S$.*

Instead of going into the details of derivations and fairness for constrained clauses, let us only remark here that the methodology explained for clauses without constraints in Section 4 produces results analogous to the ones of Theorem 4.9 for well-constrained sets of clauses.

5.5. General constrained clauses

When considering constraint inheritance for general clauses, the main proof method is the same as before. However, some additional details have to be handled, which make it altogether quite technical. For extending Theorem 5.6, the problems are caused by one case in the proof that has to be considered more carefully: the case where an instance $C\sigma$ of a constrained clause $C \mid T$ of the form $\Gamma \rightarrow x \simeq r, x \simeq r', \Delta \mid T$ generates a rule $x\sigma \Rightarrow r\sigma$ of R. Then, although $C\sigma$ is an instance with a substitution σ that is irreducible with respect to $R_{C\sigma}$, it is reducible with respect to R, since $x\sigma$ is reducible by the rule $x\sigma \Rightarrow r\sigma$ itself.

This situation has the following unpleasant consequences. If an inference with $C\sigma$ on another irreducible instance $C'\sigma$ is needed, it cannot be ensured any more that the corresponding instance $D\sigma$ of the conclusion $D \mid T''$ obtained from $C' \mid T'$ and $C \mid T$ is an irreducible instance, since D has $x \simeq r'$ in the succedent. Note that in the Horn case this cannot happen: if $x\sigma$ is the left hand side of the rule, then x cannot occur in the corresponding conclusion.

The problem is solved by refining the notion of irreducibility for these special variables occurring in an instance $C\sigma$. The problematic variables of $C\sigma$ are those variables that occur in the succedent and only in equations like $x \simeq r$ with $x\sigma \succ r\sigma$. For these variables $x\sigma$ is only required to be irreducible by rules smaller than the greatest $x\sigma \simeq r\sigma$ in $C\sigma$. With this notion of irreducibility, the proof of theorem 5.6 can be applied to general clauses, using the inference system $\mathcal{I}$ and its corresponding rule generation as in Theorem 3.10. We refer to [Nieuwenhuis and Rubio 1995] for details.

5.12. THEOREM. *The inference system* $\mathcal{I}$ *is refutation complete with equality and/or ordering constraint inheritance for well-constrained sets* S *of clauses.*

6. Paramodulation with built-in equational theories

In this section the paramodulation calculus is generalised to the case in which some of the initial axioms are considered as a built-in theory. In particular, the case in which the theory is expressed by a set E of equational axioms is considered.

There are different ways to extend paramodulation based inference systems for this purpose. The simplest one is by adding a new inference rule applying paramodulations *on* the equations of the theory (but not *with* them). An alternative to this inference rule is to associate to each clause the set of its *E-extended clauses* [Peterson and Stickel 1981], which are clauses obtained by adding to the maximal equation (if it is in the succedent) contexts coming from the equations in E. Then paramodulation is performed with the E-extended clauses as well. Due to the fact that many of these extended clauses can be shown to be redundant, this method seems to be less prolific than the first one.

In some interesting cases, like for abelian semigroups, that is, associative and commutative (AC) theories, the useful extended clauses can be easily characterized. Then it becomes possible as well to design specific inference rules instead of handling these extensions explicitly. This is the way most paramodulation calculi for the AC-case are expressed [Paul 1992, Rusinowitch and Vigneron 1995, Vigneron 1994, Nieuwenhuis and Rubio 1997] and in Section 6.2 (see also [Rubio 1996] for built-in semigroups, i.e. associative theories). This approach is considered as well for arbitrary *regular* theories in [Vigneron 1996].

Recent research concerns algebraic structures richer than abelian semigroups, like abelian groups [Stuber 1998*a*, Godoy and Nieuwenhuis 2000], cancellative abelian monoids [Ganzinger and Waldmann 1996], commutative rings [Stuber 1998*b*] or divisible torsion-free abelian groups [Waldmann 1998].

6.1. E-compatible reduction orderings

Many results in the literature on ordered paramodulation and superposition modulo E require (i) E-unification to be finitary, (ii) E-unifiability to be decidable, and (iii) the existence of an E-compatible total reduction ordering. In Section 6.3 we will describe a uniform framework in which the first two requirements turn out to be unnecessary by inheriting equality constraints. Only recently, in [Bofill, Godoy, Nieuwenhuis and Rubio 1999] it was proved that the third requirement can be dropped as well: E-compatible total reduction orderings, which were crucial in all previously existing completeness proofs completeness of ordered paramodulation calculi, are not needed for ordered paramodulation. The new results for paramodulation with non-monotonic orderings of [Bofill et al. 1999] may allow one to work with much simpler orderings. For example, in many cases one can normalize terms w.r.t. the theory E before comparing them by some total ordering on ground terms, thus obtaining a total, E-compatible, and well-founded ordering (that is not monotonic in general). However, the results for non-monotonic orderings have not been developed yet for working modulo equational theories, they are applicable only for ordered paramodulation and not for superposition, and, moreover, they are not compatible with the usual redundancy elimination techniques. Hence it seems reasonable to expect that they will be used only in contexts where E-compatible total reduction orderings do not (or are not known to) exist.

Hence it is necessary to explore the possibilities of finding E-compatible reduction orderings for different theories E and to study in which cases these orderings can be E-total, i.e. total on the E-congruence classes. This is done in the remainder of this section. The following abbreviations will be used for equational axioms: C (commutativity), A (associativity), U (unit), I (idempotence) and D (distributivity).

First we will present some positive results for theories containing associativity and/or commutativity axioms. The easiest case is C, since RPO (see section 2.2) is a C-compatible reduction ordering if all commutative function symbols have a multiset status, and it is C-total under a total precedence if a lexicographic status is assigned to all other symbols. Similarly, in fact any *permutative* theory can be considered, that is, any theory E presented by axioms of the form $f(x_1,\ldots,x_n) \simeq f(x_{\pi(1)},\ldots,x_{\pi(n)})$, where the $x_1,\ldots,x_n$ are distinct variables and π is some permutation of $1\ldots n$. If such f have multiset status, the ordering will be E-compatible. With a little more effort, it can be made total up to $=_E$ by a lexicographic combination with a second component[9].

AC-axioms are present in many interesting theories, and hence AC-compatible orderings have received much more attention than any others. It turned out to

[9]In this second component, roughly, the multisets formed by the equivalence classes of permuting arguments are compared lexicographically. For example, if $a \succ_{\mathcal{F}} b$ and E consists of $f(x_1,x_2,x_3,x_4,x_5) \simeq f(x_1,x_3,x_2,x_4,x_5)$ and $f(x_1,x_2,x_3,x_4,x_5) \simeq f(x_1,x_2,x_3,x_5,x_4)$, then we can compare lexicographically sequences of multisets of arguments $\langle\{1st\},\{2nd,3rd\}.\{4th,5th\}\rangle$, and $f(a,a,a,b,a) \succ f(a,a,b,a,a)$, since the multiset $\{a,a\}$ of the second and third argument of the first term is larger than the one of the second term, which is $\{b,a\}$.

be quite difficult to find AC-compatible reduction orderings, especially when AC-totality is also required. In fact, the first attempts were not total in general (see e.g. [Bachmair and Plaisted 1985, Ben-Cherifa and Lescanne 1987, Kapur, Sivakumar and Zhang 1990]). The first AC-compatible AC-total reduction ordering was exhibited in [Narendran and Rusinowitch 1991], while the first such ordering based on RPO appeared in [Rubio and Nieuwenhuis 1995] and further improvements on AC-orderings with RPO-scheme are developed in [Kapur and Sivakumar 1997, Rubio 1999]. For the A (associativity only) case, not many results have been developed. Of course, if A-totality is not required, any of the AC-orderings can be used. Otherwise, in [Rubio 1996], a way to obtain A-compatible A-total reduction orderings from AC-orderings is described. However, apparently some of the known RPO-like AC-total orderings could also be adapted to the A case directly. Finally, joining all the results one can obtain E-compatible E-total reduction orderings for theories E containing A-, C-, AC- and free symbols [Rubio 1994].

When considering other theories, fewer positive results can be obtained. U-compatible orderings cannot fulfill the subterm property, since if $+$ is a function symbol with unit 0 then $x + 0 =_U x$ and hence, by U-compatibility, $x + 0$ cannot be greater than x. This means that E-compatible *simplification* orderings do not exist if there are any such unit axioms in E. However, it is possible, as described in [Jouannaud and Marché 1992] and [Wertz 1992], to obtain an ACU-compatible *reduction* ordering from an AC-compatible reduction ordering, provided that all units are minimal in the given AC-ordering. But such a restriction has to be required by any E-compatible ordering such that E contains any unit axioms, i.e. $U \subseteq E$:

6.1. EXAMPLE. Let $+$ and $*$ be U-function symbols with units 0 and 1 respectively. Then if $0 \succ_E 1$, by monotonicity, $x + 0 \succ_E x + 1$, which implies, by E-compatibility (since $x+0 =_U x$), $x \succ_E x+1$, contradicting, by monotonicity, the well-foundedness of $\succ_E$. The symmetric case $1 \succ_E 0$ leads to the same contradiction. □

This example shows us that only one unit is allowed if we are interested in E-totality. There may exist U-compatible reduction orderings that are U-total but which are not simplification orderings.

The case in which E contains some idempotence axiom is even worse, since then no E-compatible well-founded ordering $\succ_E$ can be monotonic:

6.2. EXAMPLE. Let $*$ be an I-function symbol and let s and t be terms with $s \succ_E t$. Then if $\succ_E$ is monotonic we have $s * s \succ_E t * s$ and hence, by E-compatibility (since $s*s =_I s$), $s \succ_E t*s$, which together with monotonicity contradicts well-foundedness. □

Finally another interesting example is the presence of distributivity axioms. It is unknown whether there are, in general, D-compatible reduction orderings (and also E-compatible for a set E containing distributivity axioms). However, a well-known ACD-compatible ordering is the APO [Bachmair and Plaisted 1985], when there are no distribution *chains*.

6.2. Paramodulation modulo associativity and commutativity

Let us now consider the case of superposition with built-in associativity and commutativity for some function symbols.

In this section constraints are interpreted as follows: the ordering $\succ$ interpreting $>$ is now an AC-compatible AC-total reduction ordering, while $=$ is interpreted as $=_{AC}$ (the AC-equality congruence).

The full inference system for general clauses modulo AC, called $\mathcal{I}_{AC}$ includes the rules of $\mathcal{I}$ (under the new semantics for $>$ and $=$) plus the following three specific rules:

AC-superposition right:

$$\frac{\Gamma' \rightarrow l \simeq r, \Delta' \qquad\qquad \Gamma \rightarrow s \simeq t, \Delta}{\Gamma', \Gamma \rightarrow s[f(r,x)]_p \simeq t, \Delta', \Delta \quad \left|\quad \begin{array}{l} s|_p = f(l,x) \ \wedge \\ l > r \ \wedge \ l > \Gamma' \ \wedge \ gr(l \simeq r, \Delta') \ \wedge \\ s > t \ \wedge \ s > \Gamma \ \wedge \ gr(s \simeq t, \Delta) \end{array}\right.}$$

AC-superposition left:

$$\frac{\Gamma' \rightarrow l \simeq r, \Delta' \qquad\qquad \Gamma, s \simeq t \rightarrow \Delta}{\Gamma', \Gamma, s[f(r,x)]_p \simeq t \rightarrow \Delta', \Delta \quad \left|\quad \begin{array}{l} s|_p = f(l,x) \ \wedge \\ l > r \ \wedge \ l > \Gamma' \ \wedge \ gr(l \simeq r, \Delta') \ \wedge \\ s > t \ \wedge \ greq(s \simeq t, \Gamma \cup \Delta) \end{array}\right.}$$

AC-top-superposition:

$$\frac{\Gamma' \rightarrow l \simeq r, \Delta' \qquad\qquad \Gamma \rightarrow s \simeq t, \Delta}{\Gamma', \Gamma \rightarrow f(r',x') \simeq f(r,x), \Delta', \Delta \quad \left|\quad \begin{array}{l} f(l',x') = f(l,x) \ \wedge \\ l > r \ \wedge \ l > \Gamma' \ \wedge \ gr(l \simeq r, \Delta') \ \wedge \\ s > t \ \wedge \ s > \Gamma \ \wedge \ gr(s \simeq t, \Delta) \end{array}\right.}$$

In these rules, where x and x' are new variables, the term l can be restricted to be headed by the AC-symbol f. This can be expressed in the constraint language and added to the constraint. Let us define $u|_q$ to be a *maximal non-f subterm* of u if q is a position such that $top(u|_q) \neq f$ and $top(u|_{q'}) = f$ for all proper prefixes q' of q. Then, AC-top-superposition is only needed if l and l' are headed by f and share some maximal non-f subterm s but x and x' do not (some restrictions implied by this condition can be formulated in the constraint language, and hence some cases of failure of this condition can be detected as unsatisfiable constraints). Finally, the superposition inferences are needed only if $l|_p$ is non-variable (as usual), and AC-superposition is needed only if moreover $l|_p$ (which has an f as top symbol) is not immediately below another f. Some examples are given below.

The refutation completeness of $\mathcal{I}_{AC}$ can be proved by introducing a notion of *extended instance* of a clause and then adapting the construction of the rewrite system R to work modulo AC and considering these extended instances for the generation of rules. We refer to [Nieuwenhuis and Rubio 1997] for the details.

6.3. THEOREM. *The inference system $\mathcal{I}_{AC}$ is refutation complete without constraint inheritance with built-in AC-theories.*

6.3. Constraint inheritance and built-in theories

By inheriting equality constraints one can avoid one of the main drawbacks of working with built-in theories, namely the large cardinality of the set of unifiers for certain unification problems. For instance, there may be doubly exponentially many AC-unifiers for two terms [Domenjoud 1992] (in a sense, this is also an upper bound [Kapur and Narendran 1992]), and therefore as many conclusions in an inference; e.g., a minimal complete set for $x + x + x$ and $y_1 + y_2 + y_3 + y_4$ contains more than a million unifiers.

For proving refutation completeness with constraint inheritance it becomes necessary, as in the free case (see section 5.1), to consider irreducible instances. In this case the irreducibility notion for the fresh variables introduced by the three specific AC-inference rules needs to be refined. Again we refer to [Nieuwenhuis and Rubio 1997] for the details.

6.4. THEOREM. *The inference system $\mathcal{I}_{AC}$ is refutation complete with constraint inheritance for well-constrained sets S of clauses with built-in AC-theories.*

As said, by inheriting equality constraints, the computation of unifiers and the generation of many conclusions in every inference becomes unnecessary. But it is possible to go beyond. One can deal, in an effective way, with theories E with an *infinitary* E-unification problem, i.e., theories where for some unification problems any complete set of unifiers is infinite. This is the case for theories containing only associativity axioms for some function symbols, which is developed in [Rubio 1996].

Finally, one can consider any built-in theory E, even when the E-unification problem is undecidable, if an adequate inference system and ordering are found (although these ideas require a further development for concrete E). Incomplete methods can then be used for detecting cases of unsatisfiable constraints, and only when a constrained empty clause $\square \mid T$ is found, one has to start (in parallel) a semidecision procedure proving the satisfiability of T. But note that for soundness only the equality constraint part of T needs to be proved satisfiable, since the inference rules are sound as well without ordering restrictions. This method is not only interesting if no decision procedure for the E-unification problem is available: incomplete methods can be more efficient and hence more effective in practice than complete ones.

7. Symbolic constraint solving

Equality constraints are also known as *unification problems*, since they generalize the notion of unification, which usually consists in solving one single equation. Due to the large amount of applications of unification in automated deduction, logic programming and, in general, in symbolic computation, equational constraints have been used in many different applications. Hence for this topic here we refer to Chapter [chapter with id unification] of this handbook for a detailed survey.

7.1. Ordering constraint solving

Apart from the applications to pruning the search space in automated theorem proving, ordering constraint solving is useful in many other contexts like proving termination of term rewrite systems or confluence of ordered term rewrite systems [Comon et al. 1998].

Regarding the former application, constraints provide powerful termination orderings $\succ_c$ for term rewriting, defined: $s \succ_c t$ if $s\sigma \succ t\sigma$ for all ground σ. If $\succ$ is the recursive path ordering (RPO), such $\succ_c$ subsume other path orderings like the ones of [Kapur, Narendran and Sivakumar 1985, Lescanne 1990] since all these path orderings coincide on ground terms (see [Dershowitz 1987]). For example, if s is $g(f(x), f(y))$ and t is $g(g(x,y), g(x,y))$, and $f \succ_{\mathcal{F}} g$ in the precedence, then $s \not\succ_{rpo} t$, but $s \succ_c t$. Ordering constraint solving is also applicable as an underlying technique in more general contexts like the *dependency pairs* method of [Arts and Giesl 1997].

As explained in Section 4.7 of this chapter, some applications of ordering constraints to ordered strategies in theorem proving gave rise to the distinction between fixed signature semantics (solutions are built over a given signature $\mathcal{F}$) and extended signature semantics (new symbols are allowed to appear in solutions) [Nieuwenhuis and Rubio 1992*b*].

The satisfiability problem for ordering constraints was first shown decidable for fixed signatures when $\succ$ is a total LPO [Comon 1990] or a total RPO [Jouannaud and Okada 1991]. For extended signatures, decidability was shown for LPO in [Nieuwenhuis and Rubio 1995] and for RPO in [Nieuwenhuis 1993]. Regarding complexity, NP algorithms for LPO (fixed and extended signatures) and RPO (extended ones) were given in [Nieuwenhuis 1993]. Recently, an NP algorithm has been given as well for RPO under fixed signatures in [Narendran, Rusinowitch and Verma 1998]. For the AC-RPO ordering of [Rubio and Nieuwenhuis 1995], decidability was shown in [Comon, Nieuwenhuis and Rubio 1995]. NP-hardness of the satisfiability problem is known, even for one single inequation, for all these cases [Comon and Treinen 1994].

All these decision procedures use at some point the fact that a constraint C can be effectively expressed as an equivalent disjunction of expressions $s_1 > t_1 \wedge \ldots \wedge s_n > t_n$, called *solved forms,* where for each i always at least one of s_i or t_i is a variable. Solved forms are obtained by repeatedly applying the definition of the ordering by rules like:

$$f(s_1, \ldots, s_p) > t \implies s_1 \geq t \vee \ldots \vee s_p \geq t$$

if t is a non-variable term whose topmost symbol is bigger in the precedence than f. Similar rules deal with the equality relations in the constraints, like $f(\ldots) = g(\ldots) \implies \perp$ if $f \neq g$, and $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n) \implies s_1 = t_1, \wedge \ldots \wedge, s_n = t_n$.

Due to the transformations into disjunctive normal form, the number of solved forms for a given C may be exponential, even if all atoms in C are already inequalities between variables or if C consists of one single inequality. In algorithms like the ones of [Comon 1990] and [Nieuwenhuis and Rubio 1995], the computation of solved forms is only a first step that is followed by other exponential phases. This is not surprising, since this notion of solved form only involves a local analysis of the inequations considered independently. In fact any constraint $s > t$ can be expressed as the solved form $s > x \wedge x > t$, for some new variable x, which is equivalent w.r.t. satisfiability under extended signatures.

On the other hand, the NP algorithms of [Nieuwenhuis 1993] and [Narendran et al. 1998] are based on a first non-deterministic guess of a *simple system* for C, a particular constraint S of the form $s_n \ \#_n \ s_{n-1} \ \#_{n-1} \ \ldots \ \#_1 \ s_0$, where each $\#_i$ is either $=$ or $>$, and $\{s_n, \ldots, s_1\}$ is the set of *all subterms* of C. Then, roughly, C is satisfiable if, and only if, some of its simple systems contains one of its own solved forms and entails C. For each simple system, this can be checked in polynomial time, but the number of simple systems to be considered is far too large for practical usefulness.

A new family of algorithms, for full RPO and both semantics, has been introduced recently in [Nieuwenhuis and Rivero 1999]. These algorithms are based on a new notion of solved form, where properties of orderings like transitivity and monotonicity are taken into account. They are simple and, since guessing is minimised, more efficient.

8. Extensions

8.1. Paramodulation-based answer computation

Answer computation in some logic is at the heart of many applications of automated reasoning. Well-known simple examples of such mechanisms are Prolog's SLD-resolution, where the accumulated unifiers are kept as answers, or E-unification procedures for equational (or more general) theories E in which, given a goal $s = t$, answers σ are computed such that $E \models s\sigma = t\sigma$.

In [Gallier and Snyder 1989] general rules for E-unification are given. *Narrowing* was originally devised as an efficient E-unification procedure using a convergent (confluent and terminating) set of rewrite rules R for E [Fay 1979, Hullot 1980*b*]. Many extensions (to, e.g., conditional TRS's) and optimizations of narrowing have been proposed (see e.g. [Rety, Kirchner and Lescanne 1985, Bosco, Giovanetti and Moiso 1988, Hölldobler 1989, Nutt, Réty and Smolka 1989, Bockmayr, Krischer and

Werner 1992, Bockmayr and Werner 1994]). Most completeness proofs of these narrowing strategies are based on lifting arguments applied to rewrite proofs, which has limitations when applied to unrestricted general clauses, more general simplification and redundancy notions, or with constrained clauses.

The techniques developed in this chapter can be extended into an alternative approach, based on the well-known fact that in refutation theorem proving, each refutation proof provides *one* answer, like in SLD-resolution. This has been done in [Nieuwenhuis 1995], where a proof technique is developed that uniformly covers E-unification-like methods and Prolog-like resolution strategies. By narrowing modulo equational theories like AC, compact representations of solutions, expressed by AC-equality constraints, can be obtained. Computing AC-unifiers is only needed at the end if one wants to "uncompress" such a constraint into its (doubly exponentially many) concrete substitutions. In [Lynch 1997] it is shown that superposition is complete for answer computation with arbitrary selection rules (where also positive literals can be selected), thus properly extending SLD resolution to clauses with equality literals.

8.2. Paramodulation-based decidability and complexity results

The well-known close relationship between computation formalisms and deduction in some logic has been a starting point for a considerable amount of recent research in logic-based decidability and complexity analysis.

Regarding resolution-based results, for example in [Basin and Ganzinger 1996] saturatedness of sets S of clauses (without equality) with respect to different orderings implies membership in different complexity classes of the entailment problems $S \models C$ for ground C. And of course for the Datalog language of *flat* Horn clauses without equality there are a number of results from descriptive complexity theory; for example, that Datalog programs precisely capture the set of queries on a finite database decidable in finite time [Vardi 1982, Immerman 1986].

Regarding paramodulation-based decidability results, for the class of ground equations where some symbols are associative and commutative (AC) a finite confluent rewrite system can always be computed by superposition [Narendran and Rusinowitch 1991, Marché 1991], by which the word problem is decidable. In the same class, the unification problem is also decidable [Narendran and Rusinowitch 1993]. In [Marché 1996] paramodulation-based decidability results for word problems in ground presentations modulo several other theories extending AC are given, like abelian groups or commutative rings.

Concerning non-ground theories, superposition with simplification can be used as a decision procedure for the monadic class with equality [Bachmair, Ganzinger and Waldmann 1993*b*] (which is equivalent to a class of set constraints [Bachmair, Ganzinger and Waldmann 1993*a*]). Similar very recent results have been obtained for the guarded fragment [Ganzinger, Meyer and Veanes 1999, Ganzinger and de Nivelle 1999].

In [Waldmann 1999] it is shown that cancellative superposition decides the theory of divisible torsion-free abelian groups. The equational *shallow theories*, the ones axiomatized by equations where no variable occurs at depth more than one, are another fundamental class with decidable word and unification problems and even a decidable first-order theory [Comon, Haberstrau and Jouannaud 1994]. In [Nieuwenhuis 1998] it is shown that for sets of Horn clauses with equality saturated under basic paramodulation, the word and unifiability problems are in NP, and the number of minimal unifiers is simply exponential; this can be applied to shallow Horn clauses with equality. For certain Horn sets S saturated under basic superposition, the word and unifiability problems are still decidable and unification is finitary. Further results on the decidability of unification problems in Horn theories have been obtained by sorted superposition [Jacquemard, Meyer and Weidenbach 1998].

9. Perspectives

In this section some prominent research problems and future directions for research in this area are addressed.

Apart from adequate theoretical results as we have seen them in this chapter, building a state-of-the-art paramodulation-based theorem prover requires at least two more ingredients: good heuristics, and the necessary engineering skills to implement it all in an efficient way. Progress between theory and these other aspects is interacting in different ways.

On the one hand, new theoretical insights are replacing heuristics by more precise and effective techniques. For example, the completeness proof of basic paramodulation shows why no inferences below Skolem functions are needed, as conjectured by McCune [1990]. Regarding implementation techniques, ad-hoc algorithms for procedures like demodulation or subsumption are being replaced by efficient, re-usable, general-purpose indexing data structures for which the time and space requirements are well-known (see Chapter [chapter with id indexing] of this handbook).

But, on the other hand, theory is also advancing 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 applied by McCune for finding his proof of the Robbins conjecture [McCune 1997*b*] by basic paramodulation modulo associativity and commutativity (AC).

Provers like Spass [Weidenbach 1997], 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 (see below).

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. Paramodulation with constraints seems to be an adequate paradigm for doing this in a clean way. It uses specialized 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.

In the remainder of this section, some of the current theoretical and practical challenges concerning the construction of paramodulation-based provers are surveyed.

9.1. *Basicness and redundancy*

The basic restriction in paramodulation is easy to implement in most provers by marking *blocked* subterms, i.e., the point where the constraint starts. However, we have seen that full simplification by demodulation is incomplete in combination with the basic strategy. An important challenge is to develop adequate redundancy notions for the basic strategy. Although some ideas are given in [Lynch and Scharff 1998], better results are needed for practice.

In the context of E-paramodulation with constraints, another interesting problem is how to apply a constrained equation $s \simeq t \mid T$ in a demodulation step without solving the E-unification problem in T. 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.

9.2. *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. It is not clear at all 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.

9.3. *Constraint solving*

As we have seen, for taking advantage of the constraints, algorithms for constraint satisfiability checking are required. Deciding the problem in general requires exponential time for path orderings like LPO or RPO. Is there any useful ordering for which deciding the satisfiability of (e.g., only conjunctive) constraints can be done in polynomial time? Or, if the answer is negative, for which orderings can we have better practical algorithms?

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?

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 [chapter with id unification] of this handbook). Are there any sound tests detecting most cases of unsatisfiability?

9.4. Indexing data structures

For many standard operations like many-to-one matching or unification indexing data structures exist that can be used in operations like inference computation, demodulation or subsumption (see Chapter [chapter with id indexing] of this handbook). Such data structures are crucial in order to obtain a prover whose throughput remains stable while the number of clauses increases.

But for many operations no indexing data structures have been developed yet. For example, consider demodulation with equations that cannot be oriented *a priori* w.r.t. the ordering $\succ$, like the commutativity axiom. If such an equation $s \simeq t$ is found to be applicable to a term $s\sigma$, then after matching it has to be checked 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.

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?

9.5. More powerful redundancy notions

In the *Saturate* system [Nivela and Nieuwenhuis 1993, Ganzinger, Nieuwenhuis and Nivela 1999], a number of experiments with non-standard redundancy notions has been carried out. For example, *constrained rewriting* turns out to be powerful enough for deciding the confluence of ordered rewrite systems [Comon et al. 1998]. 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 the experiments it is clear that such techniques importantly reduce the number of retained clauses.

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.

9.6. More global future research directions

Up to this point, in this section we have focussed on concrete problems concerning the application of the theory explained in this chapter in actual provers. Longer-term challenges include the following two global areas.

One main area of interest concerns the integration of prover components: how to integrate dedicated procedures within general-purpose paramodulation-based provers (along the lines of Section 6), and how to integrate automated paramodulation-based provers in more general environments like proof assistants. Similarly, it has to be studied how to combine paramodulation-based provers with other automated reasoning paradigms.

A second major area of interest involves the application of the theory of paramodulation to other subfields of computer science like programming and complexity theory (along the lines of the results described in Sections 8.1 and 8.2), as well as more concrete applications like the analysis of security protocols [Weidenbach 1999].

Acknowledgments

We wish to thank the many people who helped us to improve (rewrite, polish, extend) this chapter, in particular Anatoli Degtyarev, Guillem Godoy, Jieh Hsiang, Chris Lynch, Michael Rusinowitch, and especially Andrei Voronkov.

Bibliography

Arts T. and Giesl J. [1997], Automatically proving termination where simplification orderings fail, *in* 'TAPSOFT: 7th International Joint Conference on Theory and Practice of Software Development', LNCS 1214, Springer-Verlag, pp. 261–272.

Bachmair L. [1989], Proof normalization for resolution and paramodulation, *in* 'Third int. conf. on Rewriting Techniques and Applications (RTA)', LNCS 355, Springer-Verlag, Chapel Hill, NC, USA, pp. 15–28.

Bachmair L. [1991], *Canonical equational proofs*, Birkhäuser, Boston, Mass.

Bachmair L. and Dershowitz N. [1989], 'Completion for rewriting modulo a congruence', *Theoretical Computer Science* **2 and 3**(67), 173–201.

Bachmair L. and Dershowitz N. [1994], 'Equational inference, canonical proofs, and proof orderings', *J. of the Association for Computing Machinery* **41**(2), 236–276.

BACHMAIR L., DERSHOWITZ N. AND HSIANG J. [1986], Orderings for equational proofs, *in* 'First IEEE Symposium on Logic in Computer Science (LICS)', IEEE Computer Society Press, Cambridge, Massachusetts, USA, pp. 346–357.

BACHMAIR L., DERSHOWITZ N. AND PLAISTED D. [1989], Completion Whitout Failure, *in* H. Aït-Kaci and M. Nivat, eds, 'Resolution of Equations in Algebraic Structures', Vol. 2: Rewriting Techniques, Academic Press, New York, chapter 1, pp. 1–30.

BACHMAIR L. AND GANZINGER H. [1990], On restrictions of ordered paramodulation with simplification, *in* M. E. Stickel, ed., '10th International Conference on Automated Deduction (CADE)', LNAI 449, Springer-Verlag, Kaiserslautern, FRG, pp. 427–441.

BACHMAIR L. AND GANZINGER H. [1991], Perfect model semantics for logic programs with equality, *in* K. Furukawa, ed., 'Logic Programming, Proceedings of the Eighth International Conference', The MIT Press, Paris, France, pp. 645–659.

BACHMAIR L. AND GANZINGER H. [1994*a*], Associative-commutative superposition, *in* N. Dershowitz, ed., 'Proc. 5th International Workshop on Conditional Term Rewriting Systems', LNCS 968, Springer-Verlag, Jerusalem, pp. 155–167.

BACHMAIR L. AND GANZINGER H. [1994*b*], 'Rewrite-based equational theorem proving with selection and simplification', *Journal of Logic and Computation* **4**(3), 217–247.

BACHMAIR L., GANZINGER H., LYNCH C. AND SNYDER W. [1992], Basic paramodulation and superposition, *in* D. Kapur, ed., '11th International Conference on Automated Deduction (CADE)', LNAI 607, Springer-Verlag, Saratoga Springs, New York, USA, pp. 462–476.

BACHMAIR L., GANZINGER H., LYNCH C. AND SNYDER W. [1995], 'Basic paramodulation', *Information and Computation* **121**(2), 172–192.

BACHMAIR L., GANZINGER H. AND WALDMANN U. [1993*a*], Set constraints are the monadic class, *in* 'Eighth Annual IEEE Symposium on Logic in Computer Science (LICS)', IEEE Computer Society Press, Montreal, canada, pp. 75–83.

BACHMAIR L., GANZINGER H. AND WALDMANN U. [1993*b*], 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, SpringerVerlag, pp. 83–96.

BACHMAIR L. AND PLAISTED D. A. [1985], 'Termination orderings for associative-commutative rewriting systems', *Journal of Symbolic Computation* **1**, 329–349.

BASIN D. AND GANZINGER H. [1996], Complexity Analysis Based on Ordered Resolution, *in* 'Eleventh Annual IEEE Symposium on Logic in Computer Science (LICS)', IEEE Computer Society Press, New Brunswick, New Jersey, USA, pp. 456–465.

BEN-CHERIFA A. AND LESCANNE P. [1987], 'Termination of rewriting systems by polynomial interpretations and its implementation', *Science of Computer Programming* **9**, 137–160.

BOCKMAYR A., KRISCHER S. AND WERNER A. [1992], An optimal narrowing strategy for general canonical systems, *in* M. Rusinowitch and J.-L. Rémy, eds, 'The Third International Workshop on Conditional Term Rewriting Systems', LNCS 656, Springer-Verlag, Pont-à-Mousson, France.

BOCKMAYR A. AND WERNER A. [1994], LSE narrowing for decreasing conditional term rewrite systems, *in* N. Dershowitz, ed., 'The fourth International Workshop on Conditional Term Rewriting Systems', LNCS 968, Jerusalem, pp. 167–190.

BOFILL M., GODOY G., NIEUWENHUIS R. AND RUBIO A. [1999], Paramodulation with non-monotonic orderings, *in* '14th IEEE Symposium on Logic in Computer Science (LICS)', Trento, Italy, pp. 225–233.

BOSCO P., GIOVANETTI E. AND MOISO C. [1988], 'Narrowing vs. sld-resolution', *Theoretical Computer Science* **2**(59), 3–23.

BOURELY C., CAFERRA R. AND PELTIER N. [1994], A method for building models automatically: Experiments with an extension of OTTER, *in* A. Bundy, ed., 'Proceedings of the 12th International Conference on Automated Deduction', Vol. 814 of *LNAI*, Springer, Berlin, pp. 72–86.

BRAND D. [1975], 'Proving theorems with the modification method', *SIAM Journal on Computing* **4**(4), 412–430.

COMON H. [1990], 'Solving symbolic ordering constraints', *International Journal of Foundations of Computer Science* **1**(4), 387–411.

COMON H., HABERSTRAU M. AND JOUANNAUD J.-P. [1994], 'Syntacticness, Cycle-Syntacticness and Shallow Theories', *Information and Computation* **111**(1), 154–191.

COMON H., NARENDRAN P., NIEUWENHUIS R. AND RUSINOWITCH M. [1998], Decision problems in ordered rewriting, *in* '13th IEEE Symposium on Logic in Computer Science (LICS)', Indianapolis, USA, pp. 410–422.

COMON H. AND NIEUWENHUIS R. [2000], 'Induction = I-axiomatization + first-order consistency', *Information and Computation* . To appear.

COMON H., NIEUWENHUIS R. AND RUBIO A. [1995], Orderings, AC-Theories and Symbolic Constraint Solving, *in* '10th IEEE Symposium on Logic in Computer Science (LICS)', San Diego, USA, pp. 375–385.

COMON H. AND TREINEN R. [1994], Ordering Constraints on Trees, *in* 'Proc. of Colloquium on Trees in Algebra and Programming (CAAP)', LNCS 787, Springer-Verlag, Edinburgh, Scotland, pp. 1–14.

DEGTYAREV A. [1979], The monotonic paramodulation strategy, *in* '5th All-Union Conference on Mathematical Logic', Novosibirsk. (In Russian).

DEGTYAREV A. AND VORONKOV A. [1986], 'Equality methods in machine theorem proving', *Cybernetics* **22**(3), 298–307.

DERSHOWITZ N. [1982], 'Orderings for term-rewriting systems', *Theoretical Computer Science* **17**(3), 279–301.

DERSHOWITZ N. [1987], 'Termination of rewriting', *Journal of Symbolic Computation* **3**, 69–116.

DERSHOWITZ N. [1991], Canonical sets of Horn clauses, *in* J. L. Albert, B. Monien and M. R. Artalejo, eds, 'Proceedings of the Eighteenth International Colloquium on Automata, Languages and Programming (ICALP)', LNCS 510, Springer-Verlag, Madrid, Spain, pp. 267–278.

DERSHOWITZ N. AND MANNA Z. [1979], 'Proving termnation with multiset orderings', *Comm. of ACM* **22**(8).

DOMENJOUD E. [1992], 'A technical note on AC-unification. the number of minimal unifiers of the equation $\alpha x_1 + \ldots + \alpha x_p = \beta y_1 + \ldots + \beta y_q$', *Journal of Automated Reasoning* **8**(1), 39–44.

FAY M. [1979], First-order unification in an equational theory, *in* 'Proceedings of the Fourth Workshop on Automated Deduction', Austin, TX, pp. 161–167.

GALLIER J. H. AND SNYDER W. [1989], 'Complete sets of transformations for general E-unification', *Theoretical Computer Science* **67**(2-3), 203–260.

GANZINGER H. [1991], 'A completion procedure for conditional equations', *Journal of Symbolic Computation* **11**, 51–81.

GANZINGER H. AND DE NIVELLE H. [1999], A superposition decision procedure for the guarded fragment with equality, *in* '14th IEEE Symposium on Logic in Computer Science (LICS)', Trento, Italy, pp. 295–305.

GANZINGER H., MEYER C. AND VEANES M. [1999], The two-variable guarded fragment with transitive relations, *in* '14th IEEE Symposium on Logic in Computer Science (LICS)', Trento, Italy, pp. 24–34.

GANZINGER H., NIEUWENHUIS R. AND NIVELA P. [1999], 'The Saturate System'. Software and documentation available at: `http://www.mpi-sb.mpg.de/SATURATE/Saturate.html`.

GANZINGER H. AND WALDMANN U. [1996], Theorem proving in cancellative abelian monoids, *in* M. A. McRobbie and J. K. Slaney, eds, 'Thirteenth International Conference on Automated Deduction (CADE)', Vol. 1104 of *LNAI*, Springer, Berlin, pp. 388–402.

GODOY G. AND NIEUWENHUIS R. [2000], Paramodulation with built-in abelian groups, *in* '15th IEEE Symposium on Logic in Computer Science (LICS)', Santa Barbara, USA.

HÖLLDOBLER S. [1989], *Foundations of equational logic programming*, LNCS 353, Springer-Verlag.

HSIANG J. AND RUSINOWITCH M. [1987], On word problems in equational theories, *in* T. Ottmann, ed., 'Proc. 14th Int. Colloquium Automata, Languages and Programming', LNCS 267, Springer-Verlag, Berlin, Germany, pp. 54–71.

HSIANG J. AND RUSINOWITCH M. [1991], 'Proving refutational completeness of theorem proving strategies: the transfinite semantic tree method', *Journal of the ACM* **38**(3), 559–587.

HUET G. [1980], 'Confluent reductions: abstract properties and applications to term rewriting systems', *Journal of the ACM* **27**(4), 797–821.

HULLOT J. [1980*a*], Compilation de Formes Canoniques dans les Teories Equationnelles, PhD thesis, Universite de Paris Sud, France.

HULLOT J.-M. [1980*b*], Canonical forms and unification, *in* R. Kowalski, ed., 'Fifth International Conference on Automated Deduction (CADE)', LNCS 87, Springer-Verlag, Les Arcs, France, pp. 318–334.

IMMERMAN N. [1986], 'Relational queries computable in polynomial time', *Information and Control* **68**(1–3), 86–104.

JACQUEMARD F., MEYER C. AND WEIDENBACH C. [1998], Unification in extensions of shallow equational theories, *in* T. Nipkow, ed., '9th International Conference on Rewriting Techniques and Applications (RTA)', LNCS 1379, Springer, Tsukuba, Japan, pp. 76–90.

JOUANNAUD J.-P. [1983], Confluent and coherent equational term rewriting systems: Applications to proofs in abstract data types, *in* 'Proc. 8th Colloquium on Trees in Algebra and Programming', LNCS 59, Springer-Verlag, pp. 269–283.

JOUANNAUD J.-P. AND KIRCHNER H. [1986], 'Completion of a set of rules modulo a set of equations', *SIAM Journal of Computing* **15**, 1155–1194.

JOUANNAUD J.-P. AND MARCHÉ C. [1992], 'Termination and completion modulo associativity, commutativity and identity', *Theoretical Computer Science* **104**, 29–51.

JOUANNAUD J.-P. AND OKADA M. [1991], Satisfiability of systems of ordinal notations with the subterm property is decidable, *in* '18th International Colloquium Automata, Languages and Programming (ICALP)', LNCS 510, Springer-Verlag, Madrid, Spain, pp. 455–468.

JOUANNAUD J.-P. AND WALDMANN B. [1986], Reductive conditional term rewriting systems, *in* 'Proc. Third IFIP Working Conference on Formal Description of Programming Concepts', Ebberup, Denmark.

KAMIN S. AND LEVY J.-J. [1980], Two generalizations of the recursive path ordering. Unpublished note, Dept. of Computer Science, Univ. of Illinois, Urbana, IL.

KAPLAN S. [1984], 'Conditional rewrite rules', *Theoretical Computer Science* **33**, 175–193.

KAPUR D. AND NARENDRAN P. [1992], Double exponential complexity of computing complete sets of AC-unifiers, *in* 'Seventh Annual IEEE Symposium on Logic in Computer Science', IEEE Computer Society Press, Santa Cruz, California, USA, pp. 11–21.

KAPUR D., NARENDRAN P. AND SIVAKUMAR G. [1985], A path ordering for proving termination for term rewriting systems, *in* 'Proc. of 10th Colloquium on Trees in Algebra and Programming', LNCS 185, Springer-Verlag, Germany, pp. 173–185.

KAPUR D. AND SIVAKUMAR G. [1997], A total, ground path ordering for proving termination of ac-rewrite systems, *in* H. Comon, ed., '8th International Conference on Rewriting Techniques and Applications (RTA)', LNCS 1232, Springer-Verlag, Sitges, Spain, pp. 142–156.

KAPUR D., SIVAKUMAR G. AND ZHANG H. [1990], A new method for proving termination of ac-rewrite systems, *in* 'Conf. Found. of Software Technology and Theoretical Computer Science', LNCS 472, Springer-Verlag, New Delhi, India, pp. 134–148.

KIRCHNER C., KIRCHNER H. AND RUSINOWITCH M. [1990], 'Deduction with symbolic constraints', *Revue Française d'Intelligence Artificielle* **4**(3), 9–52.

KNUTH D. AND BENDIX P. [1970], Simple word problems in universal algebras, *in* 'J. Leech, ed., Computational Problems in Abstract Algebra', Pergamon Press, Oxford, pp. 263–297.

KOUNALIS E. AND RUSINOWITCH M. [1991], 'On word problems in Horn theories', *Journal of Symbolic Computation* **11**(1–2), 113–128.

LANKFORD D. S. [1975], Canonical inference, Technical Report ATP-32, Dept. of Mathematics and Computer Science, Univ. of Texas, Austin, TX.

LANKFORD D. S. AND BALLANTYNE A. M. [1977], Decision procedures for simple equational theories with commutative-associative axioms: Complete sets of commutative-associative reductions, Technical Report Memo ATP-39, Dept. of Mathematics and Computer Science, Univ. of Texas, Austin, TX.

LESCANNE P. [1990], 'On the recursive decomposition ordering with lexicographical status and other related orderings', *Journal of Automated Reasoning* **6**(1), 39–49.

LOVELAND D. W. [1978], *Automated Theorem Proving: a Logical Basis*, 1 edn, North-Holland, Amsterdam.

LYNCH C. [1997], 'Oriented equational logic programming is complete', *Journal of Symbolic Computation* **23**(1), 23–46.

LYNCH C. AND SCHARFF C. [1998], Basic completion with E-cycle simplification, *in* J. Calmet and J. Plaza, eds, 'Proceedings of the International Conference on Artificial Intelligence and Symbolic Computation (AISC-98)', LNAI 1476, Springer Verlag, pp. 209–221.

LYNCH C. AND SNYDER W. [1993], Redundancy criteria for constrained completion, *in* C. Kirchner, ed., '5th International Conference on Rewriting Techniques and Applications (RTA)', LNCS 690, Springer-Verlag, Montreal, Canada, pp. 2–16.

MARCHÉ C. [1991], On ground AC-completion, *in* R. V. Book, ed., '4th Int. Conf. Rewriting Techniques and Applications (RTA)', LNCS 488, Springer-Verlag, Como, Italy, pp. 411–422.

MARCHÉ C. [1996], 'Normalized rewriting: An alternative to rewriting Modulo a set of equations', *Journal of Symbolic Computation* **21**(3), 253–288.

MARTIN U. AND NIPKOW T. [1990], Ordered rewriting and confluence, *in* M. E. Stickel, ed., '10th International Conference on Automated Deduction (CADE)', LNAI 449, Springer-Verlag, Kaiserslautern, FRG, pp. 366–380.

MCCUNE W. [1990], Skolem functions and equality in automated deduction, *in* W. Dietterich, Tom; Swartout, ed., 'Proceedings of the 8th National Conference on Artificial Intelligence', MIT Press, Hynes Convention Centre?, pp. 246–251.

MCCUNE W. [1994], OTTER 3.0 Reference Manual and Guide, Technical Report ANL-94/6, Argonne National Laboratory.

MCCUNE W. [1997*a*], 33 basic test problems: A practical evaluation of some paramodulation strategies, *in* R. Veroff, ed., 'Automated Reasoning and its Applications: Essays in Honor of Larry Wos', MIT Press, pp. 71–114.

MCCUNE W. [1997*b*], 'Solution of the Robbins problem', *Journal of Automated Reasoning* **19**(3), 263–276.

MCCUNE W. [1997*c*], Well behaved search and the Robbins problem, *in* H. Comon, ed., '8th International Conference on Rewriting Techniques and Applications (RTA)', LNCS 1232, Springer-Verlag, Sitges, Spain, pp. 1–7.

NARENDRAN P. AND RUSINOWITCH M. [1991], Any ground associative commutative theory has a finite canonical system, *in* '4th Int. Conf. Rewriting Techniques and Applications (RTA)', LNCS 488, Springer-Verlag, Como, Italy, pp. 423–434.

NARENDRAN P. AND RUSINOWITCH M. [1993], The Unifiability Problem in Ground AC Theories, *in* 'Eighth Annual IEEE Symposium on Logic in Computer Science', IEEE Computer Society Press, Montreal, canada, pp. 364–370.

NARENDRAN P., RUSINOWITCH M. AND VERMA R. [1998], RPO constraint solving is in NP, *in* G. Gottlob, E. Grandjean and K. Seyr, eds, '12th Int. Conference of the European Association of Computer Science Logic (CSL)', LNCS 1584, Springer-Verlag, Brno, Czech Republic.

NIEUWENHUIS R. [1993], 'Simple LPO constraint solving methods', *Information Processing Letters* **47**, 65–69.

NIEUWENHUIS R. [1995], On Narrowing, Refutation Proofs and Constraints, *in* J. Hsiang, ed., '6th International Conference on Rewriting Techniques and Applications (RTA)', LNCS 914, Springer-Verlag, Kaiserslautern, Germany, pp. 56–70.

NIEUWENHUIS R. [1998], 'Decidability and complexity analysis by basic paramodulation', *Information and Computation* **147**, 1–21.

NIEUWENHUIS R. AND NIVELA P. [1991], 'Efficient deduction in equality horn logic by horn-completion', *Information Processing Letters* **39**(1), 1–6.

NIEUWENHUIS R. AND OREJAS F. [1990], Clausal rewriting, *in* S. Kaplan and M. Okada, eds, 'Conditional and Typed Rewriting Systems, 2nd International Workshop', LNCS 516, Springer-Verlag, Montreal, Canada, pp. 246–258.

NIEUWENHUIS R. AND RIVERO J. M. [1999], Solved forms for path ordering constraints, *in* P. Narendran and M. Rusinowitch, eds, 'Tenth International Conference on Rewriting Techniques and Applications (RTA)', LNCS 1631, Springer-Verlag, Trento, Italy, pp. 1–15.

NIEUWENHUIS R. AND RUBIO A. [1992*a*], Basic superposition is complete, *in* B. Krieg-Brückner, ed., 'European Symposium on Programming', LNCS 582, Springer-Verlag, Rennes, France, pp. 371–390.

NIEUWENHUIS R. AND RUBIO A. [1992*b*], Theorem proving with ordering constrained clauses, *in* D. Kapur, ed., '11th International Conference on Automated Deduction (CADE)', LNAI 607, Saratoga Springs, New York, pp. 477–491.

NIEUWENHUIS R. AND RUBIO A. [1994], AC-Superposition with constraints: No AC-unifiers needed, *in* A. Bundy, ed., '12th International Conference on Automated Deduction (CADE)', LNAI 814, Springer-Verlag, Nancy, France, pp. 545–559.

NIEUWENHUIS R. AND RUBIO A. [1995], 'Theorem Proving with Ordering and Equality Constrained Clauses', *Journal of Symbolic Computation* **19**(4), 321–351.

NIEUWENHUIS R. AND RUBIO A. [1997], 'Paramodulation with Built-in AC-Theories and Symbolic Constraints', *Journal of Symbolic Computation* **23**(1), 1–21.

NIVELA P. AND NIEUWENHUIS R. [1993], Practical results on the saturation of full first-order clauses: Experiments with the saturate system. (system description), *in* C. Kirchner, ed., '5th International Conference on Rewriting Techniques and Applications (RTA)', LNCS 690, Springer-Verlag, Montreal, Canada, pp. 436–440.

NUTT W., RÉTY P. AND SMOLKA G. [1989], 'Basic narrowing revisited', *Journal of Symbolic Computation* **7**, 295–317.

PAIS J. AND PETERSON G. [1991], 'Using Forcing to Prove Completeness of Resolution and Paramodulation', *Journal of Symbolic Computation* **11**(1), 3–19.

PAUL E. [1992], 'A general refutational completeness result for an inference procedure based on associative-commutative unification', *Journal of Symbolic Computation* **14**(6), 577–618.

PETERSON G. E. [1983], 'A technique for establishing completeness results in theorem proving with equality', *SIAM J. on Computing* **12**(1), 82–100.

PETERSON G. E. [1990], Complete sets of reductions with constraints, *in* M. E. Stickel, ed., '10th International Conference on Automated Deduction (CADE)', LNAI 449, Springer-Verlag, Kaiserslautern, FRG, pp. 381–395.

PETERSON G. AND STICKEL M. [1981], 'Complete sets of reductions for some equational theories', *Journal Assoc. Comput. Mach.* **28**(2), 233–264.

PLOTKIN G. [1972], 'Building in equational theories', *Machine Intelligence* **7**, 73–90.

PRZYMUSIŃSKA H. AND PRZYMUSIŃSKI T. [1990], 'Weakly Stratified Logic Programs', *Fundamenta Informaticae* **XIII**, 51–65.

PRZYMUSIŃSKI T. [1988], On the declarative semantics of deductive databases and logic programs, *in* 'Foundations of deductive databases and logic programming', Morgan Kaufmann, Los Altos, CA., pp. 193–216.

RETY P., KIRCHNER C. AND LESCANNE P. [1985], NARROWER: a new algorithm for unification and its application to logic programming, *in* J.-P. Jouannaud, ed., '1st International Conference Rewriting Techniques and Applications (RTA)', LNCS 202, Springer-Verlag, Dijon, France, pp. 141–157.

ROBINSON G. A. AND WOS L. T. [1969], 'Paramodulation and theorem-proving in first order theories with equality', *Machine Intelligence* **4**, 135–150.

ROBINSON J. A. [1965], 'A machine-oriented logic based on the resolution principle', *Journal of the ACM* **12**(1), 23–41.

RUBIO A. [1994], 'Automated deduction with ordering and equality constrained clauses', PhD. Thesis, Technical University of Catalonia, Barcelona, Spain.

RUBIO A. [1995], Extension Orderings, *in* '22nd International Colloquium on Automata, Languages and Programming (ICALP)', LNCS 944, Springer-Verlag, Szeged, Hungary, pp. 511–522.

RUBIO A. [1996], Theorem proving modulo associativity, *in* '9th Int. Conference of the European Association for Computer Science Logic (CSL)', LNCS 1092, Springer-Verlag, Paderborn, Germany, pp. 452–467.

RUBIO A. [1999], A fully syntactic AC-RPO, *in* P. Narendran and M. Rusinowitch, eds, 'Tenth International Conference on Rewriting Techniques and Applications (RTA)', LNCS 1631, Springer-Verlag, Trento, Italy, pp. 133–147.

RUBIO A. AND NIEUWENHUIS R. [1995], 'A total AC-compatible ordering based on RPO', *Theoretical Computer Science* **142**(2), 209–227.

RUSINOWITCH M. AND VIGNERON L. [1995], 'Automated deduction with associative commutative operators', *J. of Applicable Algebra in Engineering, Communication and Computation* **6**(1), 23–56.

SLAGLE J. R. [1974], 'Automated theorem-proving for theories with simplifiers, commutativity, and associativity', *Journal of the ACM* **21**(4), 622–642.

SNYDER W. AND LYNCH C. [1991], Goal directed strategies for paramodulation, *in* R. V. Book, ed., '4th Int. Conf. Rewriting Techniques and Applications (RTA)', LNCS 488, Springer-Verlag, Como, Italy, pp. 150–161.

STUBER J. [1998*a*], 'Superposition theorem proving for abelian groups represented as integer modules', *Theoretical Computer Science* **208**(1–2), 149–177.

STUBER J. [1998*b*], Superposition theorem proving for commutative rings, *in* W. Bibel and P. H. Schmitt, eds, 'Automated Deduction - A Basis for Applications. Volume III. Applications', Kluwer, Dordrecht, The Netherlands, chapter 2, pp. 31–55.

SUTCLIFFE G. AND SUTTNER C. B. [1998], The CADE-14 ATP system competition, Technical Report JCU-CS-98/01, Department of Computer Science, James Cook University.
URL: *http://www.cs.jcu.edu.au/ftp/pub/techreports/98-01.ps.gz*

VARDI M. Y. [1982], The complexity of relational query languages (extended abstract), *in* 'Proceedings 14th Annual ACM Symp. on Theory of Computing, STOC'82, San Francisco, CA, USA, 5–7 May 1982', ACM Press, New York, pp. 137–146.

VIGNERON L. [1994], Associative Commutative Deduction with constraints, *in* A. Bundy, ed., '12th International Conference on Automated Deduction (CADE)', LNAI 814, Springer-Verlag, Nancy, France, pp. 530–544.

VIGNERON L. [1996], Positive deduction modulo regular theories, *in* '9th Int. Conference of the European Association for Computer Science Logic (CSL)', LNCS 1092, Springer-Verlag, Paderborn, Germany, pp. 468–486.

WALDMANN U. [1998], Superposition for divisible torsion-free abelian monoids, *in* 'Proceedings of the Fifteenth International Conference on Automated Deduction (CADE-98)', Vol. 1421 of *LNAI*, Springer, Berlin, pp. 144–159.

WALDMANN U. [1999], Cancellative superposition decides the theory of divisible torsion-free abelian groups, *in* 'Logic Programming and Automated Reasoning, Int. Conf.', LNAI 1705, Springer-Verlag, Tiblisi, Georgia, pp. 131–147.

WEIDENBACH C. [1997], 'SPASS—version 0.49', *Journal of Automated Reasoning* **18**(2), 247–252.

WEIDENBACH C. [1999], Towards an automatic analysis of security protocols in first-order logic, *in* H. Ganzinger, ed., 'Proceedings of the 16th International Conference on Automated Deduction (CADE-16)', Vol. 1632 of *LNAI*, Springer-Verlag, Berlin, pp. 314–328.

WERTZ U. [1992], First-order theorem proving modulo equations, Technical Report MPI-I-92-216, Max-Planck-Institut für Informatik, Saarbrücken.

WOS L. [1988], *Automated Reasoning: 33 Basic Research Problems*, Prentice-Hall, Englewood Cliffs.

WOS L. [1996], *The Automation of Reasoning: An Experimenter's Notebook with OTTER Tutorial*, Academic Press.

WOS L., ROBINSON G. A., CARSON D. F. AND SHALLA L. [1967], 'The concept of demodulation in theorem proving', *Journal of the ACM* **14**(4), 698–709.

ZHANG H. [1988], Reduction, Superposition, and Induction: Automated Reasoning in an Equational Logic, PhD thesis, Renselaer Polytechnic Institute.

Index

Topic index