MAX-PLANCK-INSTITUT FÜR INFORMATIK

First–Order Theorem Proving
Modulo Equations

Ulrich Wertz

MPI–I–92–216 April 1992

Im Stadtwald
66123 Saarbrücken
Germany

First–Order Theorem Proving Modulo Equations

Ulrich Wertz

MPI–I–92–216 April 1992

"Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministers für Forschung und Technologie (Betreuungskennzeichen ITS 9103) gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor."

Abstract

We present refutationally complete calculi for first-order clauses with equality. General paramodulation calculi cannot efficiently deal with equations such as associativity and commutativity axioms. Therefore we will separate a set of equations (called E-equations) from a specification and give them a special treatment, avoiding paramodulations with E-equations but using E-unification for the calculi.

Techniques for handling such E-equations known in the context of purely equational specifications (e.g. computing critical pairs with E-equations or introducing extended rules) can be adopted for specifications with full first-order clauses.

Methods for proving completeness results are based on the construction of equality Herbrand interpretations for consistent sets of clauses. These interpretations are presented as a set of ground rewrite rules and a set of ground instances of E-equations forming a Church-Rosser system. The construction of such Church-Rosser systems differs from constructions without considering E-equations in a non-trivial way.

E-equations influence the ordering involved. Methods for defining E-compatible orderings are discussed.

All these aspects are considered especially for the case that E is a set of associativity and commutativity axioms for some operator symbols (then called AC-operators). Some techniques and notions specific to specifications with AC-operators are included.

Contents

1 Introduction

When considering theorem proving, we think of situations with a given *specification* S and a *hypothesis* T. Then we want to answer the question:

Is T a valid *theorem* for S (notation $S \models T$)?

Such problems are important to many areas of computer science, e.g. logic and equational programming, analysis of specifications and verification of software. We will restrict ourselves to specifications with sets of first-order clauses (even more specific: universally quantified first-order clauses, where the only predicate symbol is "$\approx$" (*equality*); other predicates can be coded as boolean functions). Theorem provers based on resolution can efficiently work with non-equational predicates ([Stickel 86]). To handle the equality predicate by resolution like other predicates (e.g. to resolve with the symmetry and transitivity axiom) is far too inefficient. So we will prefer techniques based on paramodulation and use resolution only to incorporate the reflexivity of equality (an approach to logic programming for non-equality predicates, but modulo an equational theory can be found in [Hölldobler 88]). Transforming specifications into sets of clauses and proving theorems by resolution and paramodulation are well established techniques ([Chang/Lee 73], [Loveland 78], [Gallier 87], [Hofbauer/Kutsche 89]). As usual for paramodulation, we will avoid paramodulation into variable positions and with equality axioms (in particular, we will not paramodulate with functional reflexive axioms).

We will prove theorems by *refutation*, so reducing the problem to the question:

Is $S \cup \{\neg T\}$ inconsistent?

We will consider only theorems whose negation is again a first-order clause (or a set of first-order clauses). Of course, *inconsistency* is undecidable in general. For *saturated sets*, however, inconsistency is easy to detect:

A saturated set is inconsistent, if and only if it contains the empty clause.

The definition of *saturation* is designed in a way that the above statement holds. We use inference systems to characterize saturation: A set N of clauses is saturated, if the inferences between clauses in N do not add any 'interesting' new clauses to N, i.e. if all inferences are *redundant*. In the main part of this paper we prove our inference system to have indeed the property that for saturated sets we can decide inconsistency by simply searching for the empty clause. We call this property of the inference system *redundancy-completeness*.

So far saturation is a criterion to decide, whether the search for the empty clause is a valid proof method for inconsistency. Compare this sentence with: Saturation of equational specifications (in the sense of [Knuth/Bendix 70]) is a criterion to decide, whether rewriting is a valid proof method for validity in equational theories. The similar relationship that for saturated sets some efficient proof methods can be used is expressed in [Bertling 90].

Often sets of clauses are not saturated, but we are able to transform them into equivalent saturated sets by a (possibly infinite) process, which computes inferences and adds or deletes clauses. We call this transformation *completion*.

Completion (together with the trivial search for the empty clause and the negation of T) yields a proof method for $S \models T$ (for arbitrary, i.e. not necessary saturated, sets of clauses S and clauses $\neg T$). On the other hand, as for saturated sets efficient proof methods may become available, completion can be regarded as a compilation process, transforming

a (consistent) specification into an equivalent specification, which allows to prove theorems more efficiently. Having this application in mind, we are interested in a finite compilation, i.e. a finite completion process yielding a finite saturated set of clauses.

The main task in designing a theorem prover is to define an appropriate notion of saturation (via an inference system) and to present a method (or even procedure) for completion w.r.t. the previously defined saturation. A lot of work has been done in the area of completion, starting in 1970 with [Knuth/Bendix 70]. Since then, completion has been enhanced and improved on many aspects:

- What specifications are considered?
  (sets of equations, conditional equations (Horn clauses), full first-order clauses):
  [Knuth/Bendix 70] and many, many others consider only equational specifications. Connected with conditional rewriting ([Jouannaud/Waldmann 86], [Kaplan 84], [Kaplan 87]) methods for completion of conditional equational specifications were developed ([Kaplan/Remy 87], [Ganzinger 88a], [Ganzinger 88b], [Ganzinger 91]). Completion was extended to first-order clauses in different ways: using a canonical set of rewrite rules for boolean algebra ([Hsiang/Dershowitz 83], [Hsiang 87], [Bachmair/Dershowitz 87b]); extending the notion of critical pairs to further resolution and paramodulation inferences ([Rusinowitch 87], [Zhang 88], [Zhang/Kapur 88], [Bachmair/Ganzinger 90], [Bachmair/Ganzinger 91a], [Bachmair/Ganzinger 91c]).

- How is the completion method presented?
  (as a concrete procedure or abstracting from control by an inference system):
  To make correctness proofs applicable to a wider class of algorithms, completion was described as an inference system. The control component is reduced to the notion of *fairness*.

- How is the correctness of the method proved?
  (by traditional induction, using semantic trees, using proof orderings, constructing a Herbrand model for consistent sets):
  Induction leads to complicated and hard to read proofs ([Huet 81]). Elegant proofs, at least for standard completion algorithms, use proof orderings ([Bachmair/Dershowitz/Hsiang 86], [Bachmair 87]). Semantic trees can be found in [Rusinowitch 87]. In recent papers often a model is constructed for a saturated set to prove (refutational) completeness of the method ([Zhang 88], [Bachmair/Ganzinger 90], [Bachmair/Ganzinger 91c]).

To reduce search spaces, to obtain decision procedures and for correctness proofs, orderings on terms ([Dershowitz 82], [Dershowitz 87]) were always involved with completion methods. Computing with an equation $\ell \approx r$ means to replace instances $\ell\sigma$ of ℓ by $r\sigma$ or vice versa. This nondeterministic choice of the direction in which we read and apply the equation leads to inefficient and perhaps non-terminating computations with equations. The most efficient way to compute and reason with an equation $\ell \approx r$ is to orient it, obtaining a rewrite rule $\ell \Rightarrow r$. Then instances of this equation are applied only in one direction (replacing $\ell\sigma$ by $r\sigma$) and the above nondeterminism and non-termination are eliminated. In this way computing with a set of rules is an appropriate and efficient method to reason in equational specifications ([Dershowitz 85], [Dershowitz/Jounannaud 89], [Drosten 89], [Dershowitz 89]). One requirement to obtain a complete reasoning method when orienting

equations into rules is the existence of an ordering $>$ over terms such that $\ell\sigma > r\sigma$ for all substitutions σ.

Completion Modulo a Set of Equations

Motivated by permutative equations as the commutativity axiom, which cannot be oriented with well-founded orderings over terms thus cannot be used as rewrite rules, completion methods with a special treatment for those equations were invented. We denote the set of those equations by E and the related completion method by E-completion. The most popular and elaborated example for such a set E is the set of associativity and commutativity axioms for some operator symbols, then called AC-operators. AC-operators often occur in equational (or clausal) specifications, especially for very basic specifications, e.g. in integer arithmetic (AC-operators: addition $+$ and multiplication $*$), in boolean algebras (boolean *and* and *or*) or in specifications of sets (union $\cup$ and intersection $\cap$). There are already completion systems implemented which incorporate E-completion techniques for $E = AC$ (e.g. [Bertling/Ganzinger/Schäfers 89], [Ganzinger/Schäfers 90], [Anantharaman/Hsiang/Mzali 89]). First those special equations influenced the ordering and the notion of critical pairs. Besides of critical pairs between rules, critical pairs with E-equations became necessary. One of the first methods was only able to handle left linear rewrite rules ([Huet 80]). To deal with non-left-linear rules, stronger rewrite relations, like rewriting with E-matching or even rewriting on E-congruence classes, so-called rewriting modulo E, were introduced (e.g. [Peterson/Stickel 81]). Working with stronger rewrite relations, critical pairs were defined using E-unification instead of unification for the empty theory. Methods and their presentation were developed and became applicable to a wider class of theories E ([Jouannaud/Kirchner 86], [Bachmair 87], [Bachmair 88], [Bachmair/Dershowitz 87a]). Using a rewrite relation with runtime orientation of instances of equations, introduced in combination with unfailing completion, completion can handle E-equations without special treatment, e.g. without E-unification or E-matching. [Martin/Nipkow 90] presented saturated systems for the AC-theory of one AC-operator with three equations only, demonstrating that using the rewrite relation of orientable instances of equations, completion can handle the axioms of an AC-theory as any other equations. But nevertheless, E-completion techniques are preferable. They often improve the efficiency of completion. The need for E-completion techniques becomes evident by the fact that in all papers describing harder examples in theorem proving with AC-operators (often algebraic ring structures are involved) E-completion techniques for $E = AC$ are used ([Anantharaman/Hsiang 89], [Anantharaman/Mzali 89], [Lai 90], [Stickel 84]). At a first glance, E-completion uses more complicated (and more inefficient) operations on the base level, e.g. E-unification instead of unification. But seen as an alternative to enumerating the E-congruence classes of two terms s and t and then trying to unify (w.r.t. the empty theory) one of the terms in the congruence class of s and one in the class of t, the E-unification of s and t will often be the better choice, at least from an efficiency point of view. For E-completion we need more complex basic operations, but apply them less often. Refutational theorem proving and completion are search problems, searching for an inconsistency or for a saturated system. The search is pruned by replacing reasoning with E-equations in a nondeterministic way and in a try-and-error fashion by directing the applications of E-equations to solve a certain goal, e.g. to unify two terms. We will show this by an example:

Given an AC-operator $+$ (hence the equations $x+y \approx y+x$ and $(x+y)+z \approx x+(y+z)$

are part of our specification), a rule

$$(x + (-x)) + y \Rightarrow y$$

and a term

$$w + ((v + (-w)) + v)$$

we try to reduce the term with the rule. We cannot match the rule with this term (or a subterm of it), but we can apply the commutativity axiom at three occurrences and the associativity axiom at two occurrences, leading to the terms

$$((v + (-w)) + v) + w$$
$$w + (((-w) + v) + v)$$
$$w + (v + (v + (-w)))$$
$$(w + (v + (-w))) + v$$
$$w + (v + ((-w) + v)).$$

Again we cannot apply the rule to one of these terms, so have to further enumerate terms AC-equal to one of these five terms. Applying some AC-equations will sometimes yield the term we started with. Let us consider only the second term $w + (((-w) + v) + v)$. It is one step AC-equal to

$$(((-w) + v) + v) + w$$
$$w + ((v + (-w)) + v)$$
$$w + (v + ((-w) + v))$$
$$(w + ((-w) + v)) + v$$
$$w + ((-w) + (v + v)).$$

Note that the second term is equal to the term we started with and the third one is equal to the fifth term above. Applying the associativity axiom at the root of the last term, we finally get the term

$$(w + (-w)) + (v + v),$$

which is reducible to $v + v$ by the above rule. Replacing this inefficient search for a reducible term by enumerating AC-equal terms involving a lot of matching operations by a single AC-match (of the rule and the term using the substitution $\{x \leftarrow w,\ y \leftarrow v + v\}$) will improve efficiency although the AC-match is more expensive than a single match w.r.t. the empty theory.

There is another advantage of E-completion: Sometimes using the stronger E-operations and E-completion may yield a finite saturated specification, where usual techniques yield an infinite one (and so also an infinite completion process): E.g. assuming $+$ to be an AC-operator, there is no finite convergent rewrite system to decide the equality w.r.t. $a + b \approx c$ (speaking exactly: the specification consists of the AC-axioms for $+$ and the equation above, moreover we assume $a + b > c$), but using rewriting with AC-matching, a set of two rules ($a + b \Rightarrow c$ and $(a + b) + y \Rightarrow c + y$) is able to decide equality in the specification by rewriting with AC-matching and AC-equality testing (moreover an AC-completion system will find this set of rules automatically). E-completion may be further improved by efficiently implementing E-specific operations (e.g. E-unification) and efficiently representing terms (for $E = AC$: [Fortenbacher 88]).

Outline of the Text

After presenting some basic notions, in particular about rewrite systems which are Church-Rosser modulo E, we define two refutationally complete calculi $\mathcal{M}_E$ and $\mathcal{M}_{Ext}$ for theorem proving modulo E.

The next two sections prove the refutation completeness of these calculi enhanced by refinements for $E = AC$ (in particular for $\mathcal{M}_{Ext}$).

Then we present an abstract completion method (as part of a theorem prover) for $\mathcal{M}_E$ and $\mathcal{M}_{Ext}$ together with general criteria to show the irrelevance of clauses and inferences during completion.

Some variants of our calculi are discussed in section 8.

After providing methods to define orderings for our proof systems, we conclude our work with a comparison to related theorem proving methods.

2 Basic Definitions and Notations

An *equation* is a multiset $\{s, t\}$, where s and t are (first-order) terms. We write $s \approx t$ to denote the equation $\{s, t\}$.

A *clause* is a pair (Γ, Δ) of multisets of equations, written as $\Gamma \to \Delta$. The multiset Γ is called the *antecedent*; the multiset Δ, the *succedent* of the clause $\Gamma \to \Delta$. We usually write Γ_1, Γ_2 instead of $\Gamma_1 \cup \Gamma_2$; Γ, A or A, Γ instead of $\Gamma \cup \{A\}$; and $A_1, \ldots, A_m \to B_1, \ldots, B_n$ instead of $\{A_1, \ldots, A_m\} \to \{B_1, \ldots, B_n\}$. A clause $A_1, \ldots, A_m \to B_1, \ldots, B_n$ may be regarded as representing an implication $A_1 \wedge \cdots \wedge A_m$ *implies* $B_1 \vee \cdots \vee B_n$ or a disjunction $\neg A_1 \vee \cdots \vee \neg A_m \vee B_1 \vee \cdots \vee B_n$ of positive (B_i) and negative (negated; A_j) equational literals.

We ambiguously denote some operations on sets and on multisets by the same symbols, e.g. $\cup$ for union and $\cap$ for intersection. It should be clear from the context, whether we operate on sets or multisets.

A *ground* term (equation, clause) is a term (equation, clause) without any variables. A ground substitution replaces each variable by a ground term.

We write $u[s]$ to indicate that s is a *subterm* of u and (ambiguously) denote by $u[t]$ the result of replacing s at a particular occurrence in u by t. Sometimes we use a more precise notation: If p is an occurrence of a subterm s in t, then t/p denotes s and $t[p \leftarrow r]$ denotes the result of replacing the subterm at occurrence p of t by r. For theories with associative and commutative functions, we will also use the following notion of subterms (assuming that f is an AC-operator):

$$subterms_{AC}(f, h(t_1, \ldots, t_n)) = \begin{cases} \bigcup_{i=1}^{i=n} subterms_{AC}(f, t_i) & \text{if } f = h \\ \{h(t_1, \ldots, t_n)\} & \text{otherwise} \end{cases}$$

By $O(t)$ we denote the set of all *occurrences* (also called *positions*) in t. Occurrences or positions are represented as lists of integers using the dot-notation, e.g. 1.2 denotes the occurrence of s in $f(g(a, s), b)$. The root position is denoted by ε. The *height* of a term t is the maximal length of an occurrence in $O(t)$, e.g. the height of the previous term is 2.

A *context* c is a term containing exactly one position p with $c/p = []$, where $[]$ is regarded as a placeholder for other terms and $c[t]$ here denotes the replacement of $[]$ by t.

By $t\sigma$ we denote the result of applying the substitution σ to t, and call $t\sigma$ an *instance* of t. If $t\sigma$ is ground, we speak of a *ground instance*. We also apply substitutions to equations,

multisets of equations and clauses. The *composition* $\tau\rho$ of two substitutions τ and ρ is defined by: $x\tau\rho = (x\tau)\rho$, for all variables x.

An interpretation I is a congruence on ground terms. An interpretation I is said to satisfy a ground clause $\Gamma \rightarrow \Delta$ if either $\Gamma \not\subseteq I$ (we say "the antecedent is not satisfied") or $\Delta \cap I \neq \emptyset$ ("the succedent is satisfied by I"). For a ground clause C we also say C *is true in* I instead of I *satisfies* C; otherwise C *is false in* I. A non-ground clause D is satisfied by I, if all its ground instances $C\sigma$ are satisfied by I. A clause C is said to be *unsatisfiable*, if there is no interpretation I such that I satisfies C, e.g. the empty clause is unsatisfiable. An interpretation I is a *model* for a set of clauses N, if I satisfies all clauses in N. So speaking of a model M, we assume that the equality symbol $\approx$ is interpreted by a congruence, hence in particular the congruence axioms for $\approx$ (which are Horn clauses) are satisfied by M. N is called *consistent*, if it has a model, otherwise N is *inconsistent*. We say that N *implies* C (and write $N \models C$), if every model of N satisfies also C.

An equation $u \approx v$ is *variable preserving*, if we have $vars(u) = vars(v)$.

Let E be a set of (unconditional) and variable preserving equations. We remark already here, that most of our results hold only, if E satisfies more requirements (see definitions 4.1, 5.23 and 6.37). We use the prime to indicate E-equality, e.g. if t is a term, t' is E-equal to it (the same for clauses C and C' or substitutions σ and σ'). We write $t =_E t'$ or $t \Longleftrightarrow_E^* t'$ to express E-equality and $t \Longleftrightarrow_E t'$ if this equality is due to a single application of an instance of an E-equation. We use $[t]_E := \{t' \mid t' =_E t\}$ to denote the congruence class of all terms E-equal to t. If E is variable preserving and t is a ground term, then $[t]_E$ is a set of ground terms, too.

A substitution σ with $t\sigma =_E s\sigma$ is called an *E-unifier* of t and s. We assume that (E-) unifiers never unnecessarily instantiate variables, i.e. if σ is a unifier of s and t we have $x\sigma = x$ for all variables x not occurring in t or s.

There are clauses consisting only of one equation in the succedent and an empty antecedent (e.g. all E-equations/clauses). We sometimes write them simply as equations, i.e. $e_1 \approx e_2$ instead of $\rightarrow e_1 \approx e_2$.

Definition 2.1 (Compatibility with E)
Let E be a set of equations. An ordering $>$ on terms is *compatible with* E (or for short E-compatible), if $t > s$ implies $t' > s'$ for all $t' \in [t]_E$ and $s' \in [s]_E$.

If we have a reduction ordering $>$ which is compatible with E, we can define a reduction ordering $>_{[]}$ on E-congruence classes:

$$[t]_E >_{[]} [s]_E \qquad \text{if and only if} \qquad t > s$$

Therefore we may ambiguously write $t >_E s$ instead of $[t]_E >_E [s]_E$ and speak of an E-compatible reduction ordering $>_E$.

The non-strict version $\geq_E$ of a binary relation $>_E$ on terms is defined by $t \geq_E s$ if and only if $t >_E s$ or $t =_E s$.

3 Rewrite Systems

Definition 3.1 (Rewriting with a Set of Rules R)
Let R be a rewrite system, i.e. a set of equations $\ell \approx r$ directed to rules $\ell \Rightarrow r$ and with $vars(r) \subseteq vars(\ell)$.

A term t *rewrites to* s (we write $t \Longrightarrow_R s$) if and only if there is a position $p \in O(t)$, a rule $\ell \Rightarrow r$ and a substitution σ with $t/p = \ell\sigma$ and $s = t[u \leftarrow r\sigma]$.

A term t is called *reducible (by R, or more exactly by $\Longrightarrow_R$)* if and only if there is an s with $t \Longrightarrow_R s$. Otherwise the term t is called *irreducible (by $\Longrightarrow_R$)*.

If $\Longrightarrow$ is a binary relation, then $\Longrightarrow^+$ denotes its transitive, $\Longrightarrow^*$ its transitive and reflexive and $\Longleftrightarrow$ its symmetric closure.

Working with rewrite systems modulo a given theory E we often find rewrite relations as introduced by the following two definitions (we only give this definition to explain why we do not use them in our context, i.e. not in our proofs; but they are useful in implementations of E-completion procedures):

Definition 3.2 (Rewriting with E-Matching)
Let R be a rewrite system. A term t *rewrites with E-matching to* s (we write $t \Longrightarrow_{R \cdot E} s$) if and only if there is a position $p \in O(t)$, a rule $\ell \Rightarrow r$ and a substitution σ with $t/p =_E \ell\sigma$ and $s = t[u \leftarrow r\sigma]$.

A term t is called *reducible by R using rewriting with E-matching* or *reducible by* $\Longrightarrow_{R \cdot E}$ if and only if there is an s with $t \Longrightarrow_{R \cdot E} s$.

There is an alternative characterization of $\Longrightarrow_{R \cdot E}$: let

$$S = \{(\ell\sigma)' \Rightarrow r\sigma \mid \ell \Rightarrow r \in R,\ (\ell\sigma)' =_E \ell\sigma\};$$

we have $\Longrightarrow_{R \cdot E} = \Longrightarrow_S$. For implementation issues we prefer the former definition, because here S is in general infinite (even for finite sets R and theories E with finite congruence classes). The relations become different when considering conditional rewriting, because for conditional rewriting we have to define how to use E-equations to solve the condition (cf. [Wertz 89]).

Definition 3.3 (Rewriting modulo E)
Let R be a rewrite system. A term t *rewrites modulo E to* s (we write $t \Longrightarrow_{R/E} s$) if and only if there are terms t' and s' with $t' =_E t$, $s' =_E s$ and t' rewrites to s' using a rule of R, i.e. $t' \Longrightarrow_R s'$.

For $E = AC$ a relation that is efficiently to implement and able to simulate $\Longrightarrow_{R/AC}$ is presented in section 6.4.3. For general E it is impractical to work with $\Longrightarrow_{R/E}$ because of the arbitrary location of implicit $\Longleftrightarrow_E$-steps involved.

Definition 3.4 (Church-Rosser Property modulo E)
Let $\Longrightarrow$ be a rewrite relation. $\Longrightarrow$ is *(ground) Church-Rosser modulo E for the congruence* $\equiv$ if and only if for all (ground) terms t and s such that

$$t \equiv s$$

there are (ground) terms v and v' with

$$t \Longrightarrow^* v =_E v' \Longleftarrow^* s\ .$$

The word *modulo* in the above definition does not refer to the rewrite relation $\Longrightarrow$. Usually $\Longrightarrow$ is a relation $\Longrightarrow_R$ or $\Longrightarrow_{R \cdot E}$ (i.e. defined by a set R of rules and a method of rewriting, e.g. rewriting with E-matching). The congruence $\equiv$ is in general the congruence $\equiv_{R \cup E}$.

In the following sections we will construct an interpretation as a model for a consistent set of clauses. This interpretation is represented by a set E_C of ground instances of E-equations and a set R of ground rewrite rules. We will need interpretations which are Church-Rosser modulo E_C for $\equiv_{R \cup E_C}$. We want to use rewrite systems which are confluent (see definition below) by construction. But we cannot use any of the above rewrite relations: even if the left side of a rule is irreducible by all other rules (applied with E-matching or modulo E) the corresponding rewrite relation with E-matching or modulo E need not be confluent (modulo E), cf. example 3.5. We therefore use the "normal" rewrite relation $\Longrightarrow_R$ in the definition of an interpretation (see definition 5.8). In the rest of this section we present some notions used to prove the Church-Rosser property of rewrite systems. We outline the more complicated case using the rewrite relation with E-matching and give exact proofs for the relation we need in the rest of this paper.

Example 3.5 (Confluence using rewriting modulo E or with E-matching)
Let $+$ be an AC-operator. Consider rewriting modulo AC.

$$R = \{a + b \Rightarrow d, b + c \Rightarrow e\}$$

We have no overlappings but a divergent situation

$$d + c \Longleftarrow_R (a + b) + c =_{AC} a + (b + c) \Longrightarrow_R a + e$$

so

$$d + c \Longleftarrow_{R/AC} a + b + c \Longrightarrow_{R/AC} a + e$$

The relation $\Longrightarrow_{R/AC}$ is not Church-Rosser modulo AC for $\equiv_{R \cup AC}$. For rewriting with AC-matching consider

$$R = \{a + b \Rightarrow d, a + (b + c) \Rightarrow e\}$$

and

$$d + c \Longleftarrow_R (a + b) + c =_{AC} a + (b + c) \Longrightarrow_R e$$

so

$$d + c \Longleftarrow_{R \cdot AC} a + b + c \Longrightarrow_{R \cdot AC} e$$

The relation $\Longrightarrow_{R \cdot AC}$ is not Church-Rosser modulo AC for $\equiv_{R \cup AC}$.

The examples show us that we cannot decide Church-Rosser properties for the above rewrite relations by simply considering overlaps between left sides of rules (e.g. by confluence of critical pairs). So we will not use these relations to construct Church-Rosser rewrite systems.

Definition 3.6 (Coherence modulo E)
Let $\Longrightarrow$ be a rewrite relation. $\Longrightarrow$ is *(ground) coherent modulo E* if and only if for any (ground) terms t, t' and s such that

$$t =_E t' \Longrightarrow^+ s$$

there are terms v and v' with

$$t \Longrightarrow^* v =_E v' \Longleftarrow^* s \ .$$

In [Jouannaud/Kirchner 86] a slightly different definition is used: the condition $t \Longrightarrow^* v$ is replaced by $t \Longrightarrow^+ v$, i.e. there is at least one rewrite step required. The definitions become equivalent, however, if the relation $\Longrightarrow_{R/E}$ is terminating.

Definition 3.7 (Confluence modulo E)
Let $\Longrightarrow$ be a rewrite relation. $\Longrightarrow$ is *(ground) confluent modulo E* if and only if for any (ground) terms u, t and s such that

$$t \Longleftarrow^* u \Longrightarrow^* s$$

there are terms v and v' with

$$t \Longrightarrow^* v =_E v' \Longleftarrow^* s \ .$$

If we restrict the application of rewrite rules and E-equations in the above definitions to one step of E-equality and one step of rewriting, we speak of *local coherence modulo E* and *local confluence modulo E*. Again, the definition for local confluence in [Jouannaud/ Kirchner 86] is different to the one above. The weaker definition of Jouannaud and Kirchner is only appropriate for rewrite relations included in $\Longrightarrow_{R \cdot E}$. Jouannaud and Kirchner have shown that in case of termination of $\Longrightarrow_{R/E}$ the local confluence and coherence properties are equivalent to the Church-Rosser property. To be more precise: the Church-Rosser property holds if global confluence and global coherence are both given; from the local confluence *and* the local coherence we can conclude both global properties, but *local confluence does not imply global confluence* and *local coherence does not imply global coherence*. With the same proof we can conclude the equivalence of the ground versions of these properties.

We will give a short outline of the Church-Rosser theorem in [Jouannaud/Kirchner 86]: As usual E-critical pairs are defined similar to critical pairs, but using E-unification instead of unification (p. 1166). Similar to the [Jouannaud/Kirchner 86]-version of local confluence (page 1159), this notion of E-critical pairs is only suited for rewrite relations included in $\Longrightarrow_{R \cdot E}$. The confluence of all critical pairs implies the local confluence modulo E (as usual other peaks are either variable overlaps or applications with disjoint redexes). Note that this implication *confluence of critical pairs implies local confluence*, does only hold for $\Longrightarrow_{R \cdot E}$ and the [Jouannaud/Kirchner 86]-definition of local confluence: in example 3.5 we give a local (i.e. both reductions are one step reductions) divergence without E-critical pairs, [Jouannaud/Kirchner 86] excludes this divergence using a definition of local confluence which does not allow the application of the greater rule with E-matching.

Note also that the relation $\Longrightarrow_{R/E}$ is not confluent, if all the critical pairs are joinable (see again example 3.5); but if the set of rules contains all E-extensions (see chapter 6.1 for the definition of E-extension; for sets R closed under E-extension we have $t \Longrightarrow_{R/E} s$ *implies* $t \Longrightarrow_{R \cdot E} s'$ similar to lemma 6.70), we again can conclude the confluence (and therefore the Church-Rosser property, because $\Longrightarrow_{R/E}$ is trivially coherent) of the relation $\Longrightarrow_{R/E}$ from the confluence of all critical pairs.

In the same way the local coherence follows from the confluence of all E-critical pairs between rules and E-equations (as E-equations are not directed, we superpose on both sides of them).

Altogether, Jouannaud and Kirchner have given a method to decide the Church-Rosser property for relations $\Longrightarrow_{R \cdot E}$ (or weaker relations) by considering E-critical pairs only.

We will define rewrite systems R only, where R is a (sometimes infinite) set of ground rewrites rules. We will not use rewriting with matching or modulo E but simply $\Longrightarrow_R$.

Moreover the relation $\Longrightarrow_R$ will be confluent (and therefore also satisfy the weaker property of confluence modulo E) by construction. For this case the proof of Church-Rosser properties becomes easier (see below, lemma 3.9).

Definition 3.8 (Joinability modulo E)
Let t and s be (ground) terms. t and s are *joinable modulo E by* $\Longrightarrow$ if there are (ground) terms v and v' with $v =_E v'$, $t \Longrightarrow^* v$ and $v' \Longleftarrow^* s$. An equation $t \approx s$ is joinable, if t and s are joinable.

If E and $\Longrightarrow$ are clear from the context, we often simply say "t and s are joinable" and write $t \Downarrow s$ in this situation.

Lemma 3.9 (Sufficient Criterion for Church-Rosser Property)
Let R be a set of ground rewrite rules such that the left sides of any two different rules do not overlap. Let $\Longrightarrow_{R/E}$ be terminating. Then $\Longrightarrow_R$ is confluent. If $\Longrightarrow_R$ is in addition locally ground coherent modulo E, then $\Longrightarrow_R$ is ground Church-Rosser modulo E for $\equiv_{R \cup E}$.

Proof: First we prove local confluence: If a ground term u is reducible (in one step) by $rule_1 := \ell_1 \Rightarrow r_1$ to s and by $rule_2 := \ell_2 \Rightarrow r_2$ to t, then the rules are applied in disjoint subterms u/p_1 and u/p_2 (otherwise the left sides of $rule_1$ and $rule_2$ would overlap, contradicting our assumptions). The situation is:

$$s = u[p_1 \leftarrow r_1][p_2 \leftarrow \ell_2] \Longleftarrow_R u = u[p_1 \leftarrow \ell_1][p_2 \leftarrow \ell_2] \Longrightarrow_R u[p_1 \leftarrow \ell_1][p_2 \leftarrow r_2] = t$$

By application of $rule_2$ to s and $rule_1$ to t we get the desired confluence:

$$s = u[p_1 \leftarrow r_1][p_2 \leftarrow \ell_2] \Longrightarrow_R u[p_1 \leftarrow r_1][p_2 \leftarrow r_2] \Longleftarrow_R u[p_1 \leftarrow \ell_1][p_2 \leftarrow r_2] = t$$

Now we prove the global confluence. We need the termination of the rewrite relation $\Longrightarrow_R$ which is implied by the termination of $\Longrightarrow_{R/E}$. As an induction hypothesis assume that the confluence holds for all peaks at terms smaller than u (w.r.t. the ordering induced by $\Longrightarrow_{R/E}$), i.e. if we have a term t, $u \Longrightarrow^+_{R/E} t$ and a divergence

$$s_1 \Longleftarrow^*_R t \Longrightarrow^*_R s_2,$$

then we can join s_1 and s_2:

$$s_1 \Longrightarrow^*_R v \Longleftarrow^*_R s_2$$

Let us now consider the divergence (peak)

$$s \Longleftarrow^*_R u \Longrightarrow^*_R t.$$

If there is not at least one rewrite step in each of the reductions, then we have $s = u$ or $t = u$ and the confluence is trivial. Otherwise we have

$$s \Longleftarrow^*_R s_1 \Longleftarrow_R u \Longrightarrow_R t_1 \Longrightarrow^*_R t$$

and by local confluence

$$s \Longleftarrow^*_R s_1 \Longrightarrow^*_R v \Longleftarrow^*_R t_1 \Longrightarrow^*_R t.$$

Now by induction hypothesis (note $u \Longrightarrow^+_{R/E} s_1$ and $u \Longrightarrow^+_{R/E} t_1$) we get a derivation

$$s \Longrightarrow^*_R v_1 \Longleftarrow^*_R v \Longrightarrow^*_R v_2 \Longleftarrow^*_R t.$$

Again with induction for the peak at v (note $u \Longrightarrow_R s_1 \Longrightarrow^*_R v$ implies $u \Longrightarrow^+_{R/E} v$) we obtain a rewrite proof for $s \approx t$:

$$s \Longrightarrow^*_R v_1 \Longrightarrow^*_R w \Longleftarrow^*_R v_2 \Longleftarrow^*_R t$$

This was the proof of the confluence of $\Longrightarrow_R$. From now on we assume the local coherence modulo E of $\Longrightarrow_R$ and conclude the global coherence modulo E. For each E-equality between ground terms t and t' there is an $n \in \mathbb{N}$, and for $1 \leq i \leq n$ there are terms t_i, positions p_i and ground instances $e_{i1} \approx e_{i2}$ of equations in E such that we have an equality proof

$$t = t_1 =_E t_2 =_E \ldots =_E t_n = t'$$

and each t_i $(1 \leq i < n)$ is E-equal to t_{i+1} by application of the equation $e_{i1} \approx e_{i2}$ at position p_i of t_i. We then use two induction hypotheses:

1. We assume that for all peaks
$$u' =_E u'' \Longrightarrow^+_R s$$
where the equality proof for $u' =_E u''$ consists of less than n steps we have
$$u' \Longrightarrow^+_R v =_E v' \Longleftarrow^*_R s\ .$$

2. Moreover, let us assume that for all equality proofs $t \Longleftrightarrow^*_{R \cup E} s$ containing only terms smaller than u (w.r.t. $\Longrightarrow_{R/E}$) there exists a rewrite proof $t \Longrightarrow^*_R v' =_E v \Longleftarrow^*_R s$.

Now we consider a peak

$$u' \Longleftrightarrow^*_E u'' \Longleftrightarrow_E u \Longrightarrow_R s_1 \Longrightarrow^*_R s$$

where u'' is E-equal to u by one step of E-equality and u' is E-equal to u'' by at most $n-1$ steps of E apllications. By local coherence we have

$$u' \Longleftrightarrow^*_E u'' \Longrightarrow^+_R v' =_E v \Longleftarrow^*_R s_1 \Longrightarrow^*_R s.$$

Using the induction hypothesis 1 at u'' we get

$$u' \Longrightarrow^+_R w_1 =_E w_1' \Longleftarrow^*_R v' =_E v \Longleftarrow^*_R s_1 \Longrightarrow^*_R s.$$

The proof for $w_1 \Longleftrightarrow^*_{R \cup E} s$ contains only terms smaller than u (note that the termination of $\Longrightarrow_R$ is not sufficient to show e.g. that u is greater than v', we therefore need the termination of $\Longrightarrow_{R/E}$), so with induction hypothesis 2 applied to $w_1 \Longleftrightarrow^*_{R \cup E} s$ we finally obtain the rewrite proof

$$u' \Longrightarrow^+_R w_1 \Longrightarrow^*_R w_2 =_E w_2' \Longleftarrow^*_R s.$$

Alltogether the relation $\Longrightarrow_R$ is terminating, globally confluent and coherent modulo E. The same way as done in the Church-Rosser theorem in [Jouannaud/Kirchner 86] we can easily prove the Church-Rosser property replacing coherence and confluence peaks of equational proofs by rewrite proofs as constructed above. Iterating this process decreases the number of peaks in a proof until a rewrite proof is reached. □

4 Inference Systems

We will present inference systems for theorem proving modulo E, where E is a certain equational theory:

Definition 4.1 E is a set of (unconditional) equations with the following properties:

- Every equation $e_1 \approx e_2 \in E$ is variable preserving, i.e. $vars(e_1) = vars(e_2)$.
- E is finite.
- For all terms s and t the minimal complete set of E-unifiers $\mu CSU_E(s,t)$ is computable and finite.
- There exists a total and E-compatible reduction ordering $>_E$ comparing E-congruence classes of ground terms.

We want to present an inference system which can be used in a computer implementation of a theorem prover. We therefore require finiteness and computability. But we remark, that most of our results do not depend on these facts.

The existence of minimal sets of unifiers is one severe restriction. An even more restrictive requirement may be the existence of a total and E-compatible reduction ordering (see section 9). For nearly all theories E the existence of such an ordering is an open question.

Example 4.2 (The Theory $E = AC$)
The theory of commutative and associative function symbols, AC-theory for short, where a finite set of operator symbols f are *AC-operators*, i.e. the following equations hold for every such operator f

$$\begin{array}{rcll} f(f(x,y),z) & = & f(x,f(y,z)) & \text{(associativity)} \\ f(x,y) & = & f(y,x) & \text{(commutativity)} \end{array}$$

is an appropriate candidate for E. It is obviously finite and variable preserving. For AC-unification see [Stickel 81] and [Fages 87]. For an appropriate reduction ordering see section 9.2.

We define our inference systems based on the following ordering:

Definition 4.3 (Ordering over Equations)
The *E-multiset expression* of an (occurrence of an) equation $t_1 \approx t_2$ in a clause $\Gamma \rightarrow \Delta$ is defined as

(i) $\{\{[t_1]_E, \bot_E\}, \{[t_2]_E, \bot_E\}\}$ if $t_1 \approx t_2$ belongs to Γ.
(ii) $\{\{[t_1]_E\}, \{[t_2]_E\}\}$ if $t_1 \approx t_2$ belongs to Δ.

The ordering $>_{eq}$ over (occurrences of) ground equations is defined as the multiset extension of (the multiset extension of) $>_E$ on their E-multiset expressions. We assume $\bot_E$ to be smaller (w.r.t. $>_E$) than any congruence class of ground terms.

If the occurrences and the clauses of two equations are clear from the context, we often omit these informations and simply say that one equation is greater than the other. If we compare an equation $t \approx s$ not occurring in a clause, we identify it with its occurrence in $\rightarrow t \approx s$.

Definition 4.4 (Maximality)
Let t be a term (equation, clause) and M be a multiset of terms (equations, clauses) containing t. Let $>$ denote an ordering on terms (equations, clauses). We say that t is *maximal (w.r.t. $>$)* in M, if there is no $s \in M$ such that $s > t$. We say that t is *strictly maximal (w.r.t. $>$)* in M, if for all $s \in M \setminus \{t\}$ we neither have $s > t$ nor $s = t$. Considering E-compatible orderings we also require $s \neq_E t$ for strict maximality. Note that with $M \setminus \{t\}$ we here denote the multiset difference: deleting exactly one occurrence of t from M we get the above multiset difference.

When considering total orderings we may define strict maximality of t (w.r.t. $>$) in M by *for all $s \in M \setminus \{t\}$, we have $t > s$.*

4.1 Inference Rules

Now we present our inference system. It is an extension of the system presented by Nieuwenhuis ([Nieuwenhuis 91]), which itself is a modification of the system of Bachmair and Ganzinger ([Bachmair/Ganzinger 90], [Bachmair/Ganzinger 91a], [Bachmair/Ganzinger 91c]).

Definition 4.5 (Inference Rules)
We always assume that the premises of the following inferences have disjoint sets of variables (otherwise rename the variables of one clause). We consider the following inference rules:

1. strict superposition right

$$\frac{\Gamma_1 \rightarrow \Delta_1, s_1 \approx s_2 \quad \Gamma_2 \rightarrow \Delta_2, t_1 \approx t_2}{(\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2, t_1[u \leftarrow s_2] \approx t_2)\sigma}$$

   where $\sigma \in \mu CSU_E(t_1/u, s_1)$, t_1/u is not a variable and there exists a ground substitution σ_1 such that

   a) $t_1\sigma\sigma_1 >_E t_2\sigma\sigma_1$, $s_1\sigma\sigma_1 >_E s_2\sigma\sigma_1$, and $t_1\sigma\sigma_1 \approx t_2\sigma\sigma_1 >_{eq} s_1\sigma\sigma_1 \approx s_2\sigma\sigma_1$

   b) $s_1\sigma\sigma_1 \approx s_2\sigma\sigma_1$ is strictly maximal (w.r.t. $>_{eq}$) in $(\Gamma_1 \rightarrow \Delta_1, s_1 \approx s_2)\sigma\sigma_1$

   c) $t_1\sigma\sigma_1 \approx t_2\sigma\sigma_1$ is strictly maximal (w.r.t. $>_{eq}$) in $(\Gamma_2 \rightarrow \Delta_2, t_1 \approx t_2)\sigma\sigma_1$

2. strict superposition left

$$\frac{\Gamma_1 \rightarrow \Delta_1, s_1 \approx s_2 \quad \Gamma_2, t_1 \approx t_2 \rightarrow \Delta_2}{(\Gamma_1, \Gamma_2, t_1[u \leftarrow s_2] \approx t_2 \rightarrow \Delta_1, \Delta_2)\sigma}$$

   where $\sigma \in \mu CSU_E(t_1/u, s_1)$, t_1/u is not a variable and there exists a ground substitution σ_1 such that

   a) $t_1\sigma\sigma_1 >_E t_2\sigma\sigma_1$ and $s_1\sigma\sigma_1 >_E s_2\sigma\sigma_1$

   b) $s_1\sigma\sigma_1 \approx s_2\sigma\sigma_1$ is strictly maximal (w.r.t. $>_{eq}$) in $(\Gamma_1 \rightarrow \Delta_1, s_1 \approx s_2)\sigma\sigma_1$

   c) $t_1\sigma\sigma_1 \approx t_2\sigma\sigma_1$ is maximal (w.r.t. $>_{eq}$) in $(\Gamma_2, t_1 \approx t_2 \rightarrow \Delta_2)\sigma\sigma_1$

3. equality resolution

$$\frac{\Gamma, t_1 \approx t_2 \rightarrow \Delta}{(\Gamma \rightarrow \Delta)\sigma}$$

   where $\sigma \in \mu CSU_E(t_1, t_2)$ and there exists a ground substitution σ_1 such that

a) $t_1\sigma\sigma_1 \approx t_2\sigma\sigma_1$ is maximal (w.r.t. $>_{eq}$) in $(\Gamma, t_1 \approx t_2 \rightarrow \Delta)\sigma\sigma_1$

4. equality factoring

$$\frac{\Gamma \rightarrow \Delta, t_1 \approx s_1, t_2 \approx s_2}{(\Gamma, s_1 \approx s_2 \rightarrow \Delta, t_2 \approx s_2)\sigma}$$

where $\sigma \in \mu CSU_E(t_1, t_2)$ and there exists a ground substitution σ_1 such that

a) $t_1\sigma\sigma_1 >_E s_1\sigma\sigma_1$ and $t_2\sigma\sigma_1 >_E s_2\sigma\sigma_1$

b) $t_1\sigma\sigma_1 \approx s_1\sigma\sigma_1$ is maximal (w.r.t. $>_{eq}$) in $(\Gamma \rightarrow \Delta, t_1 \approx s_1, t_2 \approx s_2)\sigma\sigma_1$

5. E-closure

$$\frac{\Gamma \rightarrow \Delta, s_1 \approx s_2 \quad \rightarrow e_1 \approx e_2}{(\Gamma \rightarrow \Delta, e_1[u \leftarrow s_2] \approx e_2)\sigma}$$

where $\sigma \in \mu CSU_E(e_1/u, s_1)$, e_1/u is not a variable, $u \neq \varepsilon$, $e_1 \approx e_2 \in E$ and there exists a ground substitution σ_1 such that

a) $s_1\sigma\sigma_1 >_E s_2\sigma\sigma_1$

b) $s_1\sigma\sigma_1 \approx s_2\sigma\sigma_1$ is strictly maximal (w.r.t. $>_{eq}$) in $(\Gamma \rightarrow \Delta, s_1 \approx s_2)\sigma\sigma_1$

c) $e_1\sigma\sigma_1 >_E s_1\sigma\sigma_1$ (therefore $e_1\sigma\sigma_1 \neq_E s_1\sigma\sigma_1$)

This is a version of *superposition right* on E-equations.

The clauses on the top of the horizontal bars are called *premises*, and those below *conclusions.*

Definition 4.6 (Inference Systems $\mathcal{M}_{Ext}$ and $\mathcal{M}_E$)
The inference systems $\mathcal{M}_{Ext}$ consists of the following inference rules:

1. superposition right
2. superposition left
3. equality resolution
4. equality factoring

The inference system $\mathcal{M}_E$ consists of the inference rules in $\mathcal{M}_{Ext}$ and the E-closure inference.

Note that due to ordering constraints the premise of an $\mathcal{M}_{Ext}$-inference can never be an instance of an E-equation. E-equations are only used as premises of E-closure inferences. Note also that we do not superpose into variable positions nor require explicit equality axioms as premises (such as symmetry $x \approx y \rightarrow y \approx x$ or functional reflexive axioms). And we will not require such equality axioms to be contained in our specifications.

Note that the existence of a ground substitution σ_1 in the above inference rules is undecidable in general. But we can approximate the above inference rules by decidable ones, if we admit some more inferences. Let $>$ be an E-compatible, partial and stable reduction ordering over terms with variables which is completable to $>_E$, i.e. for any ground terms t and s we have $t > s$ implies $t >_E s$. Note that the empty relation is an example for $>$, but a stronger relation will result in less inferences. Similar to $>_{eq}$ we can define a partial ordering over occurrences of equations based on $>$. Replacing the maximality of a term $t\sigma\sigma_1$ or an equation $e\sigma\sigma_1$ based on $>_E$ by the maximality of $t\sigma$ or $e\sigma$ (without applying σ_1)

based on $>$, we get decidable inference rules (if $>$ is decidable). As $t\sigma\sigma_1 >_E s\sigma\sigma_1$ implies $t \not\leq s$, we have an inference π with maximality w.r.t. $>$, whenever we have an inference π with the above inference rules based on $>_E$. The opposite does not hold: for some inferences we have to require more than one comparison, e.g. $t\sigma \not\leq s\sigma$ and $t\sigma \not\leq r\sigma$. Even if we have ground substitutions σ_2 and σ_3 with $t\sigma\sigma_2 >_E s\sigma\sigma_2$ and $t\sigma\sigma_3 >_E r\sigma\sigma_3$, we do not have (in general) a ground substitution σ_1 achieving the maximality in both comparisons simultaneously. See example 8.2 and the discussion in section 8.1.

Lemma 4.7 (Soundness of Inference Rules)
Let

$$\frac{C \quad D}{B} \qquad \text{or} \qquad \frac{D}{B}$$

be an inference of $\mathcal{M}_E$. We have

$$E \cup \{C, D\} \models B \qquad \text{or} \qquad E \cup \{D\} \models B,$$

respectively

As E-unification is involved in inferences, we cannot omit the E from the above lemma. Let N be a set of clauses. Regarding $N \cup E$ as our specification, by the above lemma we do not change our specification, if we consider inferences (e.g. add conclusions of inferences to N). So what is the use of inferences? On the one hand, inferences will help us in defining properties of sets of clauses (e.g. saturation, see below), on the other hand we can regard it as a method to detect a new presentation of our specification (e.g. by adding conclusions of inferences), such that the new presentation is more appropriate for theorem proving purposes (saturated sets are appropriate for theorem proving). This way the conclusions of inferences show us interesting statements about our specification.

Example 4.8 (Superposition Right)
For equational specifications this inference is often referred to as *critical pair* construction. Let $*$ be an AC-operator. Using the AC-unifier $\{x \leftarrow inv(0), y \leftarrow 0\}$ of the terms $0 * x$ and $inv(y) * y$ we could get a superposition right inference

$$\frac{\rightarrow 0 * x \approx 0 \quad \rightarrow inv(y) * y \approx 1}{\rightarrow 0 \approx 1}.$$

So it is not appropriate to specify a multiplication operation with zero 0 and inverse operation inv (e.g. for natural numbers) by the above two clauses (unless we want to imply $0 \approx 1$, which is in general not intended, e.g. not for natural numbers). We could use

$$\rightarrow y \approx 0, inv(y) * y \approx 1$$

instead of

$$\rightarrow inv(y) * y \approx 1$$

yielding the unproblematic (because of a trivial conclusion) inference

$$\frac{\rightarrow 0 * x \approx 0 \quad \rightarrow y \approx 0, inv(y) * y \approx 1}{\rightarrow 0 \approx 0, 0 \approx 1}.$$

Example 4.9 (Superposition Left and Equality Resolution)
These inferences decrease the antecedent (often called *condition* part) of a clause and enumerate solutions for the antecedent. Let $\cup$ be an AC-operator (the *union* operation on sets). Assume the clause

$$\rightarrow x \cup \emptyset \approx x$$

belongs to our specification. We may define an (non-AC) operator $\subseteq$ by

$$x \cup y \approx z \rightarrow x \subseteq z \approx true,$$

and get the following superposition left inference between these clauses:

$$\frac{\rightarrow x \cup \emptyset \approx x \quad x \cup y \approx z \rightarrow x \subseteq z \approx true}{x \approx z \rightarrow x \subseteq z \approx true}$$

The conclusion can be the premise of a equality resolution inference:

$$\frac{x \approx z \rightarrow x \subseteq z \approx true}{\rightarrow x \subseteq x \approx true}$$

This conclusion is an unconditional equation, so we have inferred the reflexivity of $\subseteq$.

Example 4.10 (Equality Factoring)
This inference is defined for non-Horn clauses only, so not implemented in most of the rewrite based completion programs around. As an example consider the totality of a relation $\leq$:

$$\rightarrow x \leq y \approx true, y \leq x \approx true$$

is a premise of an equality factoring inference:

$$\frac{\rightarrow x \leq y \approx true, y \leq x \approx true}{true \approx true \rightarrow x \leq x \approx true}$$

Together with a trivial equality resolution inference (with the conclusion of the previous inference as premise) we again (as in the previous example) have detected the reflexivity of $\leq$.

So far we have defined an inference system adequate to reason with clauses. In the following sections we will use inferences in proofs. We will consider ground inferences only, i.e. inferences in which the premises are ground instances of clauses in a set N. Theorem provers cannot deal with the set of all ground instances of clauses in N, this set is (in general) infinite, so reason with the non-ground clauses themselves (and the inference rules are defined using non-ground clauses). But we will need a correspondence between inferences on the ground level and non-ground inferences. This correspondence is stated in the following lemmata. Note that there are ground inferences with no corresponding non-ground inference. These inferences will be redundant and (by considering reduced ground instances only, see lemma 5.24 and 6.30) we will not encounter non-liftable ground inferences in our proofs.

Lemma 4.11 (Lifting Lemma 1)
Let C be a clause and σ a ground substitution. For any inference

$$\frac{C\sigma}{F},$$

there exist substitutions τ and ρ such that (the unsubstituted clause) C is the premise of an inference

$$\frac{C}{B,}$$

with

(i) $B\rho =_E F$, where all E-equalities apply within subterms s_B of $B\rho$, such that there exists a variable x in C with $x\sigma =_E s_B$,

(ii) and $\tau\rho =_E \sigma$ (hence $C\tau\rho =_E C\sigma$).

Proof: We prove this lemma for the inference *equality resolution*, for *equality factoring* the proof is similar. Suppose $C\sigma = \Gamma\sigma, t_1\sigma \approx t_2\sigma \rightarrow \Delta\sigma$ with a maximal equation $t_1\sigma \approx t_2\sigma$, the terms of this equation are E-equal ($t_1\sigma =_E t_2\sigma$) and $C = \Gamma, t_1 \approx t_2 \rightarrow \Delta$. Then there are substitutions τ and ρ with $\tau \in \mu CSU_E(t_1, t_2)$ and $\tau\rho =_E \sigma$. ρ is the desired ground substitution that satisfies the requirements for σ_1 in definition 4.6 (because all comparisons are E-compatible). Therefore,

$$\frac{C}{\Gamma\tau \rightarrow \Delta\tau}$$

is an inference by equality resolution and $(\Gamma\tau \rightarrow \Delta\tau)\rho =_E \Gamma\sigma \rightarrow \Delta\sigma = F$. So the lemma is proved for $B = \Gamma\tau \rightarrow \Delta\tau$. □

Lemma 4.12 (Lifting Lemma 2)
Let C and D be clauses and σ a ground substitution (in particular, we include the case where $D = t_1 \approx t_2$ is an equation of E). For any inference

$$\frac{C\sigma \quad D\sigma}{F},$$

in which we superpose an equation $s_1\sigma \approx s_2\sigma$ of $C\sigma$ on an equation $t_1\sigma \approx t_2\sigma$ of $D\sigma$ at position p of $t_1\sigma$, where p is not at or below a variable position of t_1, there exist substitutions τ and ρ such that there is an inference

$$\frac{C \quad D}{B}$$

from C and D, and

- $\tau \in \mu CSU_E(s_1, t_1/p)$
- $\tau\rho =_E \sigma$, hence $C\tau\rho =_E C\sigma$ and $D\tau\rho =_E D\sigma$

- $B\rho =_E F$ and all E-steps apply within subterms s_B of $B\rho$, such that there exists a variable x in C or D with $x\tau \neq x$ and $x\sigma =_E s_B$. In other words, the applied instances E-equations contain only terms e with $y\sigma \geq_E e$ (where $y\sigma$ is a maximum (w.r.t. $>_E$) of $\{x\sigma \mid x \in vars(C) \cup vars(D),\ x\tau \neq x\}$).
  If D is an E-equation (in an E-closure inference) and if we additionally assume that no side of an E-equation is simply a variable, then the (terms of the) E-equalities are even smaller (w.r.t. $>_E$) than $t_2\sigma$.

Proof: The proof is similar to the previous lemma. The additional difficulty here is to find a position in the unsubstituted term of the right premise D, when we are given a position p in the substituted $D\sigma$. As p is not at or below a variable in D we obtain an inference

$$\frac{C \quad D}{B}$$

from C and D.

F consists of the terms of $C\sigma$ and $D\sigma$ with the exception of $s_1\sigma$ and $s_2\sigma$. Moreover $(t_1[s_1])\sigma$ is replaced by $(t_1[s_2])\sigma$. Similarly B consists of the terms of $C\tau$ and $D\tau$ except $s_1\tau$ and $s_2\tau$, and with $(t_1[s_1])\tau$ replaced by $(t_1[s_2])\tau$. The terms in F or B also inherit their 'positions' (in a certain equation in the succedent or antecedent) from the corresponding positions in $C\tau$ or $D\tau$. So the "E-difference" between $B\rho$ and F stems from the E-difference of σ and $\tau\rho$. Variables x which do not occur in t_1/p or in s_1, are not substituted by τ. Now we define ρ for these x as $x\rho = x\sigma$. Hence all E-steps apply within subterms $y\tau$ of B with $y\tau \neq y$. B inherits its variables from C and D, therefore the above bound on the size of used instances of E-equations is verified.

For the case where D is an E-equation, we even have a tighter bound for the terms of the instances of E-equations that are used to show $B\rho =_E F$, $C\tau\sigma =_E C\sigma$ and $D\tau\rho =_E D\sigma$: All variables y with $y\tau \neq y$ are involved in the unification and occur as a subterm of s_1 or t_1/p. If we instantiate them by $\tau\rho$, they are E-equal to (not necessarily proper) subterms of $(t_1/p)\tau\rho$ ($=_E s_1\tau\rho$). We have $t_1\sigma >_E (t_1/p)\sigma$ from condition c) in definition 4.6, part 5. Because t_1/p is not below a variable position of t_1, we get $(t_1/p)\tau\rho =_E (t_1/p)\sigma$ (and obviously $t_1\tau\rho =_E t_1\sigma$), so $t_1\tau\rho >_E (t_1/p)\tau\rho$. Subterms of $(t_1/p)\tau\rho$ can not be greater than $(t_1/p)\tau\rho$. So for the variables y instantiated by the unifier τ we have $t_1\tau\rho >_E (t_1/p)\tau\rho \geq_E y\tau\rho$. The instances of E-equations used for $y\tau\rho =_E y\sigma$ are not greater than the terms they prove to be E-equal. Hence they are smaller than $t_1\tau\rho$ (or smaller than the E-equal terms $t_2\tau\rho$, $t_1\sigma$ and $t_2\sigma$).

We need this bound for the proof of lemma 5.30. □

5 The System $\mathcal{M}_E$

The inference system $\mathcal{M}_E$ has already been defined (see definition 4.6). Here we will define an interpretation for sets of ground clauses and use this interpretation to prove that $\mathcal{M}_E$ is indeed an inference system for refutation theorem proving.

5.1 Interpretations for $\mathcal{M}_E$

Definition 5.1 (Ordering over Clauses)
The ordering $>_c$ is the multiset extension of $>_{eq}$ comparing the E-multiset expression of the clauses, where the E-multiset expression of a clause is the multiset of the E-multiset expressions of each occurrence of an equation in that clause.

Lemma 5.2 The following properties hold for $>_c$ and $>_{eq}$.

1) Let C and D be ground clauses with E-equal maximal (w.r.t. to the compatible ordering $>_E$) terms t_C and t_D and $D >_c C$. If a term E-equal to t_C occurs in the antecedent of C, then a term E-equal to t_D (and therefore also E-equal to t_C) occurs in the antecedent of D.

2) Let C be a clause with maximal (w.r.t. $>_E$) term t. If t occurs in the antecedent, no equation in the succedent can be maximal in C (w.r.t. $>_{eq}$).

3) The ordering $>_c$ is E-compatible, i.e. $C >_c D$ implies $C' >_c D'$.

4) The ordering $>_c$ is well-founded.

5) The ordering $>_c$ is total on E-congruence classes of clauses, i.e. $C \neq_E D$ implies either $C >_c D$ or $D >_c C$.

Proof: Part 1) and 2) hold because of the additional element $\bot_E$ in the E-multiset expression for equations in the antecedent.

3) The E-multiset expressions are identical if and only if the considered clauses are E-equal. From $C >_c D$ we know that the corresponding E-multisets are not equal. But the E-multiset expression of C' is equal to that of C and the same holds for D' and D, hence $C' >_c D'$.

4) The ordering is based on multiset extensions of well-founded orderings.

5) The ordering is based on comparisons of congruence classes of terms. These comparisons are done by a total ordering $>_E$, so $>_c$ is also total (multiset extensions of total orderings are total over finite multisets: Let C and D be multisets and M denote the multiset intersection of C and D; now consider $C \setminus M$ and $D \setminus M$: at least one of them is not empty, if C and D are not equal; if exactly one of them is empty, the corresponding set is smaller; otherwise compare a maximal element of $C \setminus M$ and $D \setminus M$: they cannot be equal (then they would be eliminated by M) so one of them is greater, because of the totality of the underlying ordering, and the multiset with the greater maximal element is the greater one). □

Lemma 5.3 Let

$$\frac{C \quad D}{B} \qquad \text{resp.} \qquad \frac{D}{B}$$

be a ground inference of $\mathcal{M}_E$. Then we have $D >_c B$ and $D \neq_E B$.

Proof: The lemma is trivial for equality resolution inferences, because an occurrence of an equation in D is eliminated (yielding B). For other inferences observe that one occurrence of an equation is replaced by a finite multiset of smaller (and not E-equal) occurrences of equations (smaller w.r.t. $>_{eq}$). Hence by definition of $>_c$ (based on $>_{eq}$), we have $D >_c B$. □

An *equality Herbrand interpretation* is a congruence on ground terms. We now define such an interpretation for sets NG of ground instances of clauses in N.

We assume the reader familiar with construction of interpretation for sets of clauses, in particular the constructions in [Bachmair/Ganzinger 91a], [Bachmair/Ganzinger 90] and [Nieuwenhuis 91]. Our construction will share an important property with the interpretation technique in these papers:

> The interpretations are represented by ground rewrite systems which are Church-Rosser by construction.

The rewrite system consists of rules produced by maximal equations of ground instances of clauses. The definition of the interpretation uses induction over the size of ground instances of clauses. This way the rule with the smallest left side is produced first, then rules are produced with increasing left sides. To each ground instance $C\sigma$ of a clause a rewrite system $R_{C\sigma}$ is assigned consisting of all rules produced by smaller instances of clauses. Different to the case for $E = \emptyset$, we construct interpretations which have to satisfy not only the clauses in a given set N, but also the equations of E. Therefore our interpretation consists of a set of ground rules and a set of ground instances of E-equations, so to each clause $C\sigma$ a rule system $R_{C\sigma}$ and a set $E_{C\sigma}$ of ground instances of E-equations are assigned. Our aim is to choose the set of rules $R_{C\sigma}$ and the set of equations $E_{C\sigma}$ such that $\Longrightarrow_{R_{C\sigma}}$ is Church-Rosser modulo $E_{C\sigma}$ for $\equiv_{R_{C\sigma} \cup E_{C\sigma}}$. The confluence is achieved by producing ground rules with irreducible left sides. For coherence we distinguish two cases:

- Coherence with $E_{C\sigma}$-equations applied *at or below* the left side of a rule: A rewrite relation with E-matching solves this problem. We approximate such a relation by a set of rules with E-equal left side (see definition 5.5). For more details see the proof of lemma 5.16.
- Coherence with $E_{C\sigma}$-equations applied *above* of the left side of a rule: We exclude such instances of E-equations from $E_{C\sigma}$ (see definition 5.8).

With this construction the proof methods of [Bachmair/Ganzinger 91a] and [Nieuwenhuis 91] can be adopted. The main difference to their proof is stated by lemma 5.30: we have to show that an interpretation satisfies sufficiently many ground instances of E-equations.

We might think of another method for defining interpretations which by construction satisfy the E-equations: we can define an interpretation such that each clause produces at most one rule and the rule is applied in a rewrite relation modulo E. Then the rule systems are not Church-Rosser in general (even if left sides of rules do not overlap, see example 3.5). Because R_C is infinite in general, it is not known if for any R_C there is a canonical system, even if we have $E = AC$ (for finite system this question was solved by [Narendran/ Rusinowitch 91] and [Marché 91]). But even if there is a Church-Rosser system for those R_C, we have to add equalities between terms arbitrarily smaller than the maximal term of the clause C. So properties similar to properties in the lemmata following the definition of interpretations cannot be proved for such a construction.

Definition 5.4 Let C be a ground clause. Let E_C^{all} be the set

$$E_C^{all} = \{\ e_1\sigma \approx e_2\sigma \ \mid\ \sigma \text{ ground substitution, } e_1 \approx e_2 \in E,\ C >_c \rightarrow e_1\sigma \approx e_2\sigma\ \}\ .$$

In the following definition we use an ordering $>_t$ over terms which is total on ground terms. Such an ordering $>_t$ always exists.

Definition 5.5 (Irreducible Sets of Rules)
Let R be a rewrite system and $\ell \approx r$ be a ground equation with $\ell >_E r$. Let $>_t$ be a total and well-founded ordering on ground terms. Let r_{min} denote the minimal (w.r.t. the total ordering $>_t$) term E-equal to r, i.e.

$$r_{min} := min_{>_t}\{r' \mid r' =_E r\}.$$

First we define a set of rules, where the left sides are E-equal to ℓ and irreducible by R:

$$rules(\ell \approx r, R) := \{\ell' \Rightarrow r_{min} \mid \ell' =_E \ell,\ \ell' \text{ irreducible by } \Longrightarrow_R\}$$

Then we exclude rules (whose left side is E-equal to ℓ) which are reducible (on their left side) by other rules with an E-equal left side:

$$irred\_rules(\ell \approx r, R) := \{\ \ell' \Rightarrow r \ \mid\ \ell' \Rightarrow r_{min} \in rules(\ell \approx r, R),\ \ell' \text{ irreducible by } \Longrightarrow_{(rules(\ell\approx r,R)\setminus\{\ell'\Rightarrow r\})}\ \}$$

In the following definition of an interpretation we want to define sets of rules which are confluent by construction, therefore we only use irreducible rules. Because we add more than one rule at a time, a rule must not reduce any of the other rules introduced at the same time (and so in general we need the second of the above sets of rules). For certain theories E (where no term is E-equal to a proper subterm of itself, e.g. $E = AC$) the two sets of rules are identical.

Due to the definition of r_{min}, E-equal clauses will produce the same set of rules (above the set of rules does not depend on the E-representative of ℓ or r). We will need this only in lemma 5.16, therefore we will drop the index min in other parts of this paper (if we consider saturated sets, then the interpretation will always satisfy $r \approx r_{min}$, hence it is of no interest for saturated sets which right side is used).

Definition 5.6 (Closure)
Let R be a ground rewrite system and Eq be a set of ground equations.

$$closure(R, Eq) := (\Longrightarrow_R \cup \Longleftrightarrow_{Eq})^*,$$

i.e. by $closure(R, Eq)$ we mean the reflexive, transitive and symmetric closure of $\Longrightarrow_R \cup \Longleftrightarrow_{Eq}$.

Definition 5.7 Let N be a set of (not necessarily ground) clauses. We define:

$$NG := \{C\sigma \mid C \in N, \sigma \text{ ground substitution}\} \cup EG \cup TG \cup \{TOP\}$$

$$NG_C := \{D \mid D \in NG, C >_c D\}$$

where

$$EG := \{(e_1 \approx e_2)\sigma \mid e_1 \approx e_2 \in E, \sigma \text{ ground substitution}\},$$
$$TG := \{t \approx t \rightarrow t \approx t \mid t \text{ ground term}\}$$

and

$$TOP := \rightarrow \top \approx \top,$$

where $\top$ is a new symbol with $\top >_E t$ for all ground terms t.

Note that the clause TOP is the strictly maximal clause of NG and satisfied in any interpretation. The E-equations and TG-clauses do not contribute to the interpretation of NG. Nevertheless we include them here as additional clauses, so we can speak of an interpretation I_{EE}, for $EE \in EG$ or I_{TT}, for $TT \in TG$. Note that a clause $t \approx t \rightarrow t \approx t$ of TG is greater than any clause which contains a term E-equal to t and which can possibly introduce new rules into the interpretation defined below. Such a clause of TG is also greater than (and not E-equal to) all E-equations containing terms E-equal to t.

Definition 5.8 (Interpretation)
Let N be a set of clauses and NG be the set of ground instances of clauses as defined above. Let $C\sigma$ be a ground instance of a clause C in N, i.e. $C\sigma \in NG$. We assume that $Rules_D$, R_D, E_D and I_D have been defined for all ground instances D in NG with $C\sigma >_c D$. We define

$$R_{C\sigma} = \left(\bigcup_{C\sigma >_c D} Rules_D \right)$$

$$E_{C\sigma} = \{e_1 \approx e_2 \in E^{all}_{C\sigma} \mid e_1 \text{ and } e_2 \text{ are both irreducible w.r.t. } \Longrightarrow_{R_{C\sigma}}\}$$
$$I_{C\sigma} = closure(R_{C\sigma}, E_{C\sigma})$$

and

$$Rules_{C\sigma} = irred\_rules(t\sigma \approx s\sigma, R_{C\sigma})$$

if all of the following conditions hold, otherwise $Rules_{C\sigma} = \emptyset$.

1. $C\sigma = \Gamma\sigma \rightarrow \Delta\sigma, t\sigma \approx s\sigma$
2. $I_{C\sigma} \not\models \Gamma\sigma \rightarrow \Delta\sigma'$ for all σ' with $\sigma' =_E \sigma$
3. $t\sigma \approx s\sigma$ is strictly maximal (w.r.t. $>_{eq}$) in $C\sigma$, $t\sigma >_E s\sigma$
4. $t\sigma'$ is not reducible by $R_{C\sigma}$ for all $\sigma' =_E \sigma$ (we do not require that each term $(t\sigma)'$ is irreducible).

For other ground clauses B in NG we define the sets R_B and E_B as above and $Rules_B := \emptyset$.

Note that $C >_c D$ implies $R_C \supseteq R_D$ and R_{TOP} is the limit of all rewrite systems (note that TOP itself does not produce a rule):

$$R_{TOP} = \bigcup_{C\sigma \in NG} Rules_{C\sigma} = \bigcup_{C\sigma \in NG} R_{C\sigma}$$

As we have $E^{all}_{TOP} = EG$, we can define:

$$E_{TOP} = \{e_1 \approx e_2 \in EG \mid e_1 \text{ and } e_2 \text{ are both irreducible w.r.t. } \Longrightarrow_{R_{TOP}}\}$$

Therefore we regard I_{TOP} as an interpretation for the whole specification $N \cup E$ and each I_C (for $C \neq TOP$) as a *partial* interpretation.

Example 5.9 Let $+$ be an AC-operator. Assume an AC-compatible ordering based on a recursive path ordering with decreasing precedence for a, b, c, d and $+$. We first consider the interpretation of a set of clauses consisting of a single ground clause

$$C = \rightarrow (a+b)+c \approx b+c.$$

The set E_C consists of some AC-equalities containing only terms smaller than $(a+b)+c$, i.e.

$$\begin{aligned} E_C \;=\; & \{b+c \approx c+b, (b+c)+d \approx b+(c+d), \\ & c+d \approx d+c, b+(b+b) \approx (b+b)+b, \ldots\}. \end{aligned}$$

As C is the smallest clause, we have $R_C = \emptyset$. We have

$$rules((a+b)+c \approx b+c, R_C) = irred\_rules((a+b)+c \approx b+c, R_C) =$$

$$\{(a+b)+c \Rightarrow b+c, a+(b+c) \Rightarrow b+c, (b+a)+c \Rightarrow b+c, b+(a+c) \Rightarrow b+c, \ldots\}.$$

Note that all terms AC-equal to $(a+b)+c$ occur as left sides of rules, but the right side of all rules is $b+c$ (e.g. there is no rule with right side $c+b$; more precisely: due to the definition of r_{min} in 5.5, we will get always the same right side; it does not matter, whether it is $b+c$ or $c+b$). There is no other clause, so $R_{TOP} = R_C$. The set E_{TOP} contains instances of AC-equation between terms not containing a subterm which is AC-equal to $(a+b)+c$, e.g. E_{TOP} contains $(a+b)+(d+c) \approx a+(b+(d+c))$ (an instance of the associativity law), but not $((a+b)+c)+d \approx (a+b)+(c+d)$, because the left side of this equation is reducible. There are instances of AC-equations which are not contained in E_{TOP}, but satisfied in I_{TOP}, e.g. $(a+b)+c \approx a+(b+c)$ is satisfied by reducing both sides to $b+c$. But the (above mentioned) equation $((a+b)+c)+d \approx (a+b)+(c+d)$ is neither contained in E_{TOP} nor in I_{TOP}. This is a disadvantage for sets of clauses which are not saturated.

Now consider a superset of $\{C\}$ which is $\mathcal{M}_E$-saturated. The missing clauses are conclusions of E-closure inferences, e.g

$$\frac{\rightarrow (a+b)+c \approx b+c \quad \rightarrow ((a+b)+c)+x \approx (a+b)+(c+x)}{\rightarrow (a+b)+(c+x) \approx (b+c)+x}$$

Let us denote the conclusion of the above inference by C_{AC} (that only finitely many such clauses are sufficient to saturate a set (for $E = AC$) is explained in section 5.3). Ground instances of such clauses C_{AC} produce rules such that all equations in $EG \setminus E_C$ become reducible, i.e. $E_{TOP} = E_C$. But nevertheless, all equations in EG are satisfied in I_{TOP}. As an example we again consider $((a+b)+c)+d \approx (a+b)+(c+d)$. Let D denote the clause

$$\rightarrow (a+b)+(c+d) \approx (b+c)+d$$

which is an instance of C_{AC}. D produces the following set of rules:

$$\{(a+b)+(c+d) \Rightarrow (b+c)+d, (a+d)+(b+c) \Rightarrow (b+c)+d, \ldots\}$$

Note that D does not produce rules which contain subterms AC-equal to $(a+b)+c$. Now by the first rule in this set and a rule produced by C we can reduce both sides of $((a+b)+c)+d \approx (a+b)+(c+d)$ to $(b+c)+d$. In general the situation is more complicated: the two sides of the equation may be reducible to AC-equal terms (which are not equal without AC) and there may be another clause which is smaller than D and producing a rule $(a+b)+(c+d) \Rightarrow r$. For details see lemma 5.30.

Lemma 5.10 (Independence from E-Representatives)
For all clauses C and ground substitutions σ and σ' (with $\sigma' =_E \sigma$), we have $I_{C\sigma} = I_{C\sigma'}$. If there exists another ground clause $D \in NG$ with $D =_E C\sigma$, then we also have $I_D = I_{C\sigma}$.

Proof: Follows from the E-compatibility of $>_c$ used in definition 5.8. □

We will use the previous lemma in 5.29. If a ground inference with premise $C\sigma$ is lifted and then instantiated to an E-equal inference, we might get an inference with premise $C\sigma'$. We do not want that the change from σ to σ' influences the redundancy of such an inference (at least not for the inferences considered in lemma 5.29). So we require $I_{C\sigma} = I_{C\sigma'}$.

Lemma 5.11 (Termination of $\Longrightarrow_{R_C}$)
The rewrite relations $\Longrightarrow_{R_C}$ and $\Longrightarrow_{R_C/E}$ are terminating for all $C \in NG$, in particular for $C = TOP$.

Proof: Every rule produced by a clause of NG is contained in $>_E$. □

Definition 5.12 (Productive Clause)
If for a clause C we have $Rules_C \neq \emptyset$, then we call C a *productive clause.*

Lemma 5.13 Let C be a ground clause in NG with maximal term t. Every rule produced by a clause G greater than C has a left side ℓ with $\ell \geq_E t$. If C is productive or greater than $\rightarrow t \approx t$ and G is greater than C, we even have $\ell >_E t$.

Proof: The left side of a rule produced by a clause D is a maximal (w.r.t. $>_E$) term of D. So if D produces a rule with left side ℓ and $t >_E \ell$ we have $C >_c D$. Hence greater clauses can only produce rules $\ell \Rightarrow r$ with $\ell \geq_E t$.

If C is productive, then all terms E-equal to t are reducible in any interpretation of a clause greater than C and no greater clause with maximal term E-equal to t fulfills condition 4 of definition 5.8. If C is greater than $\rightarrow t \approx t$, no productive clause greater than C can have a maximal equation $\ell \approx r$ in the succedent with $\ell >_E r$ and $t \geq_E \ell$. So in both cases all rules produced later have left sides greater than t. □

Note that for the above lemma C is greater than $\rightarrow t \approx t$, if a term E-equal to t occurs in the antecedent of C.

Lemma 5.14 Let C be a ground clause of NG with maximal (w.r.t. $>_E$) term t. Let D be a ground clause in NG such that $D >_c C$.

A term u with $t >_E u$ is reducible w.r.t. $\Longrightarrow_{R_D}$ if and only if u is already reducible w.r.t. $\Longrightarrow_{R_C}$.

An equation $e_1 \approx e_2$ in E_C^{all} is contained in E_D if and only if it is already contained in E_C.

Proof: The first part follows from lemma 5.13 and the trivial fact that the left side of the rule which reduces u cannot be greater than u.

If an equation $e_1 \approx e_2$ of E_C^{all} is not contained in E_C, e_1 or e_2 is reducible by $\Longrightarrow_{R_C}$ and it will remain reducible in R_D as $R_C \subseteq R_D$, and hence it is not in E_D.

If an equation $e_1 \approx e_2$ of E_C^{all} is contained in E_C, e_1 and e_2 are irreducible w.r.t. $\Longrightarrow_{R_C}$ and they will remain irreducible in R_D. If $t >_E e_1$ this follows from the first part of this lemma. If $t =_E e_1$ then C is greater than $\rightarrow t \approx t$ (see definition 5.4) so that C and any clause greater than C cannot produce a rule reducing a term E-equal to t (see lemma 5.13). □

Lemma 5.15 (Monotonicity of Interpretations)
Let C and D be ground clauses with $C >_c D$. Then we have $I_C \supseteq I_D$.

Proof: By construction we have $R_C \supseteq R_D$, and by the previous lemma all equations in E_D are also contained in E_C. □

Lemma 5.16 (Church-Rosser Property of $\Longrightarrow_{R_C}$)
For every ground clause $C \in NG$ we have:

1. A term t is reducible by $\Longrightarrow_{R_C}$ if and only if it is reducible by $\Longrightarrow_{R_C \cdot E}$ (but not necessarily to the same term).
2. There are no overlaps between rules in R_C, so $\Longrightarrow_{R_C}$ is confluent (hence confluent modulo E and modulo E_C).
3. The relation $\Longrightarrow_{R_C}$ is Church-Rosser modulo E_C for the congruence $\equiv_{R_C \cup E_C}$. (But note that for sets N of clauses which are not $\mathcal{M}_E$-saturated we have $E_C \neq E_C^{all}$ in general).

Proof: The first and second property follows from the construction of R_C (in particular from definition 5.5). With the second property and the termination (lemma 5.11) we can apply lemma 3.9 so that it remains to show the local ground coherence modulo E_C to prove the third part. We first consider a peak

$$\ell[e_1] \Longleftrightarrow_{E_C} \ell[e_2] \Longrightarrow_{R_C} r_{min},$$

where the rewrite step is an application of a rule $\ell \Rightarrow r_{min}$ and the applied E_C-equation is $e_1 \approx e_2$. It is sufficient to consider such peaks, where the application of the equation is strictly below the application of the rule, other peaks are not possible because E_C-equations are irreducible by R_C. $\ell[e_1]$ cannot be irreducible (by construction of the interpretation; note that $\ell[e_2]$ is reduced at the root and use part 1 of this lemma). If it is reducible at the root, then only by a rule $\ell[e_1] \Rightarrow r_{min}$ produced together with $\ell \Rightarrow r_{min}$ or by an E-equal clause, hence the right side of the rule is always r_{min} and we are done, because also $\ell[e_1]$ is reducible to r_{min}. Otherwise it is reducible strictly below the root and strictly above the application position of $e_1 \approx e_2$ (because the equation is irreducible). We denote the rule reducing ℓ strictly below the root by $t \Rightarrow s$. But if $t[e_1]$ is reducible (at the root) then also $t[e_2]$ is reducible (see part 1 above). But $t[e_1]$ is a subterm of $\ell[e_1]$ and so $t[e_2]$ a subterm of $\ell[e_2] = \ell$ and ℓ is reducible, which is a contradiction.

For peaks which differ from the previous one only by an additional context u, i.e. $u[\ell[e_2]]$ is reducible to $u[r_{min}]$, we use the same arguments as above.

As E-equations are irreducible it remains to consider a peak where the E_C-equation is applied at a subterm disjoint to the application position of the rule:

$$u[e_2][\ell] =_{E_C} u[e_1][\ell] \Longrightarrow_{R_C} u[e_1][r_{min}]$$

But here we have

$$u[e_2][\ell] \Longrightarrow_{R_C} u[e_2][r_{min}] =_{E_C} u[e_1][r_{min}].$$

So all peaks for local coherence are joinable. Note that for general E instead of E_C there are more peaks to consider. Peaks with an application of a rule below an equation are excluded here by the irreducibility of the equations. Because of the greater number of different peaks

for the general case we have formulated lemma 3.9 assuming local coherence instead of the joinability of peaks between rules and equations (in this situation often called *coherence pairs*). □

Lemma 5.17 An equation $u \approx v$ is satisfied in an interpretation I_C if and only if the terms u and v are joinable modulo E_C using the rewrite relation $\Longrightarrow_{R_C}$:

$$I_C \models u \approx v \qquad \text{if and only if} \qquad u \Downarrow_{R_C} v$$

Proof: Follows from the Church-Rosser property stated in the previous lemma. □

Lemma 5.18 Let C be a clause of NG with maximal term t and D a ground clause greater than C.

An equation $u \approx v$ with $t >_E u$ and $t >_E v$ is true in I_D if and only if it is true in I_C.

If a term E-equal to t occurs in the antecedent of C, then an equation $u \approx v$ with $t \geq_E u$ and $t \geq_E v$ is true in I_D if and only if it is true in I_C.

Proof: Follows from Church-Rosser property (lemma 5.16) and lemmata 5.13 and 5.14. For the second part we notice that clauses greater than C cannot produce a rule which reduces u or v or an instance of an E-equation applicable at u or v. □

We use this lemma to show that increasing interpretations preserve the truth of ground instances of clauses:

Lemma 5.19 Let B, C and D be ground clauses with $D >_c C$ and $C >_c B$ or $C =_E B$. If B is true in I_C then it is true in I_D.

Proof: If an equation in the succedent is satisfied apply lemma 5.15. Otherwise an equation in the antecedent is not satisfied and remains so by the previous lemma. □

We have $E_C \subseteq E_C^{all}$, in particular $E_{TOP} \subseteq EG$ (where EG is the set of all ground instances of equations in E), and the inclusion is (in general) proper. Because we do not have $E_C^{all} \subseteq I_C$, the following is **not true** in general:

$$I_C \models u \approx v \qquad \text{if and only if} \qquad I_C \models u' \approx v'$$

But in the next section (with one more restriction on E) we prove that for interpretations of $\mathcal{M}_E$-saturated sets we have $E_C^{all} \subseteq I_C$, in particular $EG \subseteq I_{TOP}$ and (for $\mathcal{M}_E$-saturated sets)

$$I_C \models u \approx v \qquad \text{if and only if} \qquad I_C \models u' \approx v', u' =_{E_C^{all}} u, v' =_{E_C^{all}} v$$

$$I_{TOP} \models u \approx v \qquad \text{if and only if} \qquad I_{TOP} \models u' \approx v', u' =_E u, v' =_E v.$$

5.2 Redundancy-Completeness of $\mathcal{M}_E$

The following notions are defined for the system $\mathcal{M}_E$. But they can be applied to systems similar to $\mathcal{M}_E$, too (cf. system $\mathcal{MM}_E$, section 8.1). The definitions will be modified in section 6.3 for $\mathcal{M}_{Ext}$. The notation follows [Nieuwenhuis 91] and [Bachmair/Ganzinger 91c] but we extend the notions by the use of the equational theory E.

Definition 5.20 (Redundancy of Clauses)
A ground clause C is *E-redundant* (or for short *redundant*) in a set of clauses N, if it is satisfied in its partial interpretation, i.e $I_C \models C$.

A non-ground clause C is E-redundant (in a set N) if every ground instance $C\sigma$ of it is E-redundant (in N), i.e. $I_{C\sigma} \models C\sigma$ for all ground substitutions σ.

Definition 5.21 (Redundancy of Inferences)
An inference π from ground instances $C_1, \ldots, C_n$ of clauses in $N \cup E$ and conclusion D is *E-redundant* (or for short: *redundant*) in N, if one of its premises is redundant or if the conclusion is satisfied in I_{C_j}, i.e. $I_{C_j} \models D$, where C_j is the maximal premise of the inference π.

A (non-ground) inference π from N is redundant (more precisely E-redundant in N) if all its ground instances $\pi\sigma$ are redundant.

Definition 5.22 (Saturation and Completeness)

1. Let π be an inference of $\mathcal{M}_E$ with premises $C_1, \ldots, C_n$ and conclusion D. Every inference of $\mathcal{M}_E$ with premises $C_1\sigma, \ldots, C_n\sigma$ and conclusion $D\sigma$ for a ground substitution σ is called a *ground instance* $\pi\sigma$ of π. Note that there are ground substitutions σ such that there is no inference with premises $C_i\sigma$ and conclusion $D\sigma$.
2. Let MG be a set of ground clauses. The set N of clauses is *$\mathcal{M}_E$-saturated on MG*, if every ground inference π with premises in MG is E-redundant in N.
3. A set N of clauses is *$\mathcal{M}_E$-saturated*, if every inference (of $\mathcal{M}_E$) with premises in $N \cup E$ is E-redundant in N.
4. An inference system $\mathcal{M}_E$ is *redundancy-complete*, if for every $\mathcal{M}_E$-saturated set N, N contains the empty clause, if $N \cup E$ is inconsistent.

We will now start to prove the redundancy-completeness of $\mathcal{M}_E$. Hence the inference system $\mathcal{M}_E$ will be refutation complete, even with the above redundancy built in. Note that the above redundancy for clauses and inferences is adequate for the completeness proof only. But the redundancy will lead to a general method of describing clauses and inferences irrelevant during theorem proving (see section 7). For the completeness proof we need one assumption about E:

Definition 5.23 (Restrictions on E)
In this section we consider only equational theories E, where no term is E-equal to a strict subterm of itself.

We will need the restriction in lemmata 5.25, 5.29 (part iv) and 5.30 (case B2.2.2). This restriction together with the preservation of variables (see definition 4.1) ensures that no

side of an equation is simply a variable (w.o.l.o.g. we assume $x \approx x \notin E$). We use this in lemma 4.12, which is needed to prove lemma 5.30. The most popular example which meets this restriction is $E = AC$.

Lemma 5.24 Let N be a set of clauses. Let D be a clause, σ a ground substitution and C a ground clause in NG. If there exists a variable $x \in vars(D)$ such that $x\sigma$ is reducible by R_C, then there exists a ground substitution σ_1 such that $D\sigma >_c D\sigma_1$ and $I_C \models D\sigma_1$ if and only if $I_C \models D\sigma$.

Proof: If $x\sigma \Longrightarrow_{R_C} t$, we define σ_1 to be the substitution for which $x\sigma_1 = t$ and $y\sigma_1 = y\sigma$, for all $y \neq x$. We have $I_C \models x\sigma \approx t$, and so $D\sigma_1$ is true in I_C if and only if $D\sigma$ is true in I_C. □

With restrictions 5.23 we can prove the following lemma:

Lemma 5.25 (Properties of Productive Clauses 1)
Let $G := \Gamma \rightarrow \Delta, t \approx s$ be a clause of NG with strictly maximal equation $t \approx s$, $t >_E s$ and let G be productive. Let C be a clause greater than G. If I_C satisfies all instances of E-equations between terms smaller than t, then $I_C \models t \approx s$, i.e. the productive equation will be satisfied in interpretations of greater clauses.

Proof: t is irreducible by R_G (required for productive clauses by part 4 of definition 5.8). Every t' E-equal to t is reducible in R_G enhanced with the produced rules in $irred\_rules(t \approx s, R_G)$ (note: these rules are contained in I_C). Hence there is a rule $t'' \Rightarrow s_{min}$ with $t = u[t'']$ and t is reduced to $u[s_{min}]$. If u is the empty context we have $t = t''$, $u[s_{min}] = s_{min}$ and are done, because by our assumption I_C satisfies $s \approx s_{min}$. Otherwise $t = u[t''] =_E u[t]$, hence t is E-equal to a proper subterm of itself. This contradicts definition 5.23, so we have $u = []$ and a rule $t \Rightarrow s_{min}$ is produced. □

Similarly to the previous lemma we can prove $rules(t \approx s, R_G) = irred\_rules(t \approx s, R_G)$ using definition 5.23. But we cannot guarantee that $I_C \models t' \approx s$ (for all $t' =_E t$), because t' may be reducible by a rule in R_G to a term different from s_{min} (this is a difference to the next section, cf. lemma 6.39):

Example 5.26 Let $+$ be an AC-operator. The clause

$$\rightarrow a + (b + c) \approx d$$

may produce a rule $a + (b + c) \Rightarrow d$, even if the AC-equal left side $(a + b) + c$ is reducible (e.g. by a rule $a + b \Rightarrow e$). Then the equation $(a + b) + c \approx d$ might not be satsified. In lemma 6.39 we know that $(a + b) + c$ is irreducible and can conclude that a rule to satisfy $(a + b) + c \approx d$ is produced.

Lemma 5.27 (Properties of Productive Clauses 2)
Let $G := \Gamma \rightarrow \Delta, \ell \approx r$ be a ground clause in NG. If G is a productive clause, it is not satisfied in its interpretation, i.e. $I_G \not\models G$ (which means that G is non-redundant).

Proof: If G is productive, we have $I_G \not\models \Gamma \rightarrow \Delta$. So G is only true in I_G if $\ell \approx r$ is satisfied. If G is productive, we have $\ell >_E r$ and so (because the interpretation is a Church-Rosser rewrite system, cf. lemma 5.16) ℓ has to be reducible, which contradicts part 4 in definition 5.8 for productive clauses. □

Lemma 5.28 Let $C := \Gamma \rightarrow \Delta, t \approx s$ be a ground clause and $t \approx s$ a maximal equation of C with $t >_E s$. Let D be another clause containing a term t'. If $C >_c D$ and t is irreducible by $\Longrightarrow_{R_C}$, then $R_C = R_D$ and $I_C = I_D$.

Proof: Clauses G with $C >_c G \geq_c D$ can only produce rules in a way such that all terms E-equal to t become reducible (all these clauses contain a maximal term E-equal to t). As t is irreducible by $\Longrightarrow_{R_C}$, no rule is produced and so we have $R_C = R_D$. All E-equations smaller than C are also smaller than D, so $E_C^{all} = E_D^{all}$. As these clauses define the same rewrite system, we have also $E_C = E_D$ and so $I_C = I_D$. □

Lemma 5.29 (Properties of Non-Redundant Clauses)
Let $F\sigma := \Gamma\sigma \rightarrow \Delta\sigma, t\sigma \approx s\sigma$ be a non-redundant ground instance of a clause in N and and assume that for all substitutions σ' with $\sigma' =_E \sigma$ we have $I_{F\sigma} \not\models \Gamma\sigma \rightarrow \Delta\sigma'$. Let the equation $t\sigma \approx s\sigma$ be maximal (w.r.t. $>_{eq}$) in $F\sigma$ with $t\sigma >_E s\sigma$, and let $t\sigma'$ be irreducible by $\Longrightarrow_{R_{F\sigma}}$ (for all $\sigma' =_E \sigma$). If N is $\mathcal{M}_E$-saturated on $NG_{F\sigma} \cup \{F\sigma' \mid \sigma' =_E \sigma\}$ and if $I_{F\sigma}$ satisfies all ground instances of E-equations between terms smaller than $t\sigma$, the following holds:

(i) σ is irreducible (w.r.t. $\Longrightarrow_{R_{F\sigma}}$), i.e. $x\sigma$ is irreducible for all $x \in vars(F)$

(ii) $F\sigma$ is productive.

(iii) For all $(\Gamma\sigma)'$ with $(\Gamma\sigma)' =_E \Gamma\sigma$ and all ground clauses $C >_c F\sigma$ or $C =_E F\sigma$,

$$(\Gamma\sigma)' \subseteq I_C \ .$$

(iv) For all $\Delta\sigma'$ with $\sigma' =_E \sigma$ and all ground clauses $C >_c F\sigma$ or $C =_E F\sigma$,

$$\Delta\sigma' \cap I_C = \emptyset \ .$$

Proof: The proof is by induction on the ordering $>_c$, so let us assume that (i)–(iv) hold for every suitable instance F_1 of N with $F\sigma >_c F_1$. Since $F\sigma$ is non-redundant we have $\Gamma\sigma \subseteq I_{F\sigma}$ and $\Delta\sigma \cap I_{F\sigma} = \emptyset$.

(iii) $\Gamma\sigma \subseteq I_{F\sigma}$ and every term in $\Gamma\sigma$ is smaller than $t\sigma$ (otherwise the equation $t\sigma \approx s\sigma$ is not maximal). So the instances of E-equations needed to show $\Gamma\sigma =_E (\Gamma\sigma')$ are satisfied by $I_{F\sigma}$ (by our assumptions) and we have $(\Gamma\sigma)' \subseteq I_{F\sigma}$. We have $I_C \supseteq I_{F\sigma}$ (lemmata 5.15 and 5.10), hence I_C, too, satisfies every equation in $(\Gamma\sigma)'$.

(i) Suppose there is a variable $x \in vars(F)$ such that $x\sigma$ is reducible by $\Longrightarrow_{R_{F\sigma}}$ to a term r (note that x cannot occur in t, as $t\sigma$ is irreducible by our assumptions). We define a substitution τ by $y\tau = y\sigma$ for all $y \neq x$ and $x\tau = r$. $F_1 := F\tau$ is false in $I_{F\sigma}$. We also have $F\sigma >_c F_1$. By lemma 5.28 we have $I_{F_1} \not\models F_1$, hence F_1 is non-redundant, and the maximal term of the maximal equation of F_1 is irreducible (under all substitutions τ'). Also $I_{F_1} \cap \Delta\tau' = \emptyset$ (for all $\tau' =_E \tau$), for otherwise it would contradict $I_{F\sigma} \cap \Delta\sigma' = \emptyset$, and $\Gamma\tau' \subseteq I_{F_1}$, for otherwise it would contradict $\Gamma\sigma' \subseteq I_{F\sigma}$ (by (iii)). Now we use our induction hypothesis and infer that F_1 produces a rule to reduce $t\sigma$ (induction hypothesis part (ii)). But this rule is contained in $R_{F\sigma}$ contradicting the irreducibility of $t\sigma$. So $F\sigma$ is an instance of N with a reduced substitution (and we assume this fact in the remaining parts of the proof).

(ii) If $F\sigma$ is not productive, then an equation $t_2\sigma \approx s_2\sigma$ E-equal to (but different from) the maximal equation $t\sigma \approx s\sigma$ of $F\sigma$ occurs in the succedent of $F\sigma$. In other words, the

equation $t\sigma \approx s\sigma$ is not strictly maximal and the clause is not productive for this reason. Moreover $I_{F\sigma} \models s\sigma \approx s_2\sigma$ and $I_{F\sigma} \models s\sigma' \approx s_2\sigma'$. $t\sigma >_E s\sigma$ and the needed E-equalities are by assumption contained in $I_{F\sigma}$. A clause $F_1 = \Gamma\sigma', s\sigma' \approx s_2\sigma' \rightarrow \Delta\sigma'$ smaller than $F\sigma$ can be obtained as conclusion of an instance of an equality factoring inference with premise $\Gamma \rightarrow \Delta, t \approx s$. This clause is false in $I_{F\sigma} = I_{F\sigma'}$. This contradicts the required saturation. So from now on we may assume that no other equation E-equal to $t\sigma \approx s\sigma$ occurs in the succedent of $F\sigma$ and hence that $F\sigma$ is productive.

(iv) $I_{F\sigma} \not\models \Gamma\sigma \rightarrow \Delta\sigma$, so $I_{F\sigma} \cap \Delta\sigma = \emptyset$. By our assumptions we also have $I_{F\sigma} \cap \Delta\sigma' = \emptyset$ and so by lemma 6.18 also $I_{(F\sigma)'} \cap \Delta\sigma' = \emptyset$.

Suppose an $\Delta\sigma'$ contains an equation $u\sigma' \approx v\sigma'$ which is satisfied by an I_C (with $C >_c F\sigma$). We may assume that $u\sigma >_E v\sigma$, for otherwise, if $u\sigma =_E v\sigma$, then $t\sigma >_E u\sigma$ (if $t\sigma =_E u\sigma$ the equation $t\sigma \approx s\sigma$ were not maximal in $F\sigma$) and $u\sigma' \approx v\sigma'$ is also satisfied by $I_{F\sigma}$ (by the assumption the needed E-equalities are available). By construction all interpretations are Church-Rosser systems, so $u\sigma'$ and $v\sigma'$ are reducible by R_C to $E_{F\sigma}$-equal terms (the terms cannot grow by reduction, so the needed instances of E-equations from E_C are also satisfied in $I_{F\sigma}$). The equation is not satisfied by $I_{F\sigma}$, so $u\sigma'$ is reduced with a rule of $R_C \setminus R_{F\sigma}$, hence the left side of the reducing rule is E-equal to $t\sigma$ or even greater. As $\Delta\sigma'$ does not contain a term greater (w.r.t. $>_E$) than $t\sigma$, we have $u\sigma' =_E t\sigma$. With (ii) we already know that $F\sigma$ is productive and other clauses cannot produce rules whose left side is E-equal to $t\sigma$ and which are not contained in $irred\_rules(t\sigma \approx s\sigma, R_{F\sigma})$ (this is the set of rules produced by $F\sigma$). All rules produced by $F\sigma$ have the same right side $(s\sigma)_{min}$. The equation $u\sigma' \approx v\sigma'$ (remember that $u\sigma =_E t\sigma$) is rewritten to $(s\sigma)_{min} \approx v\sigma'$ (note that with restrictions 5.23 we know that $u\sigma'$ is reduced at the root, hence to $(s\sigma)_{min}$). So the equation $(s\sigma)_{min} \approx v\sigma$, hence also $s\sigma \approx v\sigma$ is satisfied. So $s\sigma$ and $v\sigma'$ are reducible to $E_{F\sigma}$-equal terms. Because $t\sigma >_E s\sigma$ we can only use the rules of $R_{F\sigma}$ to reduce $s\sigma$ or $v\sigma'$, so $I_{F\sigma} \models s\sigma \approx v\sigma'$ (and $I_{F\sigma} \models s\sigma'' \approx v\sigma''$). We have $s\sigma \geq_E v\sigma$ (maximality of the equation) and by part (ii) (see proof above) even $s\sigma >_E v\sigma$. We consider the *equality factoring* inference

$$\pi = \frac{\Gamma\sigma \rightarrow \Delta\sigma, t\sigma \approx s\sigma}{\Gamma\sigma, s\sigma \approx v\sigma \rightarrow \Delta\sigma},$$

which can be lifted (and then instantiated) to an E-equal inference with premise $F\sigma'' = \Gamma\sigma'' \rightarrow \Delta\sigma'', t\sigma'' \approx s\sigma''$. Let us denote the conclusion $\Gamma\sigma'', s\sigma'' \approx v\sigma'' \rightarrow \Delta\sigma''$ of the lifted and then instantiated inference by $F_1\sigma''$. We have $I_{F\sigma''} = I_{F\sigma} \models s\sigma'' \approx v\sigma''$, $\Gamma\sigma'' \subseteq I_{F\sigma''}$ and $I_{F\sigma''} \cap \Delta\sigma'' = \emptyset$. We conclude $I_{F\sigma''} \not\models F_1\sigma''$. This contradicts the saturation (the conclusion $F_1\sigma''$ of the inference has to be satisfied by the interpretation of its maximal premise $F\sigma''$), so there cannot be an equation in $\Delta\sigma'$ which is satisfied in I_C. □

Lemma 5.30 (Interpretations are E-Models)
Let C be a ground clause in NG and N be $\mathcal{M}_E$-saturated on NG_C. Then $E_C^{all} \subseteq I_C$.

Proof: We use induction on the size of clauses. For all clauses CC which are smaller than C we use the induction hypothesis that $E_{CC}^{all} \subseteq I_{CC}$. Let C be a ground clause with maximal term t_{max}. The proof consists of two main cases:

Case A:
All instances $e_1\sigma \approx e_2\sigma$ of E-equations with $t_{max} >_E e_1\sigma$ are contained in I_C:
With the clause $TT := e_1\sigma \approx e_1\sigma \rightarrow e_1\sigma \approx e_1\sigma$ (note $C >_c TT$, $C \neq_E TT$) we conclude

$e_1\sigma \approx e_2\sigma \in E^{all}_{TT} \subseteq I_{TT} \subseteq I_C$ (definition of E^{all}_{TT}; above induction hypothesis for TT; monotonicity of interpretations, lemma 5.15).

Case B:
All instances $e_1\sigma \approx e_2\sigma$ of E-equations with $t_{max} =_E e_1\sigma$ are contained in I_C, if C is greater than this instance of an E-equation (i.e. if $e_1\sigma \approx e_2\sigma$ is contained in E^{all}_C):

Case B1: $e_1\sigma$ and $e_2\sigma$ are both irreducible (w.r.t. $\Longrightarrow_{R_C}$):
We have $e_1\sigma \approx e_2\sigma \in E_C \subseteq I_C$.

Case B2: $e_1\sigma$ or $e_2\sigma$ is reducible:
W.o.l.o.g. we assume $e_1\sigma$ to be reducible.

Case B2.1: for a variable x in e_1 the instance $x\sigma$ is reducible:
By lemma 5.24 we conclude $e_1\sigma \approx e_2\sigma \in I_C$ (otherwise the lemma provides a smaller instance of an E-equation that is false in I_C, this contradicts case A).

Case B2.2: no reducible variable in $e_1\sigma$:
There exists a rule $(\ell\sigma)' \Rightarrow (r\sigma)_{min} \in R_C$ produced by an instance $D\sigma := \Gamma\sigma \rightarrow \Delta\sigma, \ell\sigma \approx r\sigma$ of a clause D with $C >_c D\sigma$, and the rule reduces $e_1\sigma$ (we here use the same substitution for both instances, such a substitution can always be constructed by proper renaming).

Let EE denote the above instance of the E-equation, i.e. $EE := e_1\sigma[(\ell\sigma)'] \approx e_2\sigma$, and the subterm $(\ell\sigma)'$ is not at or below a variable of $e_1 \approx e_2 \in E$.

Case B2.2.1: *Reductions at the root:*
If $e_1\sigma$ is reduced at the root, exchange the roles of $e_1\sigma$ and $e_2\sigma$. If $e_1\sigma$ is reducible at the root, then by construction of our interpretation also all terms E-equal to $e_1\sigma$ are reducible, in particular $e_2\sigma$ is reducible. If both sides are reducible at the root, then the rules used in both reductions have simultaneously been produced and have identical right sides. This means that $e_1\sigma$ and $e_2\sigma$ are reducible to the same term and therefore $e_1\sigma \approx e_2\sigma \in I_C$.

Case B2.2.2: *Reductions below the root:*
Let one of the terms, $e_1\sigma$ say, be reducible below the head. Consider an E-closure inference

$$\frac{\Gamma\sigma \rightarrow \Delta\sigma, \ell\sigma \approx r\sigma \quad \rightarrow e_1\sigma[(\ell\sigma)'] \approx e_2\sigma}{\Gamma\sigma \rightarrow \Delta\sigma, e_2\sigma \approx e_1\sigma[r\sigma]}$$

and let DD denote its conclusion. Condition c) for the application of E-closure inferences holds for this inference, because $>_E$ is total on ground E-congruence classes, therefore a proper subterm of $e_1\sigma$ cannot be greater than $e_1\sigma$ and with restriction 5.23 we exclude $e_1\sigma =_E \ell\sigma$. This inference can be lifted by lemma 4.12 to an inference between clauses in $N \cup E$, which can be instantiated to an inference

$$\frac{D\sigma' \quad EE'}{DD'}$$

with the conclusion $DD' = \Gamma\sigma' \rightarrow \Delta\sigma', e_2\sigma' \approx (e_1[r])\sigma'$, which is E-equal to DD and the E-steps apply equalities with terms e, where $e_2\sigma >_E e$ (see lemma 4.12, note that no side of an E-equation is simply a variable, see restriction 5.23). The same bound on the size of E-equalities applies to $EE =_E EE' = (e_1 \approx e_2)\sigma'$.

We have $EE' >_c DD'$, $EE' >_c DD$, $EE' >_c D$, $C >_c EE$, $C >_c EE'$ and $C \neq_E EE$.

NG is saturated on NG_C, so the (instance of the lifted) inference is redundant. Productive clauses are non-redundant (lemma 5.27), so $D\sigma$ is non-redundant and by lemma 5.29

$D\sigma'$ is also non-redundant (note that $\ell\sigma'$ is not reducible by $\Longrightarrow_{R_{D\sigma}}$, if $D\sigma$ is productive, and $I_{D\sigma} = I_{D\sigma'}$). If $EE' = \rightarrow (e_1 \approx e_2)\sigma'$ is redundant, then we have $I_C \supseteq I_{EE'} \models EE'$ and are done because also $I_C \models EE$ (for the size of E-equalities needed to prove $EE =_E EE'$ see above; the needed instances of E-equations are satisfied by case A).

Therefore, the inference is redundant because of $I_{EE'} \models DD'$.

D is productive, so $I_D \models \Gamma\sigma$ and $I_D \models \Gamma\sigma'$. Because $I_D \subseteq I_{EE'}$ we have $I_{EE'} \models \Gamma\sigma'$. Again, D is productive, so $I_D \cap \Delta\sigma' = \emptyset$, and with lemma 5.29 we conclude $I_{EE'} \cap \Delta\sigma' = \emptyset$. So there is only one equation in the succedent of DD' which can be satisfied, namely

$$I_{EE'} \models e_2\sigma' \approx (e_1[r])\sigma' .$$

Also, the corresponding and E-equal equation in DD is satisfied (for the size of the E-equations used for $DD =_E DD'$, i.e. for $\sigma =_E \sigma'$, see lemma 4.12):

$$I_{EE'} \models e_2\sigma \approx (e_1[r])\sigma$$

$I_{EE'}$ contains also the rule $(\ell\sigma)' \Rightarrow (r\sigma)_{min}$ and satisfies $(r\sigma)_{min} \approx r\sigma$, so $I_{EE'} \models e_1\sigma[r\sigma] \approx e_1\sigma[(\ell\sigma)']$ and by transitivity $I_{EE'} \models e_2\sigma \approx e_1\sigma[(\ell\sigma)']$. This was the equation we were looking for and with $I_{EE'} \subseteq I_C$ we finish case B.

As there are no equations $e_1\sigma \approx e_2\sigma$ in E_C^{all} with $e_1\sigma >_E t_{max}$ (definition of E_C^{all}), and not even equations with $e_1\sigma =_E t_{max}$, if C is smaller or E-equal to an instance of an E-equation with terms E-equal to t_{max}, cases A and B imply $E_C^{all} \subseteq I_C$. □

Lemma 5.31 Let N be set of clauses and C be a ground clause in NG. If N is $\mathcal{M}_E$-saturated on NG_C and does not contain the empty clause, then for all clauses $D\sigma \in NG$ which are smaller than C there exists a substitution σ' such that $I_C \models D\sigma'$ and $\sigma =_E \sigma'$.

Proof: Let C be a ground clause with maximal term t_{max}. In the following proof we consider an instance $H\sigma$ of a clause H in N such that $C >_c H\sigma$ (hence $C \neq_E H\sigma$). Due to the definition of our ordering (which implies E-compatibility, cf. lemma 5.2) we also have $C >_c H\sigma'$ for all σ' with $\sigma =_E \sigma'$. As N is saturated on NG_C, all inferences with such premises $H\sigma'$ have to be redundant.

We derive a contradiction from the fact that there exists a minimal (w.r.t. $>_c$) non-empty ground clause $D_{false} := D\sigma$, $D \in N$, $C >_c D_{false}$ (hence $C \neq_E D\sigma$) and for all $\sigma' =_E \sigma$ the clause $D'_{false} := D\sigma'$ is not satisfied by I_C. By lemma 5.10 we have $I_{D_{false}} = I_{D'_{false}}$.

If D'_{false} is false in I_C, it is also false in $I_{D_{false}} = I_{D'_{false}}$. In fact, as $C >_c D'_{false}$ (from $C >_c D_{false}$ and $C \neq_E D_{false}$), we obtain $I_{D'_{false}} \subseteq I_C$ by appling lemma 5.19. So D_{false} and every D'_{false} is non-redundant.

If D_{false} (or any D'_{false}) is productive, then due to restrictions 5.23 it is satisfied by I_C (cf. lemma 5.25). So we may assume that D_{false} is non-redundant and not productive.

If for a variable x of D the term $x\sigma'$ is reducible by R_C we can construct a smaller false clause using lemma 5.24. This contradicts the minimality of D_{false}. So we also assume D_{false} and all D'_{false} to be reduced ground instances.

It follows a case analysis depending on the maximal equation in D_{false}.

a) If there is a trivial (maximal) equation $t \approx t'$ in the succedent, D_{false} is satisfied because the needed E-equality is satisfied by I_C (see definition 5.4 and lemma 5.30).

b) Let $D_{false} = \Gamma\sigma \rightarrow \Delta\sigma, t_1\sigma \approx t_2\sigma$ be an instance of $D = \Gamma \rightarrow \Delta, t_1 \approx t_2 \in N$ with maximal equation $t_1\sigma \approx t_2\sigma$, $t_1\sigma >_E t_2\sigma$.

b1) The equation $t_1\sigma \approx t_2\sigma$ is strictly maximal in D_{false}:
Since $I_C \not\models D_{false}$, the clause D_{false} has not produced the rule $t_1\sigma \Rightarrow (t_2\sigma)_{min}$. This can only be the case if $t_1\sigma'$ is reducible by $R_{D_{false}}$ (see lemma 5.29; note that D_{false} and any D'_{false} is non-redundant and not productive, the only requirement D_{false} does not fulfill is the irreducibility). Let $t_1\sigma'$ be reducible by $R_{D_{false}}$ at position u_1, u_1 a non-variable position (we consider reduced ground instances). By construction of our interpretations, also $(t_1/u_1)\sigma =_E (t_1/u_1)\sigma'$ is reducible, hence reducible at a non-variable position u_2 of t_1/u_1, so also $t_1\sigma$ is reducible at a non-variable position u of t_1. We reduce with a rule $(s_1\sigma_1)' \Rightarrow (s_2\sigma_1)_{min}$ produced by a clause $C_1\sigma_1 \in NG_C$ smaller than D_{false}, $C_1 = \Gamma_1 \rightarrow \Delta_1, s_1 \approx s_2 \in N$ and $t_1\sigma/u =_E s_1\sigma_1$. If the maximal equation of $C_1\sigma_1$ is E-equal to the maximal equation of D_{false}, then $t_1\sigma$ is reduced to $(t_2\sigma)'$ and I_C satisfies $t_2\sigma \approx (t_2\sigma)'$, so would satisfy D_{false}. So the clauses $C_1\sigma_1$ and D_{false} are not E-equal, the maximal equation of $C_1\sigma_1$ is smaller than the maximal equation of D_{false} and we can consider the *strict superposition right* inference between D_{false} and $C_1\sigma_1$:

$$\pi = \frac{C_1\sigma_1 \quad D_{false}}{\Gamma\sigma, \Gamma_1\sigma_1 \rightarrow \Delta\sigma, \Delta_1\sigma_1, t_1\sigma[u \leftarrow s_2\sigma_1] \approx t_2\sigma}$$

We denote the conclusion by D_π. The inference π can be lifted (cf. lemma 4.12), and we can instantiate the lifted inference to an inference π' E-equal to π. We show that there is exactly one equation in the succedent of $D_{\pi'}$ of π' which can be satisfied by $I_{D'_{false}}$ (we here denote the second premise of π' by $D'_{false} := D\sigma'$):

- $\Gamma\sigma'$ is satisfied:
  $\Gamma\sigma'$ is satisfied by $I_{D_{false}} = I_{D'_{false}}$ (otherwise D'_{false} is true);
- $\Gamma_1\sigma'_1$ is satisfied:
  see lemma 5.29 (note that productive clauses are non-redundant; note also $C_1\sigma_1 \neq_E D_{false}$, so $D'_{false} >_c C_1\sigma_1$).
- no equation in $\Delta\sigma'$ is satisfied:
  $I_{D'_{false}} \cap \Delta\sigma' = \emptyset$ because D'_{false} is false in I_C and $I_{D'_{false}} \subseteq I_C$;
- $\Delta_1\sigma'_1$ is not satisfied because of lemma 5.29 for the productive (hence non-redundant) clause $C_1\sigma_1$

So the only equation which can possibly be satisfied by $I_{D'_{false}}$ is the reduced equation $t_1\sigma'[u \leftarrow s_2\sigma_1] \approx t_2\sigma'$ in the succedent of the conclusion. As $I_{D'_{false}} = I_{D_{false}}$ satisfies $(s_1\sigma_1)' \approx s_2\sigma_1$, it would also satisfy the unreduced equation $t_1\sigma'[u \leftarrow (s_1\sigma_1)'] \approx t_2\sigma'$ of the succedent of D'_{false}, which is impossible therefore.
$C_1\sigma_1$ is productive (hence non-redundant) and $C_1\sigma'_1$ is non-redundant ($I_{C_1\sigma_1} = I_{C_1\sigma'_1}$, $s_1\sigma'_1$ is irreducible, lemma 5.29). D'_{false} is non-redundant by our assumptions. As π' is redundant (because of saturation), $I_{D'_{false}}$ satisfies the conclusion $D_{\pi'}$. By the above considerations, this is a contradiction to the statement that D'_{false} is false in I_C. So the clause D'_{false} cannot be false in I_C and we have found an instance $D\sigma'$ which is satisfied by I_C.

b2) There exists an equation $t_3\sigma \approx t_4\sigma$ in $\Delta\sigma$ which is E-equal to $t_1\sigma \approx t_2\sigma$:
We consider the equality factoring inference

$$\pi = \frac{\Gamma\sigma \rightarrow \Delta\sigma, t_1\sigma \approx t_2\sigma}{\Gamma\sigma, t_2\sigma \approx t_4\sigma \rightarrow \Delta\sigma}.$$

We lift π and instantiate it to an inference π' with premise D'_{false}. Because $I_{D'_{false}} \models t_2\sigma \approx t_4\sigma$ (lemma 5.30; note $t_2\sigma =_E t_4\sigma$), the inference π' is only redundant if D'_{false} is redundant. Using lemma 5.19 this contradicts $I_C \not\models D'_{false}$.

c) Let now D_{false} be a clause $\Gamma\sigma, t_1\sigma \approx (t_1\sigma)' \rightarrow \Delta\sigma$ with maximal equation $t_1\sigma \approx (t_1\sigma)'$. In this case the lemma follows in a similar way using the *equality resolution* inference

$$\frac{\Gamma\sigma, t_1\sigma \approx (t_1\sigma)' \rightarrow \Delta\sigma}{\Gamma\sigma \rightarrow \Delta\sigma}.$$

The lifted (and then instantiated) inference (with premise D'_{false}) is redundant (because of saturation). As the premise D'_{false} is non-redundant, the conclusion of the above inference is true in $I_{D'_{false}} = I_{D_{false}} \subseteq I_C$. So either the antecedent is not satisfied or an equation in the succedent is satisfied, both contradicts the fact that D'_{false} is false, so the instance $D'_{false} = D\sigma'$ is true in I_C.

d) It remains to consider $D_{false} = \Gamma\sigma, t_1\sigma \approx t_2\sigma \rightarrow \Delta\sigma$ with maximal equation $t_1\sigma \approx t_2\sigma$, $t_1\sigma >_E t_2\sigma$. In this case $I_{D_{false}} \models t_1\sigma \approx t_2\sigma$ (because D_{false} is not satisfied) and $t_1\sigma$ is reducible by $\Longrightarrow_{R_{D_{false}}}$ with a rule $(s_1\sigma_1)' \Rightarrow (s_2\sigma_1)_{min}$ produced by a clause $C_1\sigma_1$, $C_1 = \Gamma_1 \rightarrow \Delta_1, s_1 \approx s_2 \in N$ and $t_1\sigma/u = (s_1\sigma_1)'$. D_{false} is a reduced ground instance, so u is a non-variable occurrence of t_1. We consider the *strict superposition left* inference

$$\pi = \frac{C_1\sigma_1 \quad D_{false}}{\Gamma\sigma, \Gamma_1\sigma_1, t_1\sigma[u \leftarrow s_2\sigma_1] \approx t_2\sigma \rightarrow \Delta\sigma, \Delta_1\sigma_1}.$$

Similar to case b1) we get a contradiction: the premises of the inference are non-redundant, so the conclusion has to be satisfied in $I_{D'_{false}}$, but then D'_{false} cannot be false in I_C.

□

Theorem 5.32 (Redundancy-Completeness of $\mathcal{M}_E$)
The inference system $\mathcal{M}_E$ is redundancy-complete.

Proof: Let N be an $\mathcal{M}_E$-saturated set of clauses. Assume $N \cup E$ is inconsistent and N does not contain the empty clause. There are no inferences with premise TOP, so N is $\mathcal{M}_E$-saturated on NG_{TOP}. We show that I_{TOP} is a model for $N \cup E$:

By lemma 5.30 we know that EG is satisfied by I_{TOP}. Assume there exists a clause F in N and a ground substitution σ such that $I_{TOP} \not\models F\sigma$. By $EG \subseteq I_{TOP}$ we conclude $I_{TOP} \not\models F\sigma'$ for all σ' with $\sigma' =_E \sigma$. But this contradicts lemma 5.31. Hence $F\sigma$ is satisfied and I_{TOP} is indeed a model for $N \cup E$.

This is a contradiction to the inconsistency. So $N \cup E$ cannot be inconsistent or N contains the empty clause. □

5.3 System $\mathcal{M}_E$ for $E = AC$

The main difference to inference systems for $E = \emptyset$ is the additional E-closure inference (in addition to being restricted to E-compatible orderings and E-unification).

Now consider the case $E = AC$. As there are no non-variable subterms for any side of a commutativity axiom, the only AC-closure inferences occur with associativity axioms. Consider AC-closure inferences with a clause

$$\rightarrow a + b \approx c$$

and the associativity axiom $\rightarrow (x + y) + z \approx x + (y + z)$ (assume $a+b >_{AC} c$). We list the conclusions of AC-closure inferences for which the left premise is either the above clause or a clause produced from an AC-closure:

$$\begin{aligned}
&\rightarrow a + (b + x) \approx c + x \\
&\rightarrow (z + a) + b \approx z + c \\
&\rightarrow (z + a) + (b + x) \approx z + (c + x) \\
&\rightarrow (z + a) + (b + x) \approx (z + c) + x \\
&\rightarrow a + ((b + x_1) + x_2) \approx (c + x_1) + x_2 \\
&\rightarrow (z_2 + (z_1 + a)) + b \approx z_2 + (z_1 + c) \\
&\ldots
\end{aligned}$$

We observe that the 4th, 5th, 6th clause and every further clause follows from an instance of one of the first three clauses and some AC-equalities applied at terms smaller than the left side (of the equation in the succedent) of the considered clause. As an example we take the 5th clause: substitute x by $x_1 + x_2$ in the first clause:

$$\rightarrow a + (b + (x_1 + x_2)) \approx c + (x_1 + x_2)$$

Now we use the AC-equalities $b+(x_1+x_2) =_{AC} (b+x_1)+x_2$ and $c+(x_1+x_2) =_{AC} (c+x_1)+x_2$ to get the 5th clause (note: $(a + b + x_1 + x_2)\sigma >_{AC} (b + x_1 + x_2)\sigma$ because the ordering is total and hence a simplification ordering; $(a + b + x_1 + x_2)\sigma >_{AC} (c + x_1 + x_2)\sigma$ because $a + b >_{AC} c$ and the ordering is stable under contexts).

Let us assume we use an ordering on clauses with $C\tau = D$ implies $C\tau\sigma >_c D\sigma$ (i.e. the proper subsumption ordering is included). Then above we have shown the compositeness (cf. section 7; roughly speaking: compositeness implies redundancy) of the 5th clause. Similarly every clause above (except the first three clauses) is composite (hence redundant in an $\mathcal{M}_E$-saturated set). The above clause in merely an example: *for every clause* C (and every equation $\ell \approx r$ in its succedent, with $(\ell \approx r)\sigma$ strictly maximal in $C\sigma$, $\ell\sigma >_{AC} r\sigma$ (for an appropriate ground substitution σ) and the root of $\ell\sigma$ is marked with an AC-operator) *we need at most three additional clauses* to make all AC-closure inferences redundant.

If we can guarantee that the first of the above clauses is contained in N, we need only this clause (and not three clauses). We do this in the next section for the system $\mathcal{M}_{Ext}$. We conjecture that often also for $\mathcal{M}_E$ we need at most one additional clause. But we were not able to show this for arbitrary specifications and arbitrary methods of eliminating clauses.

6 The System $\mathcal{M}_{Ext}$

Excluding the E-closure inference rule from $\mathcal{M}_E$ we get the calculus $\mathcal{M}_{Ext}$ (see definition 4.6).

Example 6.1 (Incompleteness of $\mathcal{M}_{Ext}$)
Let $E := \{a+(b+c) \approx (a+b)+c\}$ and $N := \{a+(b+c) \approx d+c \rightarrow\ ,\ \rightarrow a+b \approx d\}$. The set N is $\mathcal{M}_{Ext}$-saturated, but $N \cup E$ is inconsistent. Note that N is not $\mathcal{M}_E$-saturated.

Example 6.2 (Incompleteness even for Superposition Modulo E)
Assume we extend the superposition definition in the following way:

> A term s can be superposed *modulo* E on a term t, if there exists a term t', a position $p \in O(t')$ and a substitution σ such that $t' =_E t$ and $\sigma \in \mu CSU_E(t'/p, s)$ (i.e. $(t'/p)\sigma =_E s\sigma$; note that this is different from $(t\sigma)'/p = s\sigma$: for the former we apply E-equalities before substituting, in the latter we first substitute and then apply E-equalities).

Even with this notion of superposition applied in superposition left and superposition right inferences, the system $\mathcal{M}_{Ext}$ remains incomplete. We present an example: Let $+$ be an AC-operator. We consider a set N of the following clauses:

$$c+x+x \approx d+b+e+e \rightarrow$$

$$\rightarrow c+a+a+b \approx d$$

We assume $c+a+a+b >_{AC} d$. Writing $c+x+x$ instead of $(c+x)+x$ or $c+(x+x)$ we will indicate that it has no influence on our example, which of these concrete AC-representatives of $c+x+x$ we use. This applies similarly to the other terms. N is $\mathcal{M}_{Ext}$-saturated, particularly, because there is no non-variable subterm of the first clause which is AC-unifiable with $c+a+a+b$. But the set is inconsistent: with the substitution $x \leftarrow a+b+e$ the antecedent of the first clause, which is a goal clause, is satisfied. But we cannot find this substitution by AC-unification, we have to guess it.

In the following section we will add a clause

$$\rightarrow c+a+a+b+y \approx d+y.$$

There is a minimal AC-unifier for $c+x+x$ and $c+a+a+b+y$, i.e. $\{x \leftarrow a+b+z,\ y \leftarrow b+z+z\}$. So we get a superposition left inference (in the sense of 4.5, we do not need superposition modulo AC as defined in this example) of the new clause and the above goal clause. This inference has the conclusion

$$d+b+z+z \approx d+b+e+e \rightarrow .$$

With an equality resolution inference of the clause we obtain the empty clause and prove the set of clauses to be inconsistent. Note that to saturate the set of clauses, we only need substitutions which are computed for unification (no substitution is guessed).

We consider the above example again, but under a different light: We explain the situation in terms of usual rewriting with unconditional equations (which is included in our inference system as a special case). Narrowing is the *goal solving variant* of rewriting, the

goal verifying operation. In the case where E is not empty, we know that there are more than one rewrite relations. The well known are *rewriting modulo* E ($\Longrightarrow_{R/E}$) and *rewriting with* E*-matching* ($\Longrightarrow_{R\cdot E}$). We rewrite a term t using rewriting modulo E, if we can match a subterm of an E-variant of t with the left side of a rewrite rule. What is the goal solving operation for that? We have the possibility to substitute variables of t. In the case of narrowing we find this substitution goal directed using unification. But here we have to substitute first, then looking for an E-variant of $t\sigma$ and for a rule to apply. This means we have to guess a substitution! Now the replay of the above example:

Let $+$ be an AC-operator. We search a solution for

$$c + x + x \approx d + b + e + e$$

using the rewrite system

$$R = \{c + a + a + b \Rightarrow d\}$$

The relation $\Longrightarrow_{R/AC}$ is Church-Rosser modulo AC for $\equiv_{R \cup AC}$. There is no subterm (including the term itself) of the left side $c + x + x$ of the goal which is AC-unifiable with the left side of our (ground) rule. This even holds for all terms AC-equal to $c + x + x$. We can apply it, if we substitute x by $a + b$ and then take the E-variant $(c + a + a + b) + b$. This leads to a goal $d + b = d + b + e + e$, which is a dead end street for the proof. We have to use a substitution $x \leftarrow a + b + e$ to prove our goal. But it cannot be constructed by unification.

We can construct a proof of the above goal using an additional rule $c+a+a+b+y \Rightarrow d+y$, which is known for rewriting as the AC-extended rule of $c + a + a + b \Rightarrow d$ and is only used in connection with rewriting using E-matching (here AC-matching). We find the solution by AC-unification (see above). Therefore usually the narrowing modulo E (which uses such rules) is essentially a narrowing based on a rewrite relation with E-matching ([Bockmayr 90], chapter "Conditional Rewriting and Narrowing Modulo an Equational Theory"), not using rewriting modulo E.

Such simple examples show that we have to require additional properties for N (beside of $\mathcal{M}_{Ext}$-saturation) to get redundancy-completeness. We will formulate this requirements with the help of *extended clauses* (see next section).

Let us remark here that for $E = AC$ it is possible to extend the notion of superposition (often also called paramodulation) to get a complete inference system without the use of AC-extended clauses (which are very similar to extended rules, cf. next section). In [Rusinowitch/Vigneron 91] a calculus is presented introducing the notion of ordered AC-paramodulation and ordered extended AC-paramodulation. This paper is discussed in section 10.1.

6.1 Extended Clauses

In the rewriting area *extended rules* are well known ([Peterson/Stickel 81], [Jouannaud/ Kirchner 86], [Bachmair 87], [Bachmair 88]). They are used as an alternative to clauses introduced by E-closure inferences. We generalize the notion of extension from rules to clauses.

Definition 6.3 (E-Extended Clause)
Let C be a clause $\Gamma \rightarrow \Delta, \ell \approx r$ and $e_1 \approx e_2$ be an E-equation. The clause

$$C_E = \Gamma \rightarrow \Delta, e_1[p \leftarrow \ell] \approx e_1[p \leftarrow r]$$

is an *E-extended clause* of C (for the equation $\ell \approx r$) if

- $p \in O(e_1)$ is a non-variable position,
- $p \neq \varepsilon$,
- e_1/p and ℓ are E-unifiable.

In that case we call the equation $e_1[p \leftarrow \ell] \approx e_1[p \leftarrow r]$ *the extended equation* of C_E for $\ell \approx r$ in C and $\ell \approx r$ an equation of C which has to be extended.

Note that in the above definition we do not need to compute the E-unifier of e_1/p and ℓ. That is one of the advantages of extended clauses (compared with the construction of E-closure inferences, where we compute the unifier). In the AC-case the unifier is trivial and if ℓ is not simply a variable, there can only be an AC-extended clause, if the root symbol of ℓ is an AC-operator.

Note that the relation "*is an E-extended clause of*" is a partial and terminating relation in a set of clauses. Considering a pair (C_E, C) of clauses such that C_E is an extended clause of C, then there is a unique equation in C_E which we can call *the* extended equation of C_E for C.

Definition 6.4 (Unextended Clause in N)
Let N be a set of clauses and C be a clause in N. If there is no other clause D in N such that C is an E-extended clause of D, then C is called an *unextended clause in N*.

Definition 6.5 (Extended Clause in N)
Let N be a set of clauses. Let $C_E := \Gamma \rightarrow \Delta, t \approx s$ be a clause in N. If there exists another clause $C := \Gamma \rightarrow \Delta, \ell \approx r$ in N such that C_E is an E-extended clause of C for $\ell \approx r$, $t \approx s$ is the extended equation of C_E for $\ell \approx r$ in C and

- either C is an unextended clause in N
- or else there exists a third clause D in N such that C is an E-extended clause *in N* (this is the recursion in this definition) of D and $\ell \approx r$ is the extended equation of C for D,

then C_E is called an E-extended clause *in N* of C.

Example 6.6 There may exist a (non-Horn) clause C_E which is an E-extended clause for two different clauses C_1 and C_2. If we write an equation as eq_i and its extended equation as $c[eq_i]$ for a context c, we may use the following clauses as an example for the previous statement:

$$C_1 = \rightarrow u[eq_1], eq_2$$
$$C_2 = \rightarrow eq_1, v[eq_2]$$
$$C_E = \rightarrow u[eq_1], v[eq_2]$$

Example 6.7 There exists a set N of clauses such that there exists a clause $C_E \in N$ which is an E-extended clause for $C_1 \in N$, but C_E is not an extended clause *in N* (because C_1 is already an extended clause, but C_E does not extend an extended equation of C_1):

$$C = \rightarrow eq_1, eq_2$$

$$C_1 = \rightarrow u[eq_1], eq_2$$

$$C_E = \rightarrow u[eq_1], v[eq_2]$$

There may be other E-extended clauses in N of C (see C_2 in the previous example 6.6), and further extended clauses, e.g.

$$C_{1E} = \rightarrow c[u[eq_1]], eq_2,$$

but for each equation in the succedent of C, there is at most one line of extended clauses (we have to extend only extended equations of extended clauses).

Definition 6.8 (Sets Closed under E-Extension)
A set N of clauses is *closed under E-extension* if for every clause $C \in N$

- which contains (an occurrence of) an equation $\ell \approx r$ in the succedent,
- has a ground instance $C\sigma$ such that the σ-instance of the above (occurrence of the) equation $\ell \approx r$ is strictly maximal in $C\sigma$,
- $\ell\sigma >_E r\sigma$, and
- either C is an unextended clause in N, or else
- C is an extended clause *in* N of a clause $D \in N$ and $\ell \approx r$ is the extended equation of C for D,

each extended clause C_E of C (if any) for the equation $\ell \approx r$ is also contained in N.

Definition 6.9 (Partitions of N)
Let N be a set of clauses. We define subsets Ext, $SExt$ and NE of N. A clause $C \in N$ belongs to NE, if C is an unextended clause in N. A clause $C_E \in N$ belongs to Ext if there is a clause $C \in N$ such that C_E is an extended clause in N for C. All other (extended) clauses belong to the set $SExt$ (superfluous extended clauses). We can write N as the disjoint union of NE, $SExt$ and Ext.

There may be useless extended clauses in N (cf. clause C_E of example 6.7 in a set N which does not contain the clause C_2 of example 6.6). This kind of useless clauses is excluded from Ext by the previous definition (they are contained in $SExt$). There are other useless extended clauses, i.e. clauses, where there is no substitution σ making the equation $\ell \approx r$ maximal (after instantiating the equation with a unifier of ℓ and a subterm of an E-equation). But to exclude them, we need to compute the unifier. To avoid some of them we may restrict the introduction of extended clauses to equations $\ell \approx r$, where there is at least one ground substitution σ with $(\ell \approx r)\sigma$ is strictly maximal in $C\sigma$ and $\ell\sigma >_E r\sigma$ (cf. definition of closed sets, 6.8), but the substituted ℓ may not be E-unifiable with a position in an E-equation, so again the extended clause may be useless. In the AC-case the unifier is trivial and we often have no problems to exclude such useless AC-extended clauses.

Definition 6.10 (Redundancy of E-Extended Clauses)
Let $C_E \in Ext$ be an E-extended clause in a set $N = NE \cup Ext \cup SExt$ of clauses. By definition C_E and every ground instance of it is *non-redundant.*

E-extended clauses are (nearly) always redundant by the former definition of redundancy (definition 5.20). They are nevertheless needed. To incorporate the new redundancy definition into theorem provers working with definition 5.20, we have not simply to add E-extended clauses to our set of clauses, but also to *protect* them from being erased because of redundancy (in the sense of 5.20). When using inference systems which need E-extended clauses, we therefore consider sets N of clauses which are partitioned into the set $NE \cup SExt$ and the set Ext of E-extended clauses in N (see definition 6.9).

Definition 6.11 Let $D := \Gamma \to \Delta, \ell \approx r$ be a clause in N. We define the set $Ext^*(D, \ell \approx r)$ of clauses by

1. $D \in Ext^*(D, \ell \approx r)$
2. $D_E \in Ext^*(D, \ell \approx r)$ if $D_E \in N$, $D_E = \Gamma \to \Delta, t \approx s$, there exists a clause $D_1 = \Gamma \to \Delta, t_1 \approx s_1$ in N such that $D_1 \in Ext^*(D, \ell \approx r)$ and D_E is an extended clause of D_1 for $t_1 \approx s_1$
3. no other clauses are in $Ext^*(D, \ell \approx r)$.

In the above situation we call the equation $t \approx s$ of D_E *the extended equation of* D_E *for* D *(for the equation* $\ell \approx r$*).*

$$Ext^*(D) := Ext^*(D, \ell \approx r) \cup \bigcup_{t \approx s \in \Delta} Ext^*(D, t \approx s)$$

We then define $Ext^+(D)$ by

$$Ext^+(D) := Ext^*(D) \setminus \{D\}$$

and

$$Ext^+(D, \ell \approx r) := Ext^*(D, \ell \approx r) \setminus \{D\}.$$

Lemma 6.12 (Properties of Extended Clauses)
Let $N := NE \cup Ext \cup SExt$ be a set of clauses. Let D_E be a clause of Ext. Then there exists an (unextended) clause D in NE and a position p with

- $D = \Gamma \to \Delta, \ell \approx r$ and $\ell \approx r$ is an equation of D which has to be extended
- $D_E = \Gamma \to \Delta, t \approx s$
- $D_E \in Ext^+(D, \ell \approx r)$
- $p \in O(t)$, $p \in O(s)$, $p \neq \varepsilon$
- $t/p = \ell$, $s/p = r$
- and if σ is a substitution such that $D_E\sigma$ is a ground clause, then also $D\sigma$ is a ground clause.

Proof: We use induction on the height of terms. Assume the lemma holds for all clauses $D_1 \in Ext$ with $heights(D_E) >> heights(D_1)$, where $heights(C)$ is the multiset of the heights of each term occurring in C, e.g. $heights(a(x) \approx b \to f(g(x)) \approx b) = \{1, 0, 2, 0\}$.

D_E is in Ext, so there exists a clause $D_1 \in N$, $D_1 = \Gamma \to \Delta, t_1 \approx s_1$, and $t_1 \approx s_1$ is an equation of D_1 which has to be extended. Moreover there exists a position $p_1 \in O(t)$ such that $p \neq \varepsilon$, $t/p_1 = t_1$ and $s/p_1 = s_1$. D_E is an E-extended clause in N of D_1. If

D_1 is a clause of NE we are done with $p := p_1$ and $D := D_1$. Otherwise we may use our induction hypothesis for D_1 (w.o.l.o.g. we assume D_1 to be in Ext: if all clauses that D_E extends were contained in $SExt$, then D_E itself would not belong to Ext), which gives us the existence of a clause $D \in NE$, $D = \Gamma \to \Delta, \ell \approx r$, the equation $\ell \approx r$ an the equation in D which has to be extended, $t_1/p_2 = \ell$ and $s_1/p_2 = r$ for an appropriate $p_2 \neq \varepsilon$. The lemma holds with this clause D and $p := p_1.p_2$. □

For extended clauses in $SExt$, the previous lemma does not hold. The clauses C_E, C_1 and C of example 6.6 form an example, where $C \in NE$, $C_1 \in Ext$ and for C_E there exists no clause in NE such that C_E is an E-extended clause of it.

Definition 6.13 (Ordering $>_{eq}^{ext}$ over Equations in Clauses)
Let $N := NE \cup Ext \cup SExt$ be a set of clauses. The *E-multiset expression for $\mathcal{M}_{Ext}$* of an (occurrence of an) equation $t_1 \approx t_2$ in a ground clause $C\sigma = \Gamma \to \Delta$ is defined as:

(i)	$\{([t_1]_E, 3), ([t_2]_E, 3)\}$	if $t_1 \approx t_2$ belongs to Γ
(ii)	$\{([t_1]_E, 0), ([t_2]_E, 0)\}$	if $t_1 \approx t_2$ belongs to Δ, $C \in Ext$, $C \in Ext^+(D)$ and $t_1 \approx t_2$ is an instance of the extended equation of C for D (for a clause $D \in N$), $C\sigma \neq_E D\sigma$, $t_1 \neq_E t_2$ and t_1 or t_2 is a strictly maximal (w.r.t. $>_E$) term of $C\sigma$
(iii)	$\{([t_1]_E, 1), ([t_2]_E, 1)\}$	if $t_1 \approx t_2$ belongs to Δ, $\Delta = t_1 \approx t_2$ and $t_1 =_E t_2$
(iv)	$\{([t_1]_E, 2), ([t_2]_E, 2)\}$	if none of the above cases apply

The ordering $>_{eq}^{ext}$ over (occurrences of) ground equations is defined as the multiset extension of the lexicographic combination of $>_E$ and $>$ (the ordering on natural numbers) on their E-multiset expressions for $\mathcal{M}_{Ext}$.

The complexity of an equation in the succedent is "normally" defined by point (iv) above (e.g. for nearly all equations in the succedent of $C\sigma$ if C is not an extended clause). Only the extended equation in the succedent of an extended clause is a little bit smaller, if the extended clause is possibly useful. E-equations get a special complexity: they are greater than (useful) extended equations, but smaller than non-extended equations between terms which are not E-equal.

Working with this ordering has the advantage that sometimes more (instances of) E-equations are available to show a clause or inference to be useless (cf. "composite" in section 7). If C is not an extended clause, then every equation EE applicable at terms in C is contained in E_C^{all}, so EE can be used to show C to be useless (we will prove this later: lemma 7.21). E.g. if we use rewriting with a set of rules R to simplify or eliminate clauses, we may use the relations $\Longrightarrow_{R \cdot E}$ and $\Longrightarrow_{R/E}$. This is an advantage over $\mathcal{M}_E$ (where we in general even cannot use $\Longrightarrow_{R \cdot E}$). But the price we pay is the special treatment of extended clauses (e.g. we cannot eliminate clauses in Ext).

The same way as in 5.1 we define a total ordering over clauses:

Definition 6.14 (Ordering over Clauses)
The ordering $>_c^{ext}$ is the multiset extension of $>_{eq}^{ext}$ comparing the E-multiset expression for $\mathcal{M}_{Ext}$ of the clauses. Here the E-multiset expression for $\mathcal{M}_{Ext}$ of a clause is the multiset of the E-multiset expressions for $\mathcal{M}_{Ext}$ of each occurrence of an equation in that clause.

The ordering $>_c^{ext}$ is similar to $>_c$ (see lemma 5.2), except that it is E-compatible only for clauses of the same partition, i.e. $C\sigma >_c^{ext} D\tau$ implies $C_1\sigma_1 >_c D_1\tau_1$, if $C\sigma =_E C_1\sigma_1$, $D\tau =_E D_1\tau_1$, C and C_1 belongs to the same partition and the same holds for D and D_1.

6.2 Interpretations for $\mathcal{M}_{Ext}$

An *equality Herbrand interpretation* is a congruence on ground terms. We now define such an interpretation for sets NG of ground clauses. We will follow the construction in section 5.1. The main differences are:

- We use the ordering $>_c^{ext}$ instead of $>_c$.
- We separate N into non-extended clauses NE and extended clauses $Ext \cup SExt$: the interpretation of a ground clause will be defined differently, depending on whether it is an instance of an extended clause; certain lemmata will only hold for extended clauses (of Ext), others only for non-extended clauses.

Definition 6.15 Let C be a ground clause with maximal (w.r.t. $>_E$) term t_{max}.

$$E_C^{all} := \{\ e_1\sigma \approx e_2\sigma \quad | \quad \sigma \text{ ground substitution}, e_1 \approx e_2 \in E,\ C >_c^{ext} \rightarrow e_1\sigma \approx e_2\sigma \text{ or } C =_E \rightarrow e_1\sigma \approx e_2\sigma\ \}$$

Definition 6.16 Let $N := NE \cup Ext \cup SExt$ be a set of (not necessarily ground) clauses such that N is partioned into $NE \cup Ext \cup SExt$ as defined by 6.9. We define:

$$NEG := \{C\sigma \mid C \in NE, \sigma \text{ ground substitution}\},$$

$$ExtG := \{C\sigma \mid C \in Ext, \sigma \text{ ground substitution}\},$$

$$SExtG := \{C\sigma \mid C \in SExt, \sigma \text{ ground substitution}\},$$

$$NG := NEG \cup ExtG \cup SExtG \cup TG \cup \{TOP\}, \text{ and}$$

$$NG_C := \{D \mid D \in NG, C >_c^{ext} D\},$$

where

$$TG := \{t \approx t \rightarrow t \approx t \mid t \text{ ground term}\}$$

and

$$TOP := \rightarrow \top \approx \top,$$

with $\top >_E t$ for all ground terms $t \neq \top$.

Note that the clause TOP is the strictly maximal clause of NG and satisfied in any interpretation. The TG-equations will not contribute to the interpretation of NG. Nevertheless we include them here, so we can speak of an interpretation I_{TT}, for $TT \in TG$. Note that a clause $t \approx t \rightarrow t \approx t$ of TG is greater than any clause which contains a term E-equal to t and which can possibly introduce new rules into the interpretation 6.17.

Definition 6.17 (Interpretation)
Let N and NG be sets of clauses as in the previous definition. Let C denote a clause of NG. We assume that $Rules_D$, R_D, E_D and I_D have been defined for all clauses D of NG for which $C >_c^{ext} D$. For a clause C with $C \in NEG$ or $C \in ExtG$ we define

$$R_C = \left(\bigcup_{C >_c^{ext} D} Rules_D \right)$$

$$E_C = \{e_1 \approx e_2 \in E_C^{all} \mid e_1 \text{ and } e_2 \text{ are both irreducible w.r.t. } \Longrightarrow_{R_C}\}$$

$$I_C = closure(R_C, E_C)$$

and

$$Rules_C = irred\_rules(t \approx s, R_C)$$

if all of the following conditions hold, otherwise $Rules_C = \emptyset$.

1. $C = \Gamma \rightarrow \Delta, t \approx s$
2. $I_C \not\models \Gamma \rightarrow \Delta$
3. $t \approx s$ is strictly maximal (w.r.t. $>_{eq}$) in C, $t >_E s$
4. either $C \in NEG$ and t' is not reducible by R_C (for all $t' =_E t$) or
   C is the extension of a productive clause, more precisely:
   C is in $ExtG$, there exists a clause $G\sigma$ in NG, $G\sigma = \Gamma_1 \rightarrow \Delta_1, \ell\sigma \approx r\sigma$ such that C is an instance of an extended clause of G for the equation $\ell \approx r$, $C >_c^{ext} G\sigma$, $C \neq_E G\sigma$, $t \approx s$ is the extended equation in C for $G\sigma$, $\ell\sigma \approx r\sigma$ is the strictly maximal occurrence of an equation in $G\sigma$ and $Rules_{G\sigma} \neq \emptyset$.

For any other clause $C \in NG$ (e.g. $C \in SExtG$) we define the above sets as we have done it for clauses in NEG and $ExtG$, except that C never produces any rule, i.e. $Rules_C = \emptyset$.

Lemma 6.18 (Independence from E-Representatives)
For all ground clauses C and C' with $C =_E C'$ and either both clauses are in NEG or both clauses are in $ExtG$ or both clauses are in $SExtG$, we have $I_C = I_{C'}$.

Proof: The ordering $>_c^{ext}$ has the property that $C >_c^{ext} D$ implies $C' >_c^{ext} D$, if C' belongs to the same partition as C or $C >_c^{ext} D'$, if D' belongs to the same partition as D. The lemma follows from this fact and the definition of interpretations. □

Lemma 6.19 (Productive E-Extended Clauses)
If a clause $C_E\sigma \in ExtG$ with maximal equation $t \approx s$ ($t >_E s$) is productive, then it is an extension of a productive clause G_1. If G_1 produces a rule $\ell_1 \Rightarrow r_1$, then there exists a context u_1 such that $t' =_E u_1[\ell_1]$ and $s' =_E u_1[r_1]$.

There exists also a clause $G \in NE$ (not in Ext or $SExt$) such that $C_E \in Ext^+(G)$, and $G\sigma$ is productive. If $G\sigma$ produces a rule $\ell \Rightarrow r$, then there exists a context u such that $t' =_E u[\ell]$ and $s' =_E u[r]$.

Proof: The first part follows from part 4 in definition 6.17 and the definition of extended clauses (definition 6.3). The second part follows from induction (similar to the induction in lemma 6.12) using the first part. □

Lemma 6.20 (Termination of $\Longrightarrow_{R_C}$)
The rewrite relations $\Longrightarrow_{R_C}$ and $\Longrightarrow_{R_C/E}$ are terminating.

Proof: Every rule produced by a clause of NG is contained in $>_E$. □

Lemma 6.21 Let C be a ground clause in NG with maximal term t. Every rule produced by a clause G greater than C has a left side ℓ with $\ell \geq_E t$. If C is productive or a term E-equal to t occurs in the antecedent and G is greater than C, we even have $\ell >_E t$.

Proof: The left side of a rule produced by a clause D is not smaller than a maximal (w.r.t. $>_E$) term of D. So if D produces a rule with left side ℓ and $t >_E \ell$ we have $C >_c^{ext} D$. Hence greater clauses can only produce rules $\ell \Rightarrow r$ with $\ell \geq_E t$.

If C is productive, then all terms E-equal to t are reducible in any interpretation of a clause greater than C and for every greater clause G with maximal equation $t' \approx s$ we have $irred\_rules(t' \approx s, R_G) = \emptyset$. If a term E-equal to t occurs in the antecedent of C, no clause greater than C can have a maximal equation $\ell \approx r$ in the succedent with $t \geq_E \ell$. So in both cases all rules produced later have left sides greater than t. □

Lemma 6.22 Let C be a ground clause of NG with maximal (w.r.t. $>_E$) term t. Let D be a ground clause in NG greater than C, i.e. $D >_c^{ext} C$.

A term u with $t >_E u$ is reducible w.r.t. $\Longrightarrow_{R_D}$ if and only if u is reducible w.r.t. $\Longrightarrow_{R_C}$.

An equation $e_1 \approx e_2$ with $t >_E e_1$ is contained in E_D if and only if it is already contained in E_C.

Proof: The first part follows from lemma 6.21 and the trivial fact that the left side of the rule which reduces u cannot be greater than u.

If such an equation $e_1 \approx e_2$ is not contained in E_C, e_1 or e_2 is reducible by $\Longrightarrow_{R_C}$ and it will remain reducible in R_D as $R_C \subseteq R_D$, and hence it is not in E_D.

If an equation $e_1 \approx e_2$ is contained in E_C, e_1 and e_2 are irreducible w.r.t. $\Longrightarrow_{R_C}$ and they will remain irreducible in R_D because of the first part of this lemma. □

A lemma similar to 5.15 (monotonicity of interpretations) or 5.19 (increasing chains of interpretations preserve the truth of ground clauses) does not hold. There may be instances of E-equations which disappear in the interpretations of greater clauses. But this problem is solved when considering saturated sets (see lemmata 6.45 and 6.46). We have such kind of monotonicity for the sets of rules:

Lemma 6.23 (Monotonicity of Rewrite Systems)
Let C and D be ground clauses with $C >_c^{ext} D$. Then we have $R_C \supseteq R_D$.

Proof: Follows from definition 5.8. □

Lemma 6.24 (Church-Rosser Property of $\Longrightarrow_{R_C}$)
For every ground clause C we have:

1. A term t is reducible by $\Longrightarrow_{R_C}$ if and only if it is reducible by $\Longrightarrow_{R_C \cdot E}$ (but not necessarily to the same term).
2. There are no overlaps between rules in R_C, so $\Longrightarrow_{R_C}$ is confluent (hence confluent modulo E and modulo E_C).

3. The relation $\Longrightarrow_{R_C}$ is Church-Rosser modulo E_C for the congruence $\equiv_{R_C \cup E_C}$. (But note that we have $E_C \neq E_C^{all}$ in general).

Proof: The first and second property follows from the construction of R_C (in particular from definition 5.5). With the second property and the termination (lemma 6.20) we can apply lemma 3.9 so that it remains to show the local ground coherence modulo E_C to prove the third part.

We can use the same arguments as in the proof of lemma 5.16 considering the same peaks. So we do not repeat the proof here. □

Lemma 6.25 An equation $u \approx v$ is satisfied in an interpretation I_C if and only if the terms u and v are joinable modulo E_C using the rewrite relation $\Longrightarrow_{R_C}$:

$$I_C \models u \approx v \qquad \text{if and only if} \qquad u \Downarrow_{R_C} v$$

Proof: Follows from the Church-Rosser property stated in the previous lemma. □

Lemma 6.26 Let C be a clause of NG with maximal term t and D a ground clause greater than C.

An equation $u \approx v$ with $t >_E u$ and $t >_E v$ is true in I_D if and only if it is true in I_C.

If a term E-equal to t occurs in the antecedent of C, then an equation $u \approx v$ with $t \geq_E u$ and $t \geq_E v$ is true in I_D if and only if it is true in I_C.

Proof: Follows from Church-Rosser property (lemma 6.24) and lemmata 6.21 and 6.22. For the second part we notice that clauses greater than C cannot produce a rule which reduces u or v or an intance of an E-equation applicable at u or v. □

We have $E_C \subseteq E_C^{all}$, in particular $E_{TOP} \subseteq EG$, and the inclusion is (in general) proper. Because we do not have $E_C^{all} \subseteq I_C$, the following is **not true** in general:

$$I_C \models u \approx v \qquad \text{if and only if} \qquad I_C \models u' \approx v'$$

But in the next section (with one more restriction on E, but a weaker restriction than 5.23) we prove that for interpretations of $\mathcal{M}_{Ext}$-saturated sets we have $E_C^{all} \subseteq I_C$, in particular $EG \subseteq I_{TOP}$ and (for $\mathcal{M}_{Ext}$-saturated sets)

$$I_C \models u \approx v \qquad \text{if and only if} \qquad I_C \models u' \approx v', u' =_{E_C^{all}} u, v' =_{E_C^{all}} v$$

$$I_{TOP} \models u \approx v \qquad \text{if and only if} \qquad I_{TOP} \models u' \approx v', u' =_E u, v' =_E v.$$

6.3 Redundancy-Completeness of $\mathcal{M}_{Ext}$

We first define redundancy (of clauses and inferences), saturation (of sets) and completeness (of inference systems) similarly to the corresponding definitions at the beginning of section 5.2.

Definition 6.27 (Redundancy of Clauses)
Let N be a set of clauses. Let D be a clause in $N \setminus Ext$ and let C be a ground instance of D. C is *E-redundant* (or for short *redundant*) in N, if it is satisfied in its partial interpretation, i.e $I_C \models C$.

A non-ground clause C of $N \setminus Ext$ is E-redundant (in a set N) if every ground instance $C\sigma$ of it is E-redundant in N, i.e. $I_{C\sigma} \models C\sigma$ for all ground substitutions σ.

Note that we do not apply this redundancy definition to clauses of Ext (cf. 6.10).

Definition 6.28 (Redundancy of Inferences)
An inference π from ground instances $C_1, \ldots, C_n$ of clauses in $N \cup E$ and conclusion D is *E-redundant* (or for short: *redundant*) in N, if one of its premises is redundant or if the conclusion is satisfied in I_{C_j}, i.e. $I_{C_j} \models D$, where C_j is the maximal premise of the inference π.

A (non-ground) inference π from N is redundant (more precisely E-redundant in N) if all its ground instances $\pi\sigma$ are redundant.

Definition 6.29 (Saturation and Completeness)

1. Let π be an inference of $\mathcal{M}_{Ext}$ with premises $C_1, \ldots, C_n$ and conclusion D. Every inference of $\mathcal{M}_{Ext}$ with premises $C_1\sigma, \ldots, C_n\sigma$ and conclusion $D\sigma$ for a ground substitution σ is called a *ground instance* $\pi\sigma$ of π. Note that there are ground substitutions σ such that there is no inference with premises $C_i\sigma$ and conclusion $D\sigma$.
2. Let MG be a set of ground clauses. The set N of clauses is *$\mathcal{M}_{Ext}$-saturated on MG*, if every ground inference π with premises in MG is E-redundant in N.
3. A set N of clauses is *$\mathcal{M}_{Ext}$-saturated*, if every inference (of $\mathcal{M}_{Ext}$) with premises in $N \cup E$ is E-redundant in N.
4. An inference system $\mathcal{M}_{Ext}$ is *redundancy-complete*, if for every $\mathcal{M}_{Ext}$-saturated set N which is closed under E-extension, N contains the empty clause, if $N \cup E$ is inconsistent.

Now we prove that $\mathcal{M}_{Ext}$ is redundancy complete. We often reduce our consideration to the ground case. With the help of the next lemma we even further restrict our considerations to reduced ground instances of clauses in N.

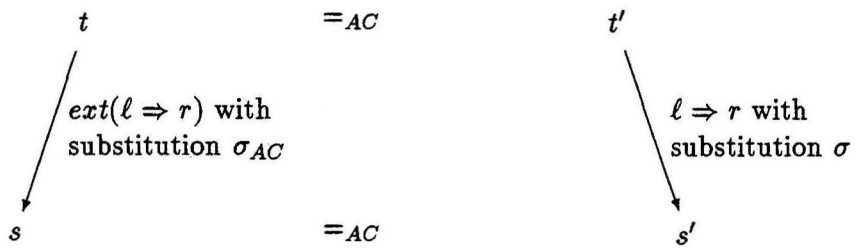
Lemma 6.30 Let N be a set of clauses. Let D be a clause, σ a ground substitution and C a ground clause in NG. If there exists a variable $x \in vars(D)$ such that $x\sigma$ is reducible by R_C, then there exists a ground substitution σ_1 such that $D\sigma >_c^{ext} D\sigma_1$ and $I_C \models D\sigma$ if and only if $I_C \models D\sigma_1$.

Proof: If $x\sigma \Longrightarrow_{R_C} t$, we define σ_1 to be the substitution for which $x\sigma_1 = t$ and $y\sigma_1 = y\sigma$, for all $y \neq x$. We have $I_C \models x\sigma \approx t$ and so $D\sigma_1$ is true in I_C if and only if $D\sigma$ is true in I_C. □

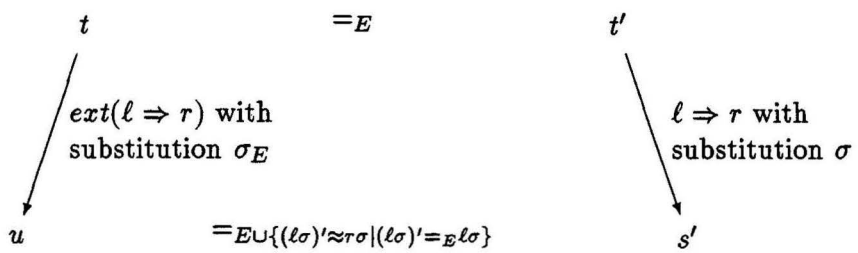
Now we present a technical lemma concerning extended clauses. Assume the clause $D\sigma$ in the following lemma is productive (we will particularly need the lemma in that case). If a term t' (E-equal to t) is reducible by rules produced by $D\sigma$ or $D_E\sigma$ (for an E-extension

$D_E \in Ext^+(D)$), the lemma (part 2) proves the existence of a clause we can superpose on t (we will need this in lemma 6.48). In lemma 6.44 we consider instances $e_1 \approx e_2$ of E-equations which are reducible. So by the lemma we can superpose a clause on the E-equation, say on a subterm of e_1. The following lemma ensures that in some situations (see part 3), we can even superpose on e_1 at the root (and therefore also on e_2; we will need this in lemma 6.44).

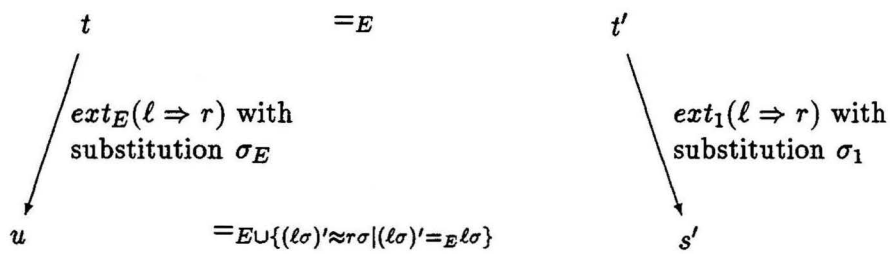
Let us with some pictures motivate the following lemma. First consider unconditional rewriting. For the convergence of coherence pairs we sometimes replace reductions with a rule $\ell \Rightarrow r$ by reduction with a corresponding extended rule $ext(\ell \Rightarrow r)$. For $E = AC$ we get the following diagram:



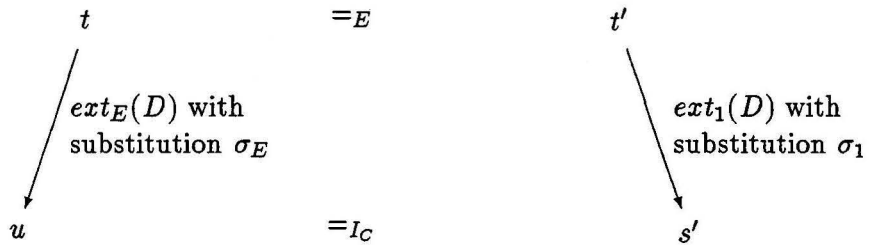
For non-linaer E we sometimes need equations E-equal to the instance $\ell\sigma \approx r\sigma$ to prove the equation at the bottom (see example 6.33):



At the right reduction we might also use an extension ext_1 of $\ell \Rightarrow r$. Then we possibly need another extension ext_E on the left (see example 6.32):



Now assume the rule $\ell\sigma \Rightarrow r\sigma$ corresponds to the maximal equation in the succedent of a clause $D\sigma$ and $D\sigma$ is smaller than a clause C. If I_C contains the equality at the bottom of the previous picture, then we get:



If t is a term contained in a clause F with $C >_c^{ext} F$, and t' is reduced to s' by a rule produced by a clause $ext_1(D)$, then we have a superposition from a clause $ext_E(D)$ into F replacing t by u. Moreover the term u is I_C-equal to s'. We need the lemma below to obtain such an inference at some places in the following completeness proof.

Lemma 6.31 (Use of Extended Clauses)
Let $NE \cup Ext \cup SExt$ be a set closed under E-extension. Let $D = \Gamma \to \Delta, \ell \approx r$ be a clause in $NE \cup Ext$, c a ground context, t a ground term and σ a ground substitution such that $\ell\sigma \approx r\sigma$ is the strictly maximal equation in $D\sigma$, $\ell\sigma >_E r\sigma$ and $t = (c[(\ell\sigma)']')'$, i.e. $t =_E c[(\ell\sigma)']$ and $(\ell\sigma)' =_E \ell\sigma$. Let C be a clause with $C >_c D\sigma$, $C \neq_E D\sigma$ and the maximal term t_{max} of C is not smaller than t, i.e. $t_{max} \geq_E t$. Assume I_C satisfies all ground instances of E-equations between terms smaller than t.

1. There exists a clause $D_E = \Gamma \to \Delta, u_E[\ell] \approx u_E[r]$ in $NE \cup Ext$ (sometimes D itself), a substitution σ_E (sometimes σ itself) with $\Gamma\sigma = \Gamma\sigma_E$, $\Delta\sigma = \Delta\sigma_E$, $(\ell\sigma)' = (\ell\sigma_E)'$, $r\sigma = r\sigma_E$, and a position p_E and a context c_E which is a prefix of t up to p_E (i.e. $c_E = t[p_E \leftarrow []]$) such that $(u_E[\ell])\sigma_E =_E t/p_E$. D_E is obtained by a finite number (sometimes 0) of E-extensions starting with D, i.e. $D_E \in Ext^*(D, \ell \approx r)$. If I_C satisfies $(\ell\sigma)'' \approx r\sigma$ (for all $(\ell\sigma)'' =_E \ell\sigma$), then $I_C \models c_E[u_E[r]\sigma_E] \approx c[r\sigma]$ (note $u_E[r]\sigma_E = u_E\sigma_E[r\sigma]$).
2. If t' is reducible at position p_1 by a rule $(u_1[\ell]\sigma)' \Rightarrow u_1[r]\sigma$ produced by $D_1\sigma$ and $D_1 \in Ext^*(D, \ell \approx r)$, then there exists a clause D_E (as above), a substitution σ_E (as above), a context c_E and a position p_E in t such that $c_E = t[p_E \leftarrow []]$ and $t/p_E =_E u_E[\ell]\sigma_E$. If I_C satisfies $(\ell\sigma)'' \approx r\sigma$ (for all $(\ell\sigma)'' =_E \ell\sigma$), then $I_C \models c_E[u_E[r]\sigma_E] \approx t'[p_1 \leftarrow u_1[r]\sigma]$ (note $c_E[u_E[r]\sigma] = t[p_E \leftarrow u_E[r]\sigma_E]$).
3. If t is a ground instance of a side of an E-equation $e \approx e e \in E$, i.e. $t = e\sigma_e$ and p is a non-variable position of e such that t (resp. $e\sigma_e$) is reducible at position p to s by application of a rule $(\ell\sigma)' \Rightarrow r\sigma$ produced by $D\sigma$, then there exists a clause D_E (as above), a substitution σ_E (as above) such that $(u_E[\ell])\sigma_E =_E t$ (i.e. $u_E[\ell]$ matches t at the root position) and $u_E\sigma_E[r\sigma] = u_E[r]\sigma_E = s$.

Proof of part 1

$t =_E c[(\ell\sigma)']$, so there is a finite chain $c[(\ell\sigma)'] = t_1 =_E t_2 =_E \ldots =_E t_{n-1} =_E t_n = t$, each t_i $(1 \leq i < n)$ is E-equal to t_{i+1} by a single application of an instance of an E-equation. The chain consists of $(n-1)$ applications of instances of E-equations. We proof this lemma by induction on n (i.e. the length of this chain).

- $n = 1$: $(c[(\ell\sigma)'])' = t = c[(\ell\sigma)']$
  Take $D_E := D$, $\sigma_E := \sigma$, $c_E := c$ and p_E is the position of $(\ell\sigma)'$ in t.

- $n \rightsquigarrow n+1$: We consider $t_1 =_E t_2$. The E-equation (for $t_1 =_E t_2$) is applied at position q of t_1 and is a τ-instance of an E-equation $e_1 \approx e_2$, i.e. $t_1 = c_1[e_1\tau]$, $t_2 = c_1[e_2\tau]$ and $c_1 = t_1[q \leftarrow []]$. t_1 and t_2 are the same terms outside of q, i.e. $t_1[q \leftarrow []] = t_2[q \leftarrow []] = c_1$.
  It takes only $(n-2)$ applications of (instances of) E-equations to show $t_2 =_E t$.
  - Case A: The equation $e_1\tau \approx e_2\tau$ is applied below $(\ell\sigma)'$. t_1 and t_2 are the same terms above of $(\ell\sigma)'$ (because they are equal above of q, which is below $(\ell\sigma)'$). $t_2 = c[(\ell\sigma)'']$ and the E-equality of t to t_2 is shown by $(n-2)$ E-steps. So by induction there exists a substitution σ_{ind}, a context c_{ind}, a position p_{ind} and a clause D_{ind} containing the term $u_{ind}[\ell]$ with $(u_{ind}[\ell])\sigma_{ind} =_E t/p_{ind}$, c_{ind} is a prefix of t up to p_{ind} and $I_C \models c_{ind}[(u_{ind}[r])\sigma_{ind}] \approx c[r\sigma]$. Take $D_E := D_{ind}$, $\sigma_E := \sigma_{ind}$, $p_E := p_{ind}$ and $c_E := c_{ind}$.
  - Case B: The equation is not applied below $(\ell\sigma)'$, but at a subterm (of t_2 and t_1) disjoint to it. Then $t_2 = c'[(\ell\sigma)']$ for a context $c' = c[q \leftarrow e_1\tau]$ and we need only $(n-2)$ E-steps to show t_2 E-equal to t. By induction there exists a substitution σ_{ind}, a context c_{ind}, a position p_{ind} and a clause D_{ind} containing the term $u_{ind}[\ell]$ with $(u_{ind}[\ell])\sigma_{ind} =_E t/p_{ind}$, c_{ind} is a prefix of t up to p_{ind} and $I_C \models c_{ind}[(u_{ind}[r])\sigma_{ind}] \approx c'[r\sigma]$. If we prove $I_C \models c'[r\sigma] \approx c[r\sigma]$, then we are done with $D_E := D_{ind}$, $\sigma_E := \sigma_{ind}$, $p_E := p_{ind}$ and $c_E := c_{ind}$. But as the E-equation is applied at a subterm disjoint to $(\ell\sigma)'$, we have $c =_E c'$ and so $c[r\sigma] =_E c'[r\sigma]$ and the applied E-equations to show the last E-equality apply at terms E-equal to $c[r\sigma]$ and $t =_E c[\ell\sigma] >_E c[r\sigma]$, so $I_C \models c'[r\sigma] \approx c[r\sigma]$ follows from our assumption (all instances of E-equation with terms smaller than t are satisfied by I_C).
  - Case C: The equation is not applied below $(\ell\sigma)'$ (otherwise see case A) nor at a subterm disjoint to it (see case B), so applied at a position above of $(\ell\sigma)'$. So there exists a position p in $e_1\tau$ with $e_1\tau/p = (\ell\sigma)'$. We have $p \neq \varepsilon$ (otherwise the application is below $(\ell\sigma)'$, see case A).
    * Case C.1: p is a non-variable position of e_1. The E-equation $e_1 \approx e_2$ has a non-variable position p in e_1 E-unifiable with $\ell\sigma$. We consider the extended clause $D_1 = \Gamma \rightarrow \Delta, e_1[p \leftarrow \ell] \approx e_1[p \leftarrow r]$ of D. With $V := \{x \mid x \in vars(e_1[p \leftarrow []])\}$ we define σ_1 to be the substitution with $y\sigma_1 := y\tau$ for all $y \in V$ and $z\sigma_1 := z\sigma$ for all $z \notin V$. We assume the variables in $e_1 \approx e_2$ and D to be disjoint, so we obviously have $\Gamma\sigma = \Gamma\sigma_1$, $\Delta\sigma = \Delta\sigma_1$, $(\ell\sigma)' = (\ell\sigma_1)'$, $r\sigma = r\sigma_1$.
      We now use the induction hypothesis for D_1 instead of D (particularly $e_1[\ell] \approx e_1[r]$ instead of $\ell \approx r$), σ_1, $e_2\sigma_1 = t_2/q$ instead of $(\ell\sigma)'$ and $c_1 := t_2[q \leftarrow []]$ instead of c. So $t =_E c_1[e_2\sigma_1]$ using $(n-2)$ E-steps. We get a substitution σ_{ind}, a context c_{ind}, a position p_{ind} and a clause D_{ind} containing the term $u_{ind}[e_1[\ell]]$ with $(u_{ind}[e_1[\ell]])\sigma_{ind} =_E t/p_{ind}$, c_{ind} is a prefix of t up to p_{ind} and $I_C \models c_{ind}[(u_{ind}[e_1[r]])\sigma_{ind}] \approx c_1[(e_1[r])\sigma_1]$. We need to show $I_C \models c_1[(e_1[r])\sigma_1] \approx c[r\sigma]$. We have $c[q \leftarrow []] = t_1[q \leftarrow []] = t_2[q \leftarrow []] = c_1$, $c/q = e_1\sigma_1[[]]$ and $c_1/q = []$, hence $c[r\sigma] = c_1[e_1\sigma_1[r\sigma]]$. With $e_1\sigma_1[r\sigma] = (e_1[r])\sigma_1$ (because $r\sigma = r\sigma_1$) and $I_C \models c_{ind}[(u_{ind}[e_1[r]])\sigma_{ind}] \approx c_1[(e_1[r])\sigma_1]$ we conclude $I_C \models c_{ind}[(u_{ind}[e_1[r]])\sigma_{ind}] \approx c[r\sigma]$. So we again set $D_E := D_{ind}$, $\sigma_E := \sigma_{ind}$, $p_E := p_{ind}$ and $c_E := c_{ind}$.

* Case C.2: p is a variable position of e_1. Let x denote this variable and $(\ell\sigma)'$ is a subterm below this variable, i.e. $x\tau = c_x[(\ell\sigma)']$ for an appropriate context c_x. The E-equations are variable preserving so there exists at least one occurrence of $(\ell\sigma)'$ in t_2 below q (say at $e_2\tau/p'$), i.e. there exists a context c_1, $c_1[(\ell\sigma)'] = t_2$ and $c_1[q \leftarrow []] = t_2[q \leftarrow []] = t_1[q \leftarrow []] =: v$. We define a substitution τ_x by $x\tau_x := c_x[r\sigma]$ and $y\tau_x := y\tau$ for all $y \neq x$. For all terms a and b, at least one of them has an occurrence of x, we get $a\tau \approx b\tau >_{eq} a\tau_x \approx b\tau_x$ and $I_C \models a\tau \approx b\tau$ if and only if $I_C \models a\tau_x \approx b\tau_x$ (because $I_C \models (\ell\sigma)' \approx r\sigma$).
We need only $(n-2)$ E-steps to get $t_2 = c_1[(\ell\sigma)'] =_E t$, so by induction there exists a substitution σ_{ind}, a context c_{ind}, a position p_{ind} and a clause D_{ind} containing the term $u_{ind}[\ell]$ with $(u_{ind}[\ell])\sigma_{ind} =_E t/p_{ind}$, c_{ind} is a prefix of t up to p_{ind} and $I_C \models c_{ind}[(u_{ind}[r])\sigma_{ind}] \approx c_1[r\sigma]$. We need $I_C \models c_{ind}[(u_{ind}[r])\sigma_{ind}] \approx c[r\sigma]$ and this will follow from the previous equation and $I_C \models c[r\sigma] \approx c_1[r\sigma]$. We have $c[r\sigma] = v[e_1\tau[p \leftarrow c_x[r\sigma]]]$, replace τ by τ_x (remember $I_C \models (\ell\sigma)' \approx r\sigma$) yielding $I_C \models v[e_1\tau[q \leftarrow c_x[r\sigma]] \approx v[e_1\tau_x[q \leftarrow c_x[r\sigma]]$. $v[e_1\tau_x[q \leftarrow c_x[r\sigma]] = v[e_1\tau_x]$, so $I_C \models c[r\sigma] \approx v[e_1\tau_x]$. Analogously (with c_1, e_2 and p' instead of c, e_1 and p) $I_C \models c_1[r\sigma] \approx v[e_2\tau_x]$. By our assumptions $I_C \models e_1\tau_x \approx e_2\tau_x$ (hence $I_C \models v[e_1\tau_x] \approx v[e_2\tau_x]$). This completes $I_C \models c_1[r\sigma] \approx c[r\sigma]$.

Proof of part 2

We have $t =_E t' = c_1[(u_1[\ell]\sigma)'] =_E c_1[u_1\sigma[\ell\sigma]]$ (for context $c_1 := t'[p_1 \leftarrow []]$). With $c = c_1[u_1\sigma]$ (note $t = (c[\ell\sigma])'$) we apply part 1.

If I_C satisfies $(\ell\sigma)'' \approx r\sigma$ (for all $(\ell\sigma)'' =_E \ell\sigma$), then by part 1 we know $I_C \models c_E[u_E[r]\sigma_E] \approx c[r\sigma]$. The last term is equal to $c_1[u_1\sigma][r\sigma] = c_1[u_1[r]]\sigma = t'[p_1 \leftarrow u_1[r]\sigma]$. So we already have $I_C \models c_E[u_E[r]\sigma_E] \approx t'[p_1 \leftarrow u_1[r]\sigma]$.

Proof of part 3

$t = e\sigma_e$ is reducible at a non-variable position p by a rule produced by $D\sigma$, so with the context $c_1 := t[p \leftarrow []]$ we have $e\sigma_e/p = c_1[(\ell\sigma)']$ and t is reduced to $s := t[p \leftarrow r\sigma] = c_1[r\sigma]$. There exists an extended clause $D_E := \Gamma \rightarrow \Delta, u_E[\ell] \approx u_E[r]$ of D with $u_E = e[p \leftarrow []]$. By combination of σ and σ_e we construct a substitution σ_E such that $u_E\sigma_E = c_1$ and $\ell\sigma_E = \ell\sigma$ (and similar for Γ, Δ and r). We get $u_E[\ell]\sigma_E =_E t$, $u_E[r]\sigma_E = c_1[r\sigma] = s$. □

With the following three examples, we will motivate that three non-obvious technical details of the previous lemma are really needed.

Example 6.32 (Extensions of Extensions)
For general E we really need the induction, so sometimes have to use an extension for a clause D_{ind} which is already an E-extended clause of D. For $E = AC$ we will later prove (similar to the lemma above, but without induction) that we do not need extensions of AC-extensions. Let E consist of

$$(x + y) + z \approx x + (y + z) \qquad \text{(Associativity)}$$
$$u * (v + w) \approx (u * v) + (u * w) \qquad \text{(Distributivity)}$$

Consider the clause

$$D := \rightarrow a + b \approx c.$$

It has an extension with the associativity:

$$D_{ind} := \rightarrow (a + b) + z \approx c + z$$

And D_{ind} itself has again an extension with the distributivity:

$$D_E := \rightarrow u * ((a + b) + z) \approx u * (c + z)$$

There are further extensions (e.g. of D with distributivity or of D_{ind} with associativity) but we do not need them in this example. We consider the following three terms (they are not ground, but we could use an arbitrary ground instance of them)

$$\begin{aligned} t' &:= u * ((a + b) + z) \\ t_2 &:= u * (a + (b + z)) \\ t &:= (u * a) + (u * (b + z)), \end{aligned}$$

which are all E-equal. t' has a subterm which can be matched with the term $a+b$ of D, but neither t_2 nor t contain such a subterm. The term $(a + b) + z$ of D_{ind} can match subterms of t' and t_2, but not of t. And finally for the left side of D_E we notice that in all of the three terms there is a subterm we can match (always the trivial subterm at $p = \varepsilon$). So to superpose upon t, we need D_E, which is an extension of an extension.

Example 6.33 The assumptions $I_C \models (\ell\sigma)' \approx r\sigma$ and I_C *satisfies all ground instances of E-equations between terms smaller than t* are both needed. But the first one only for non-linear equations in E. Let E consist of the associativity and commutativity axioms for the operator $+$ and the idempotence (the non-linear equation $x + x \approx x$) for $+$. We again consider

$$D\sigma := \rightarrow a + b \approx c,$$

so speaking in terms of lemma 6.31 we have $\ell\sigma := a+b$ and $r\sigma := c$. Let us identify $D\sigma$ with the rule $a+b \Rightarrow c$ and speak of reductions of terms. Let C be a clause such that I_C contains the above rule, which we assume to be produced by $D\sigma$. The term $t' := (a + b) + (b + a)$ can be reduced to $c + (b + a)$. Here we have no problems to reduce also $t := a + b$. t is not reduced to $c + (b + a)$, but to c. To get $I_C \models c + (b + a) \approx c$ we need $I_C \models b + a \approx c$ and $I_C \models c + c \approx c$. Both equations are valid in I_C by the assumptions of lemma 6.31 repeated at the beginning of this example.

Example 6.34 The $'$ in $t =_E c[(\ell\sigma)']$ is needed in lemma 6.31 (for induction in part 1). Let $+$ be an AC-operator and

$$D\sigma := \rightarrow (a + b) + c \approx d$$

(hence $\ell\sigma := (a + b) + c$). We consider the following three terms

$$\begin{aligned} \ell\sigma &:= (a + b) + c \\ t_2 &:= (b + a) + c \\ t &:= c + (b + a). \end{aligned}$$

$\ell\sigma$ and t are AC-equal by a two step equational proof $\ell\sigma =_{AC} t_2 =_{AC} t$. To use induction in the proof of lemma 6.31, we have to apply the lemma to $(\ell\sigma)' = t_2 \neq \ell\sigma$. So it has to be formulated for $(\ell\sigma)'$ and not simply for $(\ell\sigma)$.

Lemma 6.35 Let $D_1 = \Gamma \rightarrow \Delta, t \approx s$ be a clause in $NE \cup Ext$. If $D_1\sigma$ produces a rule $(t\sigma)' \Rightarrow s\sigma$, then there exists a clause D in NE and a position p such that $D = \Gamma \rightarrow \Delta, \ell \approx r$, D_1 is in $Ext^*(D, \ell \approx r)$, $t/p = \ell$ and $s/p = r$.

Proof: If $D_1 \in NE$ use $D := D_1$ and $p := \varepsilon$. Otherwise use lemmata 6.19 and 6.12. □

When we apply part 2 of lemma 6.31 to a productive clause $D_1\sigma$, then by the previous lemma we can always assume that D is a clause in NE, i.e. D is not an extended clause. In part 2 of lemma 6.31 we sometimes need $I_C \models (\ell\sigma)'' \approx r$ (cf. example 6.33). For some theories E this is not a trivial assumption:

Example 6.36 $E = \{f(f(x)) \approx f(x)\}$. The clause $\rightarrow f(f(a)) \approx b$ (of NE) will produce only one rule $f(a) \Rightarrow b$ and the equation in the clause will (in general) not be satisfied in interpretations of greater clauses. We assume that with the help of extended clauses such situations do not occur for saturated sets of clauses. But to show this will complicate the following proofs, and to show the property for all theories E will even be harder. As very little is known about general E (existence of total and compatible reduction orderings, existence of usable unification algorithms), we will restrict the methods such that the well known and promising cases $E = AC$ and $E = ACU$ are included.

Definition 6.37 (Restrictions on E)
In the remaining parts of this section we consider only equational theories E with the following property: If a ground term $u[t]$ is E-equal to a strict subterm t of itself, then we have $u[s] =_E s$ for all ground terms s.

We will need the above restriction in lemmata 6.39, 6.43 and 6.47.

Example 6.38 The theory $ACU := \{x + y \approx y + x, (x + y) + z \approx x + (y + z), x + 0 \approx x\}$, i.e. $+$ is an AC-operator with unit 0, fulfills the above restrictions.

With the help of the following lemma we conclude that the assumption about I_C stated in part 2 of lemma 6.31 (we mean $I_C \models (\ell\sigma)'' \approx r\sigma$) is always met. Note that we need the above restriction for the following proof.

Lemma 6.39 (Properties of Productive Clauses 1)
Let $G := \Gamma \rightarrow \Delta, t \approx s$ be a clause of NEG with strictly maximal equation $t \approx s$, $t >_E s$ and let G be productive. Let C be a clause such that I_C contains the rules produced by G (e.g. if $C >_c^{ext} G$). If I_C satisfies all instances of E-equations between terms smaller than t, then $I_C \models t' \approx s$ for all terms t' with $t' =_E t$, i.e. the productive equation and all equations with E-equal left side will be satisfied in interpretations containing the produced rules.

Proof: t and every term t' E-equal to it is irreducible by R_G (required for productive clauses by part 4 of definition 6.17). Every t' E-equal to t is reducible with the produced rules in $irred\_rules(t \approx s, R_G)$ (note: these rules are contained in I_C). Hence there exists a rule $t'' \Rightarrow s$ with $t' = u[t'']$ and t' is reduced to $u[s]$. If u is the empty context we have $t' = t''$, $u[s] = s$ and are done. Otherwise $t' = u[t''] =_E u[t']$, hence t' is E-equal to a proper subterm of itself and by definition 6.37 we have also $u[s] =_E s$. We reduce t' to $u[s]$ and because this term is smaller than t the instances of E-equations to show $u[s] \approx s$ are satisfied by I_C. Together with the reduction step we have $I_C \models t' \approx s$. □

With the same assumptions as in the previous lemma, I_C even satisfies all equations $t' \approx s'$. The lemma proves it for $s = s'$ but requires also $I_C \models s' \approx s$, hence $I_C \models t' \approx s'$.

Lemma 6.40 (Properties of Productive Clauses 2)
Let $G := \Gamma \to \Delta, \ell \approx r$ be a ground clause in NG. If G is productive, it is non-redundant.

Proof: If G is of $ExtG$, it is non-redundant by definition. Otherwise $G \in NEG$. G is redundant, only if it is satisfied in its own interpretation. G is productive and we have $I_G \not\models \Gamma \to \Delta$. So G is only true in I_G if $\ell \approx r$ is satisfied. If G is productive we have $\ell >_E r$ and so (because the interpretation is a Church-Rosser rewrite system, cf. lemma 6.24) ℓ has to be reducible (if $\ell \approx r$ should be satisfied), which contradicts part 4 in definition 6.17, hence G is non-redundant. □

Lemma 6.41 Let C and D be ground clauses in NEG with maximal (w.r.t. $>_E$) term E-equal to t. If $C >_c^{ext} D$ and t is irreducible by $\Longrightarrow_{R_C}$, then $R_C = R_D$ and $I_C = I_D$.

Proof: If a clause G with $C >_c^{ext} G \geq_c^{ext} D$ produces rules contained in R_C, all terms E-equal to t become reducible. As t is irreducible by $\Longrightarrow_{R_C}$, no rule is produced and so we have $R_C = R_D$. Because the maximal terms of C and D are E-equal and both clauses are in NEG we have $E_C^{all} = E_D^{all}$. As these clauses define the same rewrite system, $E_C = E_D$ and so $I_C = I_D$. □

Lemma 6.42 (Properties of Non-Redundant Clauses)
Let $F := D\sigma := \Gamma \to \Delta, t \approx s$ be a non-redundant ground instance of a clause D in NE. Let the equation $t \approx s$ be maximal (w.r.t. $>_{eq}$) in F with $t >_E s$, and let t' be irreducible by $\Longrightarrow_{R_F}$ (for all t' with $t' =_E t$). If N is $\mathcal{M}_{Ext}$-saturated on $NG_F \cup \{F' \mid F' = D\sigma', \sigma' =_E \sigma\}$ and if I_F satisfies all ground instances of E-equations between terms smaller than t, the following holds:

(i) σ is irreducible (w.r.t. $\Longrightarrow_{R_F}$), i.e. $x\sigma$ is irreducible for all $x \in vars(D)$

(ii) F is productive.

(iii) For all Γ' with $\Gamma' =_E \Gamma$ and all ground clauses $C >_c^{ext} F$ or $C =_E F$,

$$\Gamma' \subseteq I_C \ .$$

(iv) For all Δ' with $\Delta' =_E \Delta$ and all ground clauses $C >_c^{ext} F$ or $C =_E F$,

$$\Delta' \cap I_C = \emptyset \ .$$

Proof: The proof is by induction on the ordering $>_c^{ext}$, so let us assume that (i)–(iv) hold for every suitable instance F_1 of NE with $F >_c^{ext} F_1$. Since F is non-redundant we have $\Gamma \subseteq I_F$ and $\Delta \cap I_F = \emptyset$.

(iii) $I_F \not\models \Gamma \to \Delta$, so Γ is satisfied in I_F. By our assumptions all instances of E-equations applicable at terms in the antecedent (which are all smaller (w.r.t. $>_E$) than t) are contained in I_F, hence $\Gamma' \subseteq I_F$. All rules and equations in the difference between R_C and R_F and between E_C and E_F contain a maximal term ℓ with $\ell \geq_E t$ (from $C >_c^{ext} F$ or $C =_E F$). These rules and equations cannot be used to join an equation in Γ', hence I_C, too, satisfies every equation in Γ'.

(i) Suppose there exists a variable $x \in vars(D)$ such that $x\sigma$ is reducible by $\Longrightarrow_{R_F}$ (note that $x\sigma$ cannot occur in t, as t is irreducible by our assumptions). By lemma 6.30 there exists a ground instance $F_1 = D\tau = \Gamma_1 \rightarrow \Delta_1, t \approx s_1$ such that $F >_c^{ext} F_1$ and $I_F \not\models F_1$. By lemma 6.41 we have $I_F = I_{F_1}$, which implies that F_1 is false in its interpretation and hence non-redundant. We use our induction hypothesis and infer that F_1 produces rules to reduce t and all terms E-equal to it (induction hypothesis part (ii)). But these rules are contained in R_F contradicting the irreducibility of t. So F is an instance with a reduced substitution (and we assume this fact in the remaining parts of the proof).

(ii) If F is not productive because an equation $t'' \approx s'$ E-equal to (but different from) the maximal equation $t \approx s$ of F occurs in the succedent of F. Then the equation is not strictly maximal and so the clause is not productive for this reason. Moreover $I_F \models s \approx s'$. $t >_E s$ and so the needed E-equalities are by assumption contained in I_F. A clause $F_1 = \Gamma', s' \approx s'' \rightarrow \Delta'$ smaller than F can be obtained by equality factoring: as conclusion of the instance of an equality factoring inference with premise $F' = D\sigma'$. We have $I_F = I_{F'}$ (lemma 6.18). By saturation F_1 has to be satisfied by $I_F = I_{F'}$. This yields a contradiction: Γ' is satisfied, see above; if an equation $u' \approx v'$ with $t >_E u$ and $t >_E v$ is satisfied, then also $u \approx v$ is satisfied and F redundant; all equations $t'' \approx r$ in Δ' with terms E-equal to t are not satisfied because t and all terms E-equal to it are irreducible. So from now on we may assume that no other equation E-equal to $t \approx s$ occurs in the succedent of F and hence that F is productive.

(iv) $I_F \not\models \Gamma \rightarrow \Delta$, in particular $I_F \cap \Delta = \emptyset$. Suppose an Δ' contains an equation $u \approx v$ which is satisfied by I_F. We may assume that $u >_E v$, for otherwise, if $u =_E v$ then either $t >_E u$ and $u' \approx v'$ is also satisfied by I_F (by the assumption the needed instances of E-equations are contained in I_F), which contradicts F to be non-redundant, or we have $u =_E v =_E t$ and the equation $t \approx s$ with $t >_E s$ is not a maximal equation of F, which also contradicts our assumptions. By construction I_F is a Church-Rosser system, so u and v are reducible (by R_F) to E_F-equal terms. If we have $t >_E u$ and $I_F \models u \approx v$, then I_F also satisfies $u' \approx v' \in \Delta$ (the needed instances of E-equations are again available by our assumptions), which contradicts F to be non-redundant. If we have $t =_E u$, then u is irreducible (by our assumptions) and $I_F \not\models u \approx v$. So no equation in Δ' can be satisfied by I_F.

Suppose an Δ' contains an equation $u \approx v$ which is satisfied by an I_C (with $C >_c^{ext} F$ or $C =_E F$).

We have $t =_E u >_E v$: for $t =_E v >_E u$ rename u and v; if we have $t >_E u$ and $t >_E v$ then also I_F will satisfy $u \approx v$ contradicting the result above (note that all rules of $R_C \setminus R_F$ and all equations of $E_C \setminus E_F$ contain a maximal term not smaller than t, so cannot be used to join u and v); $t =_E u =_E v$ contradicts the maximality of $t \approx s$ in F.

By construction all interpretations are Church-Rosser systems, so u and v are reducible with R_C to E_F-equal terms (the terms cannot grow by reduction, so the needed instances of E-equations are also satisfied in I_F).

A rule in R_F cannot reduce $u =_E t$, because every term E-equal to t is irreducible (our assumption). All rules in $R_C \setminus R_F$ (if any) have left sides not smaller than t. Rules with left side greater (w.r.t. $>_E$) than t cannot reduce u, so a rule with left side E-equal to t reduces u. With (ii) we already know that F is productive and no other clause can produce a rule whose left side is E-equal to t and which is not contained in $irred\_rules(t \approx s, R_F)$ (this is the set of rules produced by F). So the rules produced by F are contained in R_C.

The equation $u \approx v$ (remember that $u =_E t$) is rewritten to $r \approx v$ and the sides r and v are reducible to E_F-equal terms. Because $t >_E r$ we can only use the rules of R_F to reduce r or v, so $I_F \models r \approx v$. We have $s \geq_E v$ (maximality of the equation) and by part (ii) (see proof above) even $s >_E v$.

Above we have supposed $I_C \models u \approx v$. With $I_C \models u' \approx s$ (by lemma 6.39, note $u' =_E t''$ and the rules produced by F are contained in I_C, see above) and the reduction from u' to r we have $I_C \models s \approx r$. As the rules needed to join s and r are already contained in I_F we combine this with $I_F \models r \approx v$ and $I_F \models v \approx v'$ to $I_F \models s \approx v'$ (and by our assumptions also $I_F \models s'' \approx v''$). We consider the *equality factoring* inference π

$$\frac{\Gamma \to \Delta, t \approx s}{\Gamma, s \approx v' \to \Delta},$$

which can be lifted (and then instantiated) to an E-equal inference with premise F'. Let us denote the conclusion $\Gamma'', s'' \approx v'' \to \Delta''$ of the lifted and then instantiated inference by F_1. With $I_F \models s'' \approx v''$ and $I_F \not\models F$ we conclude $I_F \not\models F_1$. Lemma 6.18 implies $I_{F'} \not\models F_1$. This contradicts the saturation, so there cannot be an equation in Δ' which is satisfied in I_C. □

Lemma 6.43 (Complexity of Extended Clauses)
Let $D_E \in Ext$ be an extended clause in $Ext^*(D, \ell \approx r)$. Let $D = \Gamma \to \Delta, \ell \approx r$ and $D_E = \Gamma \to \Delta, u[\ell] \approx u[r]$. Let σ be a ground substitution such that $D_E\sigma \neq_E D\sigma$, the equation $\ell\sigma \approx r\sigma$ is strictly maximal (w.r.t. $>_{eq}$) in D and $\ell\sigma >_E r\sigma$. Then $D_E\sigma$ is smaller than $\to (u[\ell])\sigma \approx (u[\ell])\sigma$ (i.e. the case (ii) of definition 6.13 of $>_{eq}^{ext}$ applies to the maximal equation of $D_E\sigma$).

Proof: We have to show that case (ii) of definition 6.13 applies to $(u[\ell] \approx u[r])\sigma$. Only one requirement is not obviously satisfied: the term $(u[\ell])\sigma$ has to be strictly maximal in $D_E\sigma$. If $(u[\ell])\sigma =_E \ell\sigma$, then with our restrictions 6.37 we also have $(u[r])\sigma =_E r\sigma$ and both clauses are E-equal. We exclude this by our assumption $D\sigma \neq_E D_E\sigma$. So we have $(u[\ell])\sigma >_E \ell\sigma$. From $\ell\sigma >_E r\sigma$ we conclude $(u[\ell])\sigma >_E (u[r])\sigma$. As all terms in $\Gamma\sigma$ and $\Delta\sigma$ are not greater than $\ell\sigma$ (because $\ell\sigma \approx r\sigma$ is strictly maximal in $D\sigma$), we conclude that $(u[\ell])\sigma$ is indeed the maximal term of $D_E\sigma$. □

Lemma 6.44 (Interpretations are E-Models)
Let $N := NE \cup Ext \cup SExt$ be a set of clauses. Let C be a ground clause in NG. If N is $\mathcal{M}_{Ext}$-saturated on NG_C and closed under extension, then:

$$E_C^{all} \subseteq I_C \ .$$

Proof: We use induction on the size of clauses. For all clauses CC which are smaller than C we use $E_{CC}^{all} \subseteq I_{CC}$ as an induction hypothesis. Let C be a ground clause with maximal term t_{max}. The proof consists of two main cases:

Case A:
All instances $e_1\sigma \approx e_2\sigma$ of E-equations with $t_{max} >_E e_1\sigma$ are contained in I_C:
With the clause $TT := e_1\sigma \approx e_1\sigma \to e_1\sigma \approx e_1\sigma$ (note $C >_c^{ext} TT$) we conclude $e_1\sigma \approx e_2\sigma \in E_{TT}^{all} \subseteq I_{TT} \subseteq I_C$ (definition of E_{TT}^{all}; induction hypothesis for TT; rules which reduce such instances of E-equations cannot be produced by TT or any greater clause).

Case B:
All instances $e_1\sigma \approx e_2\sigma$ of E-equations with $t_{max} =_E e_1\sigma$ are contained in I_C, if C is greater than $\rightarrow t_{max} \approx t_{max}$ (or E-equal to it), i.e. if the equation is contained in E_C^{all}:

Case B1: $e_1\sigma$ and $e_2\sigma$ are both irreducible (w.r.t. $\Longrightarrow_{R_C}$):
We have $e_1\sigma \approx e_2\sigma \in E_C \subseteq I_C$ (if the equation is contained in E_C^{all}).

Case B2: $e_1\sigma$ or $e_2\sigma$ is reducible:

Case B2.1: a variable x in e_1 or e_2 and $x\sigma$ is reducible:
By lemma 6.30 we conclude $e_1\sigma \approx e_2\sigma \in I_C$ (otherwise the lemma provides a smaller instance which is false in I_C, this contradicts case A).

Case B2.2: no reducible variable in e_1 and e_2:
There exists a rule $(c_1[\ell_1])' \Rightarrow c_1[r_1]$ produced by an instance $D_1\sigma := \Gamma_1 \rightarrow \Delta_1, c_1[\ell_1] \approx c_2[r_1]$ of a clauses D_1 with $C >_c^{ext} D_1\sigma$. So there exists a clause $G_1 \in NE$ such that D_1 is in $Ext^*(G_1)$ with $G_1\sigma = \Gamma_1 \rightarrow \Delta_1, \ell_1 \approx r_1$ and $C >_c^{ext} G_1\sigma$ (lemma 6.35). The rule reduces $e_1\sigma$ at position p_1 to a term s_1. The position p_1 is a non-variable position of e_1. By lemma 6.31 (part 3 of the lemma) there exists a clause D_{1E} (in general an extended clause originated from G_1) and a substitution σ_1 such that $D_{1E}\sigma_1 = \Gamma_1 \rightarrow \Delta_1, u_1[\ell_1] \approx u_1[r_1]$, $e_1\sigma =_E u_1[\ell_1]\sigma_1$ and $I_C \models u_1[r_1]\sigma_1 \approx s_1$.

The maximal term of $D_{1E}\sigma_1$ is E-equal to t_{max}. For $D_{1E}\sigma_1 =_E G_1\sigma$ we do not need D_{1E} (use G_1) and have already $C >_c^{ext} G_1\sigma$. For $D_{1E}\sigma_1 \neq_E G_1\sigma$ we know that D_E is an extended clause, hence smaller than C (see lemma 6.43). From the existence of such an extended clause (or the fact that a clause of NEG produces rules with left side E-equal to t_{max}), we conclude that all terms E-equal to t_{max} are reducible in I_C. Note that for this conclusion we need that E-extended clauses produce rules. In particular $e_2\sigma$ is reducible ($e_1\sigma$ and $e_2\sigma$ are E-equal). So there exists a rule $(c_2[\ell_2])' \Rightarrow c_2[r_2]$ in R_C produced by an instance $D_2\sigma := \Gamma_2 \rightarrow \Delta_2, c_2[\ell_2] \approx c_2[r_2]$ of a clause D_2 with $C >_c^{ext} D_2\sigma$. So there exists a clause $G_2 \in NE$ such that D_2 is in $Ext^*(G_2)$ with $G_2\sigma = \Gamma_2 \rightarrow \Delta_2, \ell_2 \approx r_2$ and $C >_c^{ext} G_2\sigma$ (lemma 6.35). The rule reduces $e_2\sigma$ at position p_2 to s_2 (we here use the same substitution for all instances, such a substitution can always be constructed by proper renaming). The position p_2 is a non-variable position of e_2. By lemma 6.31 there exists a clause D_{2E} and a substitution σ_2 such that $D_{2E}\sigma_2 = \Gamma_2 \rightarrow \Delta_2, u_2[\ell_2] \approx u_2[r_2]$, $e_2\sigma =_E u_2[\ell_2]\sigma_2$ and $I_C \models u_2[r_2]\sigma_2 \approx s_2$. $e_1\sigma$ and $e_2\sigma$ are E-equal, so we may superpose $D_{1E}\sigma_1$ and $D_{2E}\sigma_2$ (both terms are reducible at a non-variable position, so neither e_1 nor e_2 is simply a variable). Also $D_2\sigma$ and $G_2\sigma$ are smaller than C.

If we have $D_{1E}\sigma_1 =_E D_{2E}\sigma_2$, then $I_C \models s_1 \approx u_1[r_1]\sigma_1$, $u_1[r_1]\sigma_1 =_E u_2[r_2]\sigma_2$ (so $I_C \models u_1[r_1]\sigma_1 \approx u_2[r_2]\sigma_2$ by case A) and $I_C \models u_2[r_2]\sigma_2 \approx s_2$. Together with the reduction steps (also contained in I_C), we get $I_C \models e_1\sigma \approx e_2\sigma$ and are done.

Otherwise ($D_{1E}\sigma_1 \neq_E D_{2E}\sigma_2$) we get one of the following superposition right inferences:

$$\frac{\Gamma_1 \rightarrow \Delta_1, u_1[\ell_1]\sigma_1 \approx u_1[r_1]\sigma_1 \quad \Gamma_2 \rightarrow \Delta_2, u_2[\ell_2]\sigma_2 \approx u_2[r_2]\sigma_2}{\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2, u_1[r_1]\sigma_1 \approx u_2[r_2]\sigma_2}$$

or

$$\frac{\Gamma_2 \rightarrow \Delta_2, u_2[\ell_2]\sigma_2 \approx u_2[r_2]\sigma_2 \quad \Gamma_1 \rightarrow \Delta_1, u_1[\ell_1]\sigma_1 \approx u_1[r_1]\sigma_1}{\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2, u_1[r_1]\sigma_1 \approx u_2[r_2]\sigma_2},$$

which have the same conclusion (only the order of premises changes). Both premises are either productive clauses or instances of extended clauses so non-redundant. The inference

can be lifted and instantiated to an E-equal ground inference with conclusion $\Gamma_1', \Gamma_2' \rightarrow \Delta_1', \Delta_2', (u_1[r_1]\sigma_1)' \approx (u_2[r_2]\sigma_2)'$. Also premises of the lifted and instantiated inference are non-redundant. By saturation this inference has to be redundant, i.e. the conclusion has to be satisfied (in the interpretation of the maximal premise). $G_1\sigma$ and $G_2\sigma$ are productive. By case A and our induction hypothesis we know that the interpretations of these clauses satisfy all ground instances of E-equations between terms smaller than their maximal term. By lemma 6.40 they are non-redundant, so we apply lemma 6.42 and conclude that Γ_1' and Γ_2' are satisfied but no equation in Δ_1' or Δ_2' can by true. So the conclusion is satisfied only if the equation $(u_1[r_1]\sigma_1)' \approx (u_2[r_2]\sigma_2)'$ is true, i.e. both sides are reducible to E-equal terms. These sides are both smaller than $e_1\sigma =_E t_{max}$, so the needed rules are already contained in I_C. With case A we know that I_C satisfies this equation and the E-equal one $u_1[r_1]\sigma_1 \approx u_2[r_2]\sigma_2$. But I_C also provides the rules and equations to show these terms equal to s_1 resp. s_2 (see above; it is a conclusion from lemma 6.31) and the rules to reduce $e_1\sigma$ to s_1 and $e_2\sigma$ to s_2, so $I_C \models e_1\sigma \approx e_2\sigma$.

As there are no equations $e_1 \approx e_2$ in E_C^{all} with $e_1 >_E t_{max}$ (definition of E_C^{all}), cases A and B imply $E_C^{all} \subseteq I_C$. □

Now we are able to prove lemmata similar to 5.15 and 5.19, but now for $\mathcal{M}_{Ext}$-saturated and closed sets of clauses.

Lemma 6.45 (Monotonicity of Interpretations)
Let C and D be ground clauses of NG with $C >_c^{ext} D$. If N is $\mathcal{M}_{Ext}$-saturated on NG_C and closed under E-extension, then we have $I_C \supseteq I_D$.

Proof: By construction we have $R_C \supseteq R_D$. By the previous lemma all equations in E_D are also satisfied in I_C ($E_D \subseteq E_D^{all} \subseteq E_C^{all} \subseteq I_C$; first inclusion by definition, second one by $C >_c^{ext} D$, third one by lemma 6.44). □

Lemma 6.46 Let N be $\mathcal{M}_{Ext}$-saturated and closed under E-extension. Let B, C and D be ground clauses of NG with $D >_c^{ext} C$ and $C >_c^{ext} B$ or $C =_E B$. If B is true in I_C then it remains true in I_D.

Proof: If an equation in the succedent is satisfied apply lemma 6.45. Otherwise an equation in the antecedent is not satisfied and remains so by lemma 6.26. □

With restrictions 6.37 we can prove that extended clauses are always greater than the clauses they extend (or E-equal to them):

Lemma 6.47 Let $D_E\sigma$ be a ground instance of $D_E \in Ext \cup SExt$. Then there exists a clause D in NE such that $D_E\sigma >_c^{ext} D\sigma$ or $D_E\sigma =_E D\sigma$ and $D_E\sigma$ is satisfied, if $D\sigma$ is satisfied.

Proof: Let D_E be the clause $\Gamma \rightarrow \Delta, t \approx s$ and $t \approx s$ an extended equation in D_E.

First we consider the case $D_E \in Ext$: By lemma 6.12 we know $D_E \in Ext^+(D)$ (for a clause $D \in NE$), $D = \Gamma \rightarrow \Delta, \ell \approx r$, $t/p = \ell$, $s/p = r$ and $p \neq \varepsilon$. So the comparison of $D_E\sigma$ and $D\sigma$ depends only on the equations $t\sigma \approx s\sigma$ and $\ell\sigma \approx r\sigma$.

- Case 1: $\ell\sigma >_E r\sigma$
  Then (by stability against contexts) we have $t\sigma >_E s\sigma$.

- Case 1.1: $t\sigma =_E \ell\sigma$
  If we have $t\sigma =_E \ell\sigma = t\sigma/p$, i.e. t is E-equal to a subterm of itself, then by restrictions 6.37 we know $s\sigma =_E r\sigma$ and so $D_E\sigma$ and $D\sigma$ are E-equal.
- Case 1.2: $t\sigma >_E \ell\sigma$
  The extended equation is greater than $\ell\sigma \approx r\sigma$ because $t\sigma >_E \ell\sigma >_E r\sigma$.
- Case 2: $\ell\sigma =_E r\sigma$
  Then we have also $t\sigma =_E s\sigma$.
- Case 2.1: $t\sigma =_E \ell\sigma$
  All considered terms are E-equal, so $D_E\sigma$ and $D\sigma$ are E-equal.
- Case 2.2: $t\sigma >_E \ell\sigma$
  The extended equation is greater than $\ell\sigma \approx r\sigma$ because $t\sigma >_E \ell\sigma =_E r\sigma$.

So we know $D_E\sigma >_c^{ext} D\sigma$ or $D_E\sigma =_E D\sigma$.

If $D\sigma$ is true, because its antecedent is not satisfied, then the same holds for $D_E\sigma$ (they have the same antecedent). Also if an equation in $\Delta\sigma$ is satisfied, the same equation in the succedent of $D_E\sigma$ is satisfied. We can write $t\sigma \approx s\sigma$ as $u[\ell\sigma] \approx u[r\sigma]$, so this equation is satisfied, if $\ell\sigma \approx r\sigma$ is true. In any case $D_E\sigma$ is satisfied, if $D\sigma$ is satisfied.

Now we consider $D_E \in SExt$. Similar to lemma 6.12 we can find a clause $D \in NE$ such that D_E extends some (at least two) equations of the succedent of D. Similar to the paragraph before, if D is satisfied then any extended clause originated from D is also satisfied. That implication does not depend on the number of equations which are extended from D to D_E. If $D\sigma$ is not smaller than $D_E\sigma$, then it is E-equal to it (for any term t of $D\sigma$ there exists a term $c[t]$ occurring in $D_E\sigma$). So the lemma holds for extended clauses in $SExt$. □

Lemma 6.48 Let $N := NE \cup Ext \cup SExt$ be set of clauses and C be a ground clause in NG. If N is $\mathcal{M}_{Ext}$-saturated on NG_C, closed under extension and does not contain the empty clause, then I_C satisfies all clauses $D\sigma \in NG$ smaller than C.

Proof: Let C be a ground clause with maximal term t_{max}. In the following proof we consider an instance $H\sigma$ of a clause H in N, such that $C >_c^{ext} H\sigma$ and $C \neq_E H\sigma$. Due to the definition of our ordering we also have $C >_c^{ext} H\sigma'$ for all σ' with $\sigma =_E \sigma'$. As N is saturated on NG_C, all inferences with such premises $H\sigma'$ have to be redundant.

We derive a contradiction from the fact that there exists a minimal (w.r.t. $>_c^{ext}$) ground clause D_{false}, $C >_c^{ext} D_{false}$, D_{false} different from the empty clause, $D_{false} \in NEG$ and D_{false} is not satisfied by I_C. By lemma 6.47, then also all clauses in $(ExtG \cup SExtG) \cap NG_C$ are satisfied by I_C. Hence we may assume $D_{false} \in NEG$.

If the clause D_{false} is false in C, it is also false in its own interpretation $I_{D_{false}}$ (otherwise we get a clause which is true in its interpretation and false in the interpretation of a greater clause, this contradicts lemma 6.46). So D_{false} is non-redundant. Similarly we conclude $D'_{false} := D\sigma'$ to be non-redundant and $I_C \not\models D'_{false}$ (by the fact that $D_{false} \in NEG$ and lemma 6.44 we conclude that $I_{D_{false}} = I_{D'_{false}}$ and I_C satisfy all E-equalities needed to show $D_{false} =_E D'_{false}$). If D_{false} is productive, then by lemma 6.39 it is satisfied by I_C. So we may assume that D_{false} is non-redundant and not productive. If for a variable x of D, $x\sigma'$ is reducible by R_C (for any $\sigma' =_E \sigma$), then we can construct a smaller false clause using lemma 6.30. This contradicts the minimality of D_{false}. So we also assume that D_{false}

and every $D\sigma'$ are reduced ground instances. It follows a case analysis depending on the maximal equation in D_{false}.

a) If there is a trivial (maximal) equation $t \approx t'$ in the succedent, D_{false} is satisfied because the needed E-equality is contained in I_C (see lemma 6.44 and definition 6.15).

b) Let $D_{false} = \Gamma\sigma \to \Delta\sigma, t_1\sigma \approx t_2\sigma$ be an instance of $D = \Gamma \to \Delta, t_1 \approx t_2 \in N$ with maximal equation $t_1\sigma \approx t_2\sigma$, $t_1\sigma >_E t_2\sigma$.

b1) The equation $t_1\sigma \approx t_2\sigma$ is strictly maximal in D_{false}:
Since $I_{D_{false}} \not\models D_{false}$, the clause D_{false} has not produced the rule $t_1\sigma \Rightarrow t_2\sigma$. This can only be the case if $(t_1\sigma)'$ is reducible by $R_{D_{false}}$ (see lemma 6.42; note that D_{false} is non-redundant and not productive, the only requirement D_{false} does not fulfill is the irreducibility). Let $(t_1\sigma)'$ be reducible by $R_{D_{false}}$ with a rule $(u_1[s_1]\sigma_1)' \Rightarrow u_1[s_2]\sigma_1$ produced by a clause $C_1\sigma_1 \in NG_C$ smaller than D_{false}, $C_1 = \Gamma_1 \to \Delta_1, u_1[s_1] \approx u_1[s_2]$ and $(t_1\sigma)'/p =_E s_1\sigma_1$. By lemma 6.35 productive clauses have always this form and there exists a clause $G := \Gamma \to \Delta, s_1 \approx s_2$ in NE such that $G\sigma_1$ is productive and $D_{false} >_c^{ext} G\sigma_1$ and $C_1 \in Ext^*(G, s_1 \approx s_2)$. By lemma 6.31 (part 2) there exists a clause $C_E := \Gamma_1 \to \Delta_1, u_E[s_1] \approx u_E[s_2]$, $C_E \in Ext^*(G)$, a substitution σ_E and a position u with $u_E[s_1]\sigma_E =_E t_1\sigma/u$, $\Gamma_1\sigma_1 = \Gamma_1\sigma_E$, $\Delta_1\sigma_1 = \Delta_1\sigma_E$, $s_1\sigma_1 = s_1\sigma_E$, $s_2\sigma_1 = s_2\sigma_E$ and $I_C \models t_1\sigma[u \leftarrow u_E[s_2]\sigma_E] \approx (t_1\sigma)'[p \leftarrow u_1[s_2]\sigma_1]$. If u is at a variable position of t (or below a variable position: then use an additional context in the following), then with $x\sigma' := u_E[s_1]\sigma_E$ (we assume the considered variable is denoted by x) we can define a substitution σ' such that $D\sigma'$ is reducible below the variable x (note that $s_1\sigma_1$ is reducible because $G\sigma_1$ is productive). But all $D\sigma'$ are reduced instances (see above), so u is a non-variable position.
The equation $u_E[s_1]\sigma_E \approx u_E[s_2]\sigma_E$ is the extension of a productive equation, so satisfied by $I_{D_{false}}$ (the clause $G\sigma_1$ containing the productive equation is smaller than D_{false}). If the equations $t_1\sigma \approx t_2\sigma$ and $u_E[s_1]\sigma_E \approx u_E[s_2]\sigma_E$ are E-equal, then by lemma 6.44 and definition 6.15 I_C satisfies also the E-equal equation $t_1\sigma \approx t_2\sigma$ and D_{false} would be true (contradicting that D_{false} is false). Otherwise one of the equations is greater than the other and we consider the *strict superposition right* inference between D_{false} and $C_E\sigma_E$:

$$\pi = \frac{\Gamma_1\sigma_E \to \Delta_1\sigma_E, (u_E[s_1] \approx u_E[s_2])\sigma_E \quad D_{false}}{\Gamma\sigma, \Gamma_1\sigma_1 \to \Delta\sigma, \Delta_1\sigma_1, t_1\sigma[u \leftarrow u_E[s_2]\sigma_E] \approx t_2\sigma}$$

or the above inference with exchanged premises (if $u_E[s_1]\sigma_E \approx u_E[s_2]\sigma_E$ is greater than $t_1\sigma \approx t_2\sigma$). Note that the clause $C_E\sigma_E$ is smaller than C (see lemma 6.43; in the case where C is itself an extended clause with smaller complexity for its maximal equation, we may need the fact that the maximal term of D_{false} and therefore also the maximal term of $C_E\sigma$ is smaller than the maximal term of C, otherwise the NEG-clause D_{false} would be greater than the $ExtG$-clause C).
We denote the conclusion by D_π. The inference π can be lifted (cf. lemma 4.12), and we can instantiate the lifted inference to an inference π' E-equal to π. We show that there is exactly one equation in the succedent of $D_{\pi'}$ of π' which can be satisfied by the interpretation of its maximal premise. We may assume that C_E is either G or an extension of a clause $G_1 \in Ext^*(G)$ such that $G\sigma_1 \neq_E G_1\sigma_1$ (otherwise we can use

G_1 instead of C_E; if G_1 is also not appropriate we iterate the process). In both cases we get $D'_{false} >_c^{ext} C_E\sigma'$ (if C_E is really an extended clause note that $D_{false} \in NEG$, so D'_{false} is greater than $C_E\sigma_E$ because of the smaller complexity of the extended equation in $C_E\sigma_E$). So the maximal premise is D'_{false}.

- $\Gamma\sigma' \subseteq I_{D'_{false}}$:
  $\Gamma\sigma'$ is satisfied by $I_{D'_{false}}$ (otherwise D'_{false} is true);
- $\Gamma_1\sigma'_1 \subseteq I_{D'_{false}}$:
  $\Gamma_1\sigma_1$ is satisfied by $I_{G\sigma_1}$ and remains so by lemma 6.42.
- no equation in $\Delta\sigma'$ is satisfied:
  $I_{D_{false}} \cap \Delta\sigma' = \emptyset$ because D_{false} is false in I_C (and lemma 6.44); note $I_{D_{false}} = I_{D'_{false}}$ (lemma 6.18)
- $\Delta_1\sigma'_1$ is not satisfied because of lemma 6.42 for the productive (hence non-redundant) clause $G\sigma_1$

So the only equation which can possibly be satisfied by the interpretation $I_{D'_{false}}$ is the reduced equation $t_1\sigma'[u \leftarrow u_E[s_2]\sigma'_E] \approx t_2\sigma'$ in the succedent of the conclusion. If it is satisfied, then also the E-equal equation $t_1\sigma[u \leftarrow u_E[s_2]\sigma_E] \approx t_2\sigma$ is satisfied (again all needed instances of E-equations are provided by lemma 6.44). By lemma 6.31 also $I_C \models t\sigma[u \leftarrow u_E[s_2]\sigma_E] \approx (t\sigma)'[p \leftarrow u_1[s_2]\sigma_1]$. Because I_C contains $(u_1[s_1\sigma])' \Rightarrow u_1[s_2\sigma]$, it would also satisfy the unreduced equation $(t_1\sigma)'[u \leftarrow (u_1[s_1\sigma])'] \approx t_2\sigma$ and with lemma 6.44 also the E-equal equation $t_1\sigma \approx t_2\sigma$ of the succedent of D_{false}. By lemma 6.44 we would have $I_C \supseteq I_{D'_{false}} \models t_1\sigma \approx t_2\sigma$. This would mean: D_{false} is true in I_C.
$C_E\sigma'_E$ and $D\sigma'$ are non-redundant ($C_E\sigma_E$ is either productive or an extended clause, so $C_E\sigma'$ is also non-redundant; $D\sigma'$ is not satisfied by I_C, because this holds for $D\sigma$ and we have lemma 6.44; if $D\sigma'$ is not satisfied in I_C, it is not satisfied in its own interpretation for the same reasons as $D\sigma$ is not satisfied in its interpretation, see beginning of this proof). As π' is redundant (because of saturation), the interpretation of the maximal premise of π' satisfies the conclusion $D_{\pi'}$. By the above considerations, this is a contradiction to the statement that the clause D_{false} is false in I_C. So the clause D_{false} cannot be false in I_C.

b2) There exists an equation $t_3\sigma \approx t_4\sigma$ in $\Delta\sigma$ which is E-equal to $t_1\sigma \approx t_2\sigma$:
We consider the equality factoring inference

$$\pi = \frac{\Gamma\sigma \rightarrow \Delta\sigma, t_1\sigma \approx t_2\sigma}{\Gamma\sigma, t_2\sigma \approx t_4\sigma \rightarrow \Delta\sigma}.$$

We lift π and instantiate it to an inference π' with premise D'_{false}. Because $I_{D'_{false}} \models t_2\sigma \approx t_4\sigma$ (lemma 6.44; note $t_2\sigma =_E t_4\sigma$), the inference π is only redundant if D'_{false} is redundant. Using lemma 6.46 this contradicts $I_C \not\models D'_{false}$.

c) Let now D_{false} be a clause $\Gamma\sigma, t_1\sigma \approx (t_1\sigma)' \rightarrow \Delta\sigma$ with maximal equation $t_1\sigma \approx (t_1\sigma)'$. In this case the lemma follows in a similar way using the *equality resolution* inference

$$\frac{\Gamma\sigma, t_1\sigma \approx (t_1\sigma)' \rightarrow \Delta\sigma}{\Gamma\sigma \rightarrow \Delta\sigma.}$$

The lifted (and then instantiated) inference is redundant (saturation of N). As the premise is non-redundant, the conclusion of the above inference is true in $I_{D'_{false}} = I_{D_{false}} \subseteq I_C$. So either the antecedent is not satisfied or an equation in the succedent is satisfied, both contradicts the fact that D_{false} is false.

d) It remains to consider $D_{false} = \Gamma\sigma, t_1\sigma \approx t_2\sigma \rightarrow \Delta\sigma$ with maximal equation $t_1\sigma \approx t_2\sigma$, $t_1\sigma >_E t_2\sigma$. In this case $I_C \not\models t_1\sigma \approx t_2\sigma$ (because D_{false} is not satisfied) and $t_1\sigma$ is reducible by $\Longrightarrow_{R_C}$ (also by $\Longrightarrow_{R_{D_{false}}}$) with a rule $(s_1\sigma_1)' \Rightarrow s_2\sigma_1$ produced by a clause $C_1\sigma_1$, $C_1 = \Gamma_1 \rightarrow \Delta_1, s_1 \approx s_2 \in N$ and $t_1\sigma/u = (s_1\sigma_1)'$. D_{false} is a reduced ground instance, so u is a non-variable occurrence of t_1. We consider the *strict superposition left* inference

$$\pi = \frac{C_1\sigma_1 \quad D_{false}}{\Gamma\sigma, \Gamma_1\sigma_1, t_1\sigma[u \leftarrow s_2\sigma_1] \approx t_2\sigma \rightarrow \Delta\sigma, \Delta_1\sigma_1}.$$

Similar to case b1) we get a contradiction: the premises of the inference are non-redundant, so the conclusion has to be satisfied in $I_{D'_{false}}$, but then D_{false} cannot be false in I_C.

□

Theorem 6.49 (Redundancy-Completeness of $\mathcal{M}_{Ext}$)
The inference system $\mathcal{M}_{Ext}$ is redundancy-complete for sets closed under E-extension.

Proof: Let $N := NE \cup Ext \cup SExt$ be an $\mathcal{M}_{Ext}$-saturated set of clauses closed under E-extension. Assume $N \cup E$ is inconsistent and N does not contain the empty clause. There are no inferences with premise TOP, so N is $\mathcal{M}_{Ext}$-saturated on NG_{TOP}. By lemma 6.44 we have $EG \subseteq I_{TOP}$. By the previous lemma 6.48 we know that I_{TOP} satsifies all ground clauses in NG. Hence every clause $C\sigma$ with C in $N \cup E$ and σ an arbitrary ground substitution is satisfied. So I_{TOP} is indeed a model for $N \cup E$.

This is a contradiction to the inconsistency. So $N \cup E$ cannot be inconsistent or N contains the empty clause. □

Corollary 6.50 (Inferences with Extended Clauses)
The system $\mathcal{M}_{Ext}$ remains redundancy-complete if we exclude equality factoring and equality resolution inferences with premise $C_E\sigma$, whenever C_E is an E-extended clause in Ext (and σ an arbitrary substitution). Moreover every superposition into a non-extended equation of an E-extended clauses is superfluous, e.g. superposition left inferences into the antecedent of an E-extended clause are not necessary. Also superpositions upon an occurrence strictly below the root of a term in an extended equation or into a non-maximal term of an extended clause are not necessary.

Proof: They are not needed in any previous lemmata used for the completeness proof. □

6.4 Extensions for the Theory AC

6.4.1 Extensions and Context Variables

With AC-theories we mean theories with some AC operators (but the following holds also for theories with some ACU-operators). In commutativity axioms (like $x + y \approx y + x$; and also in identity axioms like $x + 0 \approx x$) there are no terms having a non-variable proper subterm. So only associativity axioms (like $x+(y+z) \approx (x+y)+z$) lead to AC-extensions. These AC-extensions starting with a clause

$$\rightarrow a + b \approx c$$

are

$$\begin{aligned}
&\rightarrow (a+b)+x \approx c+x \\
&\rightarrow z+(a+b) \approx z+c \\
&\rightarrow ((a+b)+x)+x_2 \approx (c+x)+x_2 \\
&\ldots,
\end{aligned}$$

i.e. we need infinitely many clauses to get closed (under AC-extension) sets. But the second extension is AC-equal to the first one, and all other clauses are AC-equal to instances of the first one. So if any of the above left sides AC-matches a term, the first one will also match. Also the right sides of the extended equations are AC-equal, so if we replace a term matched by the left side of any of the above extensions by the (instantiated) right side $r\sigma$, we can do the same with the first rule and obtain a term E-equal to $r\sigma$. We will use these facts to show that we do not need more than one AC-extension and no extensions of extensions. We need extensions only for lemma 6.31. Below we present an AC-version of this lemma using at most one extension step.

Definition 6.51 (Context Variable)
Let C be a clause $\Gamma \rightarrow \Delta, \ell \approx r$, $+$ an AC-operator and x a variable. x is called a *context variable in* ℓ if

- $\ell = \ell_1 + \ell_2$, i.e. the root of ℓ is marked with $+$
- x does not occur in Γ or Δ
- x occurs once in ℓ
- x occurs directly below a $+$ operator:
  $x \in subterms_{AC}(+, \ell)$

x is called a *full context variable (in* ℓ*)* if

- x is a context variable in ℓ
- $r = r_1 + r_2$ or $r = x$, i.e. the root of r is $+$ or r is only the variable x
- x occurs once in r
- x occurs directly below a $+$ operator (on both sides):
  $x \in subterms_{AC}(+, r)$

Definition 6.52 (E-Superterm)
t/p is an *E-superterm of t/q in t* if and only if there exists

- an equation $e_1 \approx e_2 \in E$,
- a non-variable position $p_e \in O(e_1)$, $p_e \neq \varepsilon$
- a position $q_1 \in O(t)$ strictly above of q
- and a substitution σ with $(e_1/p_e)\sigma =_E t/q$ and $t/q_1 =_E e_1\sigma$
- and either $q_1 = q$ or t/q is an E-superterm of t/q_1 in t.

We say a subterm t/q to be a *maximal E-superterm* of t/p, if it is not properly contained in any E-superterm of t/p. We say t/q to be a maximal E-superterm, if there is no E-superterm of t/q in t.

There are subterms of t which have no E-superterm (e.g. if their root symbol does not occur in any E-equation). If a subterm t_1 has an E-superterm t_2 than there may be an E-equal term t_2' and some subterm t_1/p_1 which has no E-variant that is a subterm of t_2'.

Example 6.53 Let $+$ be an AC-operator, $t = (a+b)+c$, $t_1 = a+b$ and $t_2' = t' = (a+c)+b$. t_1 has an AC-superterm in t (which is t) but neither t_1 nor its only AC-variant $b + a$ is a subterm of t'.

Deciding whether a subterm has an E-superterm might be a difficult (and expensive) operation, depending on the theory E. For $E = AC$ it is a trivial operation: a subterm t/p has an AC-superterm in t, if the root operator of t/p is the AC-operator f and the node directly above t/p in t is marked with the same operator f.

Definition 6.54 (AC-Extended Clause)
Let $+$ be an AC-operator and C be a clause $\Gamma \rightarrow \Delta, \ell \approx r$ and $\ell = \ell_1 + \ell_2$ or $\ell = v$ (for a variable v). If there exists a ground substitution σ such that $\ell\sigma \approx r\sigma$ is the strictly maximal equation in $C\sigma$, $root(\ell\sigma) = +$ and $\ell\sigma >_{AC} r\sigma$, then the clause $C_{AC} = \Gamma \rightarrow \Delta, \ell + x \approx r + x$ is *the AC-extended clause* of C (for the equation $\ell \approx r$). We here assume x to be a new variable (i.e. a variable not occurring in other clauses). In that case we call the equation $\ell + x \approx r + x$ *the extended equation* for $\ell \approx r$ in C.

Note that we have at most one AC-extended clause for every equation in the succedent of a clause C (and for every AC-operator), in particular, we do not consider the symmetric (and AC-equal) clause $\Gamma \rightarrow \Delta, x + \ell \approx x + r$, which is an AC-extended clause in the sense of the general definition for E-extended clauses (definition 6.3).

Lemma 6.55 (Context Variables in AC-Extended Clauses)
If C is an AC-extended clause (i.e. there is another clause D such that C is the AC-extended clause for D), then the extended equation in C contains a full context variable.

Proof: Trivial (see definitions 6.51 and 6.54). □

Definition 6.56 (Sets Closed under AC-Extension)
A set N of clauses is *closed under AC-extension* if for every clause C which

- is unextended in N and
- has a ground instance $C\sigma$ containing an equation $\ell\sigma \approx r\sigma$ in the succedent that is strictly maximal in $C\sigma$ ($\ell\sigma >_{AC} r\sigma$) and the root of $\ell\sigma$ is marked with an AC-operator

the extended clause C_{AC} of C for the equation $\ell \approx r$ is also contained in N.

Different to definition 6.8, we do not need the notion of an *extended clause in* N, because we do not consider extensions of extensions.

Lemma 6.57 (Finite Closed Sets)
If N is a finite set of clauses, then there exists a finite set M of clauses such that $N \subseteq M$ and M is closed under AC-extension.

Proof: By the closedness definition above, no extensions are needed for extended clauses.

Each clause C in N has only a finite number n of equations in its succedent. For each such equation there exist at most two AC-extended clauses of C for each of the finitely many AC-operators (note that $E = AC$ is finite by definition 4.1). If no side of the equation in the succedent is simply a variable, then we get at most one AC-extended clause for such an equation.

Let Ext denote the set of all AC-extended clauses for clauses in N, then $M := N \cup Ext$ is closed under AC-extension: All extensions required for clauses in N are contained and extensions for clauses in Ext are not needed. □

Lemma 6.58 (Contexts and Productive Clauses)
Let $D\sigma = \Gamma \rightarrow \Delta, \ell\sigma \approx r\sigma$ be a productive ground clause with strictly maximal equation $\ell\sigma \approx r\sigma$ (and $\ell\sigma >_{AC} r\sigma$). Let $D_1\sigma$ be a productive ground clause, $D_1 \in Ext^*(D, \ell \approx r)$ and the maximal equation of $D_1\sigma$ is $u_1[\ell\sigma] \approx u_1[r\sigma]$. Let the root of $\ell\sigma$ be marked with f.

- If f is not an AC-operator, then $D_1 = D$ (hence $u_1 = []$).
- If f is an AC-operator $+$, then $u_1 = []$ or the root of u_1 and all nodes on the path from the root to the occurrence of $[]$ are also marked with $+$, so $u_1 =_{AC} [] + c_1 + \cdots + c_n$ $(n \geq 1)$.
- $u_1 = []$ or the root of u_1 is the same AC-operator as on the root of $\ell\sigma$.

Proof: If f is not an AC-operator, then all terms E-equal to $u_1[\ell\sigma]$ contain a subterm $(\ell\sigma)'$ which is reducible, because $D\sigma$ is productive. Hence $D_1\sigma$ is not productive unless it is $D\sigma$ itself.

Now consider the case $f = +$ and $u_1 \neq []$. If there exists a node marked with $g \neq +$ on the path from the root of u_1 to $[]$, then the clause D_1 cannot be a productive AC-extended clause of D. So all nodes on the above path are marked with $+$ and we can write u_1 as $[] + c_1 + \ldots + c_n$.

The third part follows from the previous parts. □

Lemma 6.59 (Properties for AC-Theories)
Let c be a context such that $c =_{AC} [] + c_1 + \ldots + c_n$ $(n \geq 1)$. We have $c[r] =_{AC} (c[r])'$ for every term r and the root of any term AC-equal to $c[r]$ is marked with $+$.

Let t be a ground term and t/p_{AC} be a maximal AC-superterm of t/p_{AC} in t. Every term t' contains a subterm $(t/p_{AC})'$, say at position $p'_{AC} \in O(t')$, and $t[p_{AC} \leftarrow r] =_{AC} t'[p'_{AC} \leftarrow r']$ (for all terms r and r' with $r =_{AC} r'$). Moreover $(t/p_{AC})'$ is a maximal AC-superterm of itself in t'.

Proof: The first part of the lemma is obvious for AC-theories (all equations are linear).

Now consider the single step proof $t = t_1 =_{AC} t_2 =_{AC} \ldots =_{AC} t_n = t'$. By the definition of AC-superterm each t_i contains a subterm s_i AC-equal to t/p_{AC}. Moreover s_i is a maximal AC-superterm of itself in t_i. Because of the linearity of the AC-equations we can rearrange and split the above chain into $t =_{AC} t_j =_{AC} t'$ and the AC-equation in the first part apply below p_{AC} and in the second part above this position and moreover the subterm $(t/p_{AC})'$ (in every term of the second part of the chain) is below a variable of the applied AC-equations. So (by linearity) we can apply the same AC-equations as in the second part (with a substitution in which $(t/p_{AC})'$ is replaced by r for a certain variable) to show $t[p_{AC} \leftarrow r] =_{AC} t'[p'_{AC} \leftarrow r]$. The required equality now follows from the previous one and $r =_{AC} r'$. □

Now we present the promised AC-version of lemma 6.31.

Lemma 6.60 (Use of AC-Extended Clauses)
Let $NE \cup Ext \cup SExt$ be a set of clauses closed under AC-extension (in the sense of 6.56). Let $D = \Gamma \rightarrow \Delta, \ell \approx r$ be a clause in $NE \cup Ext$ and σ a ground substitution such that $\ell\sigma \approx r\sigma$ is a strictly maximal equation in $D\sigma$, $\ell\sigma >_E r\sigma$ and $D\sigma$ is productive. Let t be a ground term. Assume I_C satisfies all ground instances of E-equations between terms smaller than t.

1. If t' is reducible at position p to a term $s := t'[p \leftarrow u_1[r\sigma]]$ by a rule $(u_1[\ell\sigma])' \Rightarrow u_1[r\sigma]$ produced by $D_1\sigma$ and $D_1 \in Ext^*(D, \ell \approx r)$, then there exists a clause $D_{AC} = \Gamma \rightarrow \Delta, \ell_{AC} \approx r_{AC}$ in $NE \cup Ext$ (sometimes D itself), a substitution σ_{AC} (sometimes σ itself) with $\Gamma\sigma = \Gamma\sigma_{AC}$, $\Delta\sigma = \Delta\sigma_{AC}$, $\ell_{AC}\sigma_{AC} =_{AC} t/p_{AC}$ and $t[p_{AC} \leftarrow r_{AC}\sigma_{AC}] =_{AC} t'[p \leftarrow u_1[r\sigma]] = s$ (hence $I_C \models t[p_{AC} \leftarrow r_{AC}\sigma_{AC}] \approx s$), where t/p_{AC} is the maximal AC-superterm of t/p in t. Either $D_{AC} = D$ or D_{AC} is the AC-extended clause of D for $\ell \approx r$.
2. In particular, if t is a ground instance of a side of an associativity axiom (other equation of AC do not have non-variable subterms) $e \approx ee \in E$, i.e. $t = e\sigma_e$ and p is a non-variable position of e such that t (resp. $e\sigma_e$) is reducible at position p to s by application of a rule generated by $D\sigma$, then there exists a clause $D_{AC} = \Gamma \rightarrow \Delta, \ell_{AC} \approx r_{AC}$ and a substitution σ_{AC} (as above) with $\ell_{AC}\sigma_{AC} =_{AC} t$ (i.e. ℓ_{AC} matches t at the root position) and $I_C \models r_{AC}\sigma_{AC} \approx s$.

In both cases we have either $D_{AC}\sigma_{AC} = D\sigma$ or $\rightarrow t \approx t >_c^{ext} D_{AC}\sigma_{AC}$ (and the maximal equation of $D_{AC}\sigma_{AC}$ gets the special complexity for extended clauses in part (ii), definition of $>_{eq}^{ext}$, 6.13).

Proof of part 1

If the root of t/p is not an AC-operator, then $p_{AC} = p$, $D_1 = D$ (hence $u_1 = []$ and we can use $\sigma_{AC} := \sigma$; see also lemma 6.58). $t[p_{AC} \leftarrow r_{AC}\sigma_{AC}] =_{AC} t'[p \leftarrow u_1[r\sigma]]$ follows from 6.59.

Similar to lemma 6.59 we need only to consider the case that all AC-equalities are applied below t/p_{AC}, i.e. $t'[p_{AC} \leftarrow []] = t[p_{AC} \leftarrow []]$. Assume the root operator of $(u_1[\ell\sigma])'$, hence the root of $u_1[\ell\sigma]$, t/p (lemma 6.59) and t/p_{AC} (definition of AC-superterm) is the AC-operator $+$. By lemma 6.58 the root operator of $\ell\sigma$ is also $+$. We have $t'/p_{AC} = c_{AC}[t'/p] =_{AC} c_{AC}[u_1[\ell\sigma]]$ (by our above consideration also $t/p_{AC} =_{AC} c_{AC}[u_1[\ell\sigma]]$) and

define $context := c_{AC}[u_1]$. By lemma 6.58 and definition of AC-superterm *context* is either the empty context or AC-equal to $c_1 + c_2 + \cdots + c_n + []$ $(n \geq 1)$. We distinguish some cases:

- Case A: $context = []$
  We have $p = p_{AC}$, $u_1 = []$ and the applied rule is $(\ell\sigma)' \Rightarrow r$. Simply take $\sigma_{AC} := \sigma$ and $D_{AC} := D$.
- Case B: $context \neq []$ and $D \in Ext$
  Then D contains a full context variable x in ℓ. Take $D_{AC} := D$, so $\ell_{AC} = \ell$ and $r_{AC} = r$. We have $\ell =_{AC} \ell_1 + x$ and $r =_{AC} r_1 + x$ (for appropriate ℓ_1 and r_1). Now define σ_{AC} by $x\sigma_{AC} := x\sigma + c_1 + \cdots + c_n$ and $y\sigma_{AC} := y\sigma$ for all $y \neq x$. We have:

  $$t/p_{AC} =_{AC} \ell\sigma + c_1 + \cdots + c_n =_{AC} \ell_1\sigma + x\sigma + c_1 + \cdots + c_n =_{AC} \ell_1\sigma + x\sigma_{AC} =_{AC} \ell_{AC}\sigma_{AC}$$

  Because of lemma 6.59 we get

  $$c_{AC}[u_1[r\sigma]] = t[p \leftarrow u_1[r\sigma]]/p_{AC} = s/p_{AC} =_{AC} r\sigma + c_1 + \cdots + c_n$$

  $$=_{AC} r_1\sigma + x\sigma + c_1 + \cdots + c_n =_{AC} r_1\sigma + x\sigma_{AC} =_{AC} r_{AC}\sigma_{AC}$$

  If we put this AC-equality into the context $t[p_{AC} \leftarrow []]$ we immediately get $t[p_{AC} \leftarrow r_{AC}\sigma_{AC}] =_{AC} s$.
  Obviously $\ell\sigma_{AC} >_{AC} \ell\sigma$ and $\ell\sigma_{AC}$ is the maximal term of the extended clause D under the substitution σ_{AC}. Therefore $\ell\sigma_{AC} \approx r\sigma_{AC}$ gets the special complexity (in definition 6.13, case (ii)) and we have $\rightarrow t \approx t >_c^{ext} D\sigma_{AC}$.
- Case C: $context \neq []$ and $D \in NE$
  Here we take the AC-extended clause of D for $\ell \approx r$, i.e. $D_{AC} = \Gamma \rightarrow \Delta, \ell + x \approx r + x$, so $\ell_{AC} = \ell + x$ and $r_{AC} = r + x$ and x is a context variable of D_{AC}. Now define σ_{AC} by $x\sigma_{AC} := c_1 + \cdots + c_n$ and $y\sigma_{AC} := y\sigma$ for all $y \neq x$. We have:

  $$t/p_{AC} =_{AC} \ell\sigma + c_1 + \cdots + c_n =_{AC} \ell\sigma + x\sigma_{AC} =_E \ell_{AC}\sigma_{AC}$$

  Because of lemma 6.59 we get

  $$c_{AC}[u_1[r\sigma]] = t[p \leftarrow u_1[r\sigma]]/p_{AC} = s/p_{AC} =_{AC} r\sigma + c_1 + \cdots + c_n$$

  $$=_{AC} r\sigma + x\sigma_{AC} =_{AC} r_{AC}\sigma_{AC}.$$

  If we put this AC-equality into the context $t[p_{AC} \leftarrow []]$ we immediately get $t[p_{AC} \leftarrow r_{AC}\sigma_{AC}] =_{AC} s$.
  Obviously $\ell_{AC}\sigma_{AC} >_{AC} \ell\sigma$ and $\ell_{AC}\sigma_{AC}$ is the maximal term of the extended clause D_{AC} under the substitution σ_{AC}. Therefore $\ell_{AC}\sigma_{AC} \approx r_{AC}\sigma_{AC}$ gets the special complexity (in definition 6.13, case (ii)) and we have $\rightarrow t \approx t >_c^{ext} D_{AC}\sigma_{AC}$.

Proof of part 2

Here we have $p_{AC} = \varepsilon$, so part 2 is simply an application of part 1. □

Remark to the previous lemma: Case B proves that extensions for extended clauses are not necessary. It uses only the fact that D contains a context variable x. So replacing $D \in Ext$ in case B by *D contains a context variable x in ℓ* and also replacing $D \in NE$ in case C

by D *does not contain a context variable in* ℓ, the lemma proves that for superpositions we do not need extensions for clauses containing a context variable (in the considered equation). But if the clause D contains a context variable and belongs to NE, we do not get $\rightarrow t \approx t >_c^{ext} D\sigma_{AC}$, because clauses in NEG do not get a reduced complexity in the definition of $>_{eq}^{ext}$ (6.13). The reduced complexity is needed in lemmata 6.44 and 6.48 to show the used extended clause to be smaller than the considered clause C or $\rightarrow t \approx t$. We will reuse this proof with the above remark in lemma 6.67.

This lemma looks quite different to lemma 6.31 (in particular the proofs are different). To prove refutational completeness of $\mathcal{M}_{Ext}$ we used only parts 2 and 3 of lemma 6.31. Part 1 is used for induction to prove the lemma itself (we in general need extensions of extensions of productive clauses, but already the first extension might not produce rules). Here (in lemma 6.60) we can skip part 1 (of lemma 6.31) because *we need at most one* AC*-extension step.*

We also prove a stronger property: If a term is reducible at position p we can superpose on the maximal AC-superterm of t/p and not simply on t/p. So in the AC-case we can also superpose on t/p_{AC} and for general E on maximal E-superterms. But we doubt, whether it is interesting for general E. We are sure it is interesting for AC: It is *sufficient to superpose upon maximal* AC*-superterms*, so all possible AC-variants are treated the same way and in an implementation we can abstract from certain AC-variants, e.g. use *flattened terms.*

But the lemmata (6.31 and 6.60) share the main property: If a term t' is reducible, then we can superpose a clause on any AC-variant t of t' (and the term obtained by the superposition replacement is in our interpretation equal to the reduced term).

Using the lemma 6.60 (parts 1 and 2) instead of 6.31 (parts 2 and 3) we can prove the refutational completeness for AC-theories as done for the general case.

Lemma 6.61 (Superpositions upon Maximal AC-Superterms)
If we restrict the superposition left and superposition right inferences such that we superpose upon maximal AC-superterms only, the inference system remains redundancy-complete.

Proof: If we can superpose (the greater side of the maximal equation of a productive clause) upon a subterm t/p (occurring in another clause; note that other inferences are not considered in previous proofs), then by lemma 6.60 we always find a clause such that we can also superpose upon the maximal AC-superterm of t/p in t. Replacing an inference π with a superposition upon a term having a proper AC-superterm by an inference π_{max} with superposition upon the maximal AC-superterm, we get an AC-equal conclusion. Moreover for any ground instance, we know that the AC-equalities to show the conclusions of π and π_{max} to be AC-equal contain only terms smaller than the maximal term of the inference. Whenever we need a superposition inference in the completeness proof for $\mathcal{M}_{Ext}$, we know that the needed AC-equalities are satisfied. So the completeness proof remains valid, if we exclude the above inferences. □

6.4.2 Further Reducing the Number of AC-Extended Clauses

Unnecessary extended clauses may have a strong influence on the efficiency of completion or theorem proving methods. Therefore we now want to give a weaker (than 6.56) closedness definition (we call it AC-closed sets), sometimes resulting in much less AC-extended clauses (see example 6.65).

Definition 6.62 (AC-Closed Sets)
A set N of clauses is *AC-closed* if for every clause $C = \Gamma \to \Delta, \ell \approx r$ which

- is unextended and
- has a ground instance $C\sigma$ such that the equation $\ell\sigma \approx r\sigma$ is strictly maximal in $C\sigma$, $\ell\sigma >_{AC} r\sigma$ and $root(\ell\sigma) = +$ for an AC-operator $+$

either the AC-extended clause of C for $\ell \approx r$ is contained in N or C contains a context variable x in ℓ and there are finitely many clauses $D_i \in N \cup AC$ such that for all ground terms c there are ground substitutions ρ_i with the following properties:

- $D_i\rho_i$ contains only terms smaller than $\ell\sigma + c$ and
- $\{D_i\rho_i\} \models r\sigma + c \approx r\sigma_1$,

where σ_1 denotes the substitution with $y\sigma_1 = y\sigma$ for all $y \neq x$ and $x\sigma_1 = x\sigma + c$.

Let us in the latter case call C a *self extending clause (in N)*.

Lemma 6.63 Sets closed under AC-extension (by definition 6.56) are always AC-closed (hence there are always finite AC-closed sets for finite sets of clauses).

Proof: Always use the "either" branch in the above definition. □

But in the previous definition, self extending clauses (a subset of clauses containing a context variable) do no need extensions. A clause with a full context variable (e.g. every AC-extended clause) is self extending:

Example 6.64 Let x denote a full context variable in $C = \Gamma \to \Delta, \ell + x \approx r + x$. Then $\{C\}$ is AC-closed (and also $\{C\} \cup N$ for an arbitrary AC-closed set N). Let σ, c and σ_1 be defined as in 6.62. We have to show that $(r + x)\sigma + c \approx (r + x)\sigma_1$ follows from other clause. We need only the AC-equality $(r\sigma + x\sigma) + c \approx r\sigma + (x\sigma + c)$, which obviously contains only terms smaller than $(\ell + x)\sigma + c$.

But there are also clauses C with context variables (not full context variables) such that C does not need extensions.

Example 6.65 (Natural Numbers)
We consider a specification of natural numbers with 0 and the successor function s as constructors and AC-operator symbols $+$ (addition) and $*$ (multiplication).

$$
\begin{aligned}
Nat = \ \{ \quad & \to x + 0 \approx x \\
& \to s(y) + x \approx s(y + x) \\
& \to x * 0 \approx 0 \\
& \to s(y) * x \approx y * x + x \\
& \to x * (y + z) \approx x * y + x * z) \ \}
\end{aligned}
$$

The set is AC-closed (if we assume an ordering $>_{AC}$ such that each of the above equations is orientable from left to right): W.o.l.o.g. we consider a substitution σ replacing the variables in the clauses by arbitrary ground terms denoted by $n, n_1, n_2, \ldots$ and the context variable x (from definition 6.62) by m. We will show that for every clauses C of *Nat* we do not need any extension, because C is self extending in *Nat*.

- $\rightarrow x + 0 \approx x$: This clause already contains a full context variable (so does not need an extension, cf. previous example).
- $\rightarrow s(y) + x \approx s(y + x)$: The clause contains a context variable x. It remains to show that $s(n_1 + (n_2 + m)) \approx s(n_1 + n_2) + m$ follows from clauses containing only terms smaller than $(s(n_1) + n_2) + m$. It follows from the clause itself with the substitution $\{y \leftarrow n_1 + n_2, x \leftarrow m\}$ and the AC-equality $s((n_1 + n_2) + m) =_{AC} s(n_1 + (n_2 + m))$.
- $\rightarrow x * 0 \approx 0$: the clause contains a context variable x and $0 * m \approx 0$ follows from the clause itself by substituting x with m.
- $\rightarrow s(y) * x \approx y * x + x$: The context variable is named x. Now consider $(n_1 * n_2 + n_2) * m \approx n_1 * (n_2 * m) + n_2 * m$: it follows with some AC-equalities and the instance $\rightarrow (n_1 * n_2 + n_2) * m \approx (n_1 * n_2) * m + n_2 * m$ of the last clause (the distributivity law).
- $\rightarrow x * (y + z) \approx x * y + x * z$: Again the clause contains a context variable x. We conclude $(n_1 * n_2 + n_1 * n_3) * m \approx n_1 * m * n_2 + n_1 * m * n_3$ by another application of the distributivity axiom to the left side of this equation (and some small AC-equalities).

With definition 6.56 we need AC-extended clauses for all clauses. This would result in many unnecessary superposition inferences with these useless extended clauses.

Example 6.66 As a second example we present a structure with an associative and commutative multiplication operation, a zero 0 (clause 2), a neutral element 1 (clause 3), which is different from 0 (clause 1, hence there is no trivial model for this specification), and an inverse operation for multiplication (clauses 4 and 5). Moreover the structure is zero divisior free (clause 7). Note that it contains three non-Horn clauses (clauses 4, 5 and 7).

$$
\begin{array}{ll}
(1) & 1 \approx 0 \rightarrow \\
(2) & \rightarrow x \cdot 0 \approx 0 \\
(3) & \rightarrow x \cdot 1 \approx x \\
(4) & \rightarrow x \cdot inv(x) \approx 1, x \approx 0 \\
(5) & \rightarrow (x \cdot inv(x)) \cdot y \approx 1 \cdot y, x \approx 0 \\
(6) & \rightarrow inv(1) \approx 1 \\
(7) & w \cdot v \approx 0 \rightarrow w \approx 0, v \approx 0
\end{array}
$$

The set of clauses is AC-closed. Clause (5) is the AC-extended clause for (4). Any other clause does not need an extension: clauses (1) and (6) do not contain a multiplication operator; clause (7) has its maximal equation in the antecedent; clauses (2) and (3) are self-extending, clause (5) too, because it is itself an extended clause.

Inferences from the above clauses are redundant. As an example we consider a superposition left inference between clauses (5) and (7):

$$
\frac{\rightarrow (x \cdot inv(x)) \cdot y \approx 1 \cdot y, x \approx 0 \quad inv(x) \cdot (x \cdot y) \approx 0 \rightarrow inv(x) \approx 0, x \cdot y \approx 0}{1 \cdot y \approx 0 \rightarrow inv(x) \approx 0, x \cdot y \approx 0, x \approx 0}
$$

We have to substitute clause (7) by $\sigma = \{w \leftarrow inv(x),\ v \leftarrow x \cdot y\}$ to obtain the right premise of the above inference. Because clause (7) is symmetric in w and v, the redundancy of a

similar inference with $\sigma_{sym} = \{v \leftarrow inv(x),\ w \leftarrow x \cdot y\}$ follows from the redundancy of the above inference.

Now we prove the redundancy of the above inference. Consider a ground instance of the above inference with substitution σ_1. If the equation $1 \cdot y\sigma_1 \approx 0$ in the antecedent of the conclusion is not satisfied, the ground instance is true. Otherwise we have $1 \cdot y\sigma_1 \approx 0$, and by clause (3) also $y\sigma_1 \approx 0$. With clause (2) we conclude $x\sigma_1 \cdot y\sigma_1 \approx 0$, hence an equation in the succedent of the conclusion is satisfied and the conclusion is true in this case, too.

The self extending clauses do not really fit in our previously presented framework, which is based on the separation of NE and Ext. Here some instances of self extending clauses have to be treated as NEG-clauses, others as $ExtG$-clauses.

We will only sketch, which modifications are needed to work with self extending clauses. We will denote a self extending clause by S and assume $S = \Gamma \rightarrow \Delta, \ell + x \approx r$ (for a context variable x in $\ell + x$). For the interpretation (cf. definition 6.17), instances $S\sigma$ are in general treated as NEG-clauses, except x is substituted by $t_1 + c$, then they are treated as $ExtG$-clauses, e.g. produce rules if $S\tau$ is productive (where τ is similar to σ, except that x is substituted by t_1 only). So we may regard $S\sigma$ as an extended clause for $S\tau$. Also the complexity of such a σ-instance of S has to be similar to the complexity of $ExtG$-clauses (cf. definition 6.13, case (ii)).

Then we can formulate a lemma similar to 6.60, but now using definition 6.62. We will emphasize the differences to lemma 6.60 using *italics*.

Lemma 6.67 (Use of AC-Closed Sets)
Let N be an *AC-closed* set of clauses (see definition 6.62). Let $D = \Gamma \rightarrow \Delta, \ell \approx r$ be a clause in N and σ a ground substitution such that $\ell\sigma \approx r\sigma$ is a strictly maximal equation in $D\sigma$, $\ell\sigma >_E r\sigma$ and $D\sigma$ is productive. Let t be a ground term. Assume I_C satisfies all ground instances of E-equations between terms smaller than t *and all ground instances of clauses in N containing only terms smaller than t.*

1. If t' is reducible at position p to a term $s := t'[p \leftarrow u_1[r\sigma]]$ by a rule $(u_1[\ell\sigma])' \Rightarrow u_1[r\sigma]$ produced by $D_1\sigma$ and $D_1 \in Ext^*(D, \ell \approx r)$, then there exists a clause $D_{AC} = \Gamma \rightarrow \Delta, \ell_{AC} \approx r_{AC}$ in N (sometimes D itself), a substitution σ_{AC} (sometimes σ itself) with $\Gamma\sigma = \Gamma\sigma_{AC}$, $\Delta\sigma = \Delta\sigma_{AC}$, $\ell_{AC}\sigma_{AC} =_{AC} t/p_{AC}$ and $I_C \models t[p_{AC} \leftarrow r_{AC}\sigma_{AC}] \approx s$, where t/p_{AC} is the maximal AC-superterm of t/p in t.
2. In particular, if t is a ground instance of a side of an associativity axiom (other equation of AC do not have non-variable subterms) $e \approx ee \in E$, i.e. $t = e\sigma_e$ and p is a non-variable position of e such that t (resp. $e\sigma_e$) is reducible at position p to s by application of a rule produced by $D\sigma$, then there exists a clause $D_{AC} = \Gamma \rightarrow \Delta, \ell_{AC} \approx r_{AC}$ and a substitution σ_{AC} (as above) with $\ell_{AC}\sigma_{AC} =_{AC} t$ (i.e. ℓ_{AC} matches t at the root position) and $I_C \models r_{AC}\sigma_{AC} \approx s$.

In both cases we have either $D_{AC}\sigma_{AC} = D\sigma$ or $\rightarrow t \approx t >_c^{ext} D_{AC}\sigma_{AC}$ (and the maximal equation of $D_{AC}\sigma_{AC}$ gets the special complexity for extended clauses in part (ii), definition of $>_{eq}^{ext}$, 6.13).

Proof: The proof is very similar to the one of lemma 6.60. Whenever lemma 6.60 needs an AC-extended clause (i.e., when the clause does not contain an appropriate context variable, see also the remarks following lemma 6.60), then either there exists an extended

clause C_{AC} by definition 6.62, or another instance of the already considered clause D, which has properties similar to AC-extended clauses. Note that we do not require $t[p_{AC} \leftarrow r_{AC}\sigma_{AC}] =_{AC} t[p \leftarrow c[r\sigma]] = s$ but only $I_C \models t[p_{AC} \leftarrow r_{AC}\sigma_{AC}] \approx s$ (because for self extending clauses we may need further clauses of N to show this equality, it is in general not simply an AC-equality) and that I_C has to satisfy more clauses than in lemma 6.60.

Note that we obtain the match of ℓ and t/p_{AC} in case B (proof of lemma 6.60), if ℓ contains a context variable. We do not need a full context variable for the match. But for a full context variable we get $t[p_{AC} \leftarrow r_{AC}\sigma_{AC}] =_{AC} s$ and with a context variable which is not a full context variable, we need the clauses D_i to show that a similar equation holds in I_C. □

The previous completeness proofs already contain *I_C satisfies all clauses smaller than C* as an induction hypothesis (for some lemmata; we can add it as a hypothesis to the whole proof). So we can use this lemma the same way we used lemma 6.31 to show the redundancy-completeness of $\mathcal{M}_{Ext}$ for AC-closed sets.

Continuing the above consideration (with the example S, $S\sigma$ and $S\tau$) about modifications in the completeness proof, we have to remark that a lemma similar to 6.47 (there we show that $D_E\sigma$ is satisfied if the corresponding non-extended clause $D\sigma$ is satisfied) does not hold for $S\sigma$ and $S\tau$. $S\tau \not\models S\sigma$, but using the AC-equality $(\ell\sigma + t_1) + c \approx \ell\sigma + (t_1 + c)$ we have $\{(\ell\sigma + t_1) + c \approx \ell\sigma + (t_1 + c), S\tau\} \models S\sigma$. So again the clause $S\sigma$ is satisfied, if all NEG-clauses and all AC-equalities are satisfied. But the above AC-equality is (in general) not satisfied in $I_{S\sigma}$ nor the interpretation of greater $ExtG$-clauses with the same maximal term $\ell\sigma + t_1 + c$ (or a term E-equal to it), so we have to modify lemma 6.48: if C is an NEG-clause, it satisfies all smaller clauses (no modification needed), but if C is an $ExtG$-clause, then I_C satisfies only every clause in NG which contains only terms smaller than the maximal term of C. This lemma is sufficient to prove that all clauses are satisfied by I_{TOP}. So we can use I_{TOP} to show the redundancy-completeness of $\mathcal{M}_{Ext}$ for AC-closed sets (cf. theorem 6.49).

Note that we have to modify lemma 6.48. So whenever we use this lemma in the following (e.g. in connection with compositeness, cf. section 7), we need some modification to use the following for AC-closed sets (e.g. an inference with maximal premise $S\sigma$ is composite, if its conclusion follows from clauses containing only terms smaller than the maximal term of $S\sigma$, so $ExtG$-clauses with the same maximal term cannot be used).

6.4.3 Implementing AC-Rewriting

Working with equational specifications or sets of Horn clauses (which are both contained in our framework as special cases) we often use rewriting or conditional rewriting to show clauses (equations) to be redundant (i.e. for simplification and elimination purposes). For $E = AC$ rewriting with AC-matching is often used. We will here present a slightly weaker rewrite relation having the advantages that it is easy to implement and works also with *flattened terms*. We will give the definition for the unconditional case. It can be lifted to conditional rewriting for definite Horn clauses the usual way.

Definition 6.68 (Rewriting with Maximal AC-Matching)
Let R be a rewrite system. A term t *rewrites with maximal AC-matching to s* (we write $t \Longrightarrow_{R^{AC}} s$) if and only if there exists a position $p \in O(t)$ such that t/p is the maximal AC-superterm of t/p in t, a rule $\ell \Rightarrow r$ and a substitution σ with $t/p =_{AC} \ell\sigma$ and $s = t[p \leftarrow r\sigma]$.

Note that rules are applied with AC-matching, but only at maximal AC-superterms. We will study the properties of the interesting rewrite relation $\Longrightarrow_{R^{AC}}$ in greater detail.

Definition 6.69 Let R be a set of rewrite rules, represented as a set of clauses N, where N contains a clause $\rightarrow \ell \approx r$ for each rule $\ell \Rightarrow r$ of R. Let N_1 denote the set of all AC-extended clauses (in the sense of AC-specific definition 6.54) for clauses in N. R_{ext} denotes the rewrite system consisting of a rule $\ell \Rightarrow r$ for each clause $\rightarrow \ell \approx r$ in $N \cup N_1$ (note that $N \cup N_1$ is closed under AC-extension).

Lemma 6.70 Let R be a set of rules and t and s be ground terms. We have:

$$\begin{array}{rl} t \Longrightarrow_{R/AC} s & \text{if and only if} \\ t \Longrightarrow_{R_{ext} \cdot AC} s' & \text{if and only if} \\ t \Longrightarrow_{R_{ext}^{AC}} s'' & \end{array}$$

Proof: By definition of AC-extension we have $\Longleftrightarrow_R = \Longleftrightarrow_{R_{ext}}$. Regarding a rewrite step

$$t \Longrightarrow_{R_{ext} \cdot AC} s$$

as an abbreviation of

$$t =_{AC} t' \Longrightarrow_{R_{ext}} s,$$

we can translate it to

$$t =_{AC} t' \Longrightarrow_R s.$$

This implies $\Longrightarrow_{R_{ext} \cdot AC} \subseteq \Longrightarrow_{R/AC}$. The inclusion $\Longrightarrow_{R_{ext}^{AC}} \subseteq \Longrightarrow_{R_{ext} \cdot AC}$ is obvious. Both inclusions yield a proof for the if-direction of the lemma.

Now consider

$$t \Longrightarrow_{R/AC} s$$

as an abbreviation for

$$t =_{AC} t' \Longrightarrow_{R/AC} s' =_{AC} s,$$

where the rule $\ell \Rightarrow r$ is applied at position p' of t' under the substitution σ. Let t'/p'_{AC} denote the maximal AC-superterm of t'/p' in t'. By lemma 6.59 there exists a position p_{AC} of t such that $t/p_{AC} =_{AC} t'/p'_{AC}$ and $t[p_{AC} \leftarrow u] =_{AC} t'[p'_{AC} \leftarrow u']$ (for all terms u and u' with $u =_{AC} u'$). Moreover $(t/p_{AC})'$ is a maximal AC-superterm of itself in t. Note that $t'/p'_{AC} =_{AC} c[\ell\sigma]$ for an appropriate context $c = c_1 + \ldots + c_n + []$ and $s =_{AC} t'[p'_{AC} \leftarrow c[r\sigma]]$. Similar to lemma 6.60, there exists a rule $\ell_1 \Rightarrow r_1$ in R_{ext} which matches t/p_{AC} at the root and reduces it to a term AC-equal to $c[r\sigma]$. As only the notation differs from lemma 6.60, we will repeat only a sketch of the case analysis:

- For $p' = p'_{AC}$ we use the same rule $\ell \Rightarrow r$ under the same substitution σ.
- If the rule $\ell \Rightarrow r$ contains a context variable x, we extend the substitution such that x also matches the context part c (not the [] of course).
- Otherwise we use the extended rule.

Because t/p_{AC} is a maximal AC-superterm, this rule reduces t to a term s'' using the rewrite relation with maximal AC-matching. So we have shown that the first line of the lemma implies the third one. But with $\Longrightarrow_{R_{ext}^{AC}} \subseteq \Longrightarrow_{R_{ext} \cdot AC}$ this implies also the second statement. □

Using R_{ext} and AC-matching we get the same reduction power as using rewriting modulo AC. Even more, we get a coherent rewrite relation:

Lemma 6.71 Let R be an arbitrary set of rules. The rewrite relations $\Longrightarrow_{R/AC}$, $\Longrightarrow_{R_{ext} \cdot AC}$ and $\Longrightarrow_{R_{ext}^{AC}}$ are coherent modulo AC. (But they are not confluent in general.) Moreover

$$\begin{array}{rll} t \Longrightarrow_{R_{ext} \cdot AC} s & \text{implies} & t' \Longrightarrow_{R_{ext} \cdot AC} s'' \\ t \Longrightarrow_{R_{ext}^{AC}} s & \text{implies} & t' \Longrightarrow_{R_{ext}^{AC}} s'' \end{array}$$

(More exactly: the rewrite step implies that for all t' with $t' =_{AC} t$ there exists an s' with $s' =_{AC} s$ and t' can be reduced to s'.)

Proof: Rewriting modulo AC is coherent (modulo AC) by definition. We now prove only the "moreover" part of the lemma, which implies coherence, so consider

$$t' =_{AC} t \Longrightarrow_{R_{ext} \cdot AC} s.$$

By the previous lemma we have

$$t' \Longrightarrow_{R/AC} s'$$

and applying the lemma again we get

$$t' \Longrightarrow_{R_{ext} \cdot AC} s''.$$

That is the coherence modulo AC of $\Longrightarrow_{R_{ext} \cdot AC}$. The same proofs applies to $\Longrightarrow_{R_{ext}^{AC}}$ instead of $\Longrightarrow_{R_{ext} \cdot AC}$. □

This lemma ensures that we can apply lemma 6.70 to arbitrary chains of rewriting, not only to one-step rewriting:

Lemma 6.72 Let R be a set of rules and t and s be ground terms. We have:

$$\begin{array}{rl} t \Longrightarrow_{R/AC}^{*} s & \text{if and only if} \\ t \Longrightarrow_{R_{ext} \cdot AC}^{*} s' & \text{if and only if} \\ t \Longrightarrow_{R_{ext}^{AC}}^{*} s'' & \end{array}$$

Proof: We will only consider a chain of two rewrite steps and simulate a derivation with rewriting modulo AC by rewriting with AC-matching. Then the remaining parts of the proof become obvious. Consider

$$t \Longrightarrow_{R/AC} s \Longrightarrow_{R/AC} r.$$

By lemma 6.70 we have two rewrite steps

$$t \Longrightarrow_{R_{ext} \cdot AC} s' \quad \text{and} \quad s \Longrightarrow_{R_{ext} \cdot AC} r'.$$

By the previous lemma applied to the second rewrite step we yield

$$s' \Longrightarrow_{R_{ext} \cdot AC} r''.$$

So we can combine rewrite steps to

$$t \Longrightarrow_{R_{ext} \cdot AC} s' \Longrightarrow_{R_{ext} \cdot AC} r''.$$

□

So we can simulate the relation $\Longrightarrow_{R/AC}$ by rewriting with maximal AC-matching in the extended set of rules. Often in AC-completion procedures based on $\mathcal{M}_{Ext}$, the set of clauses is kept closed under AC-extension, so the rewrite system related to the set of clauses (or a subset of it) is often already closed under AC-extension (i.e. $R = R_{ext}$) and we can immediately use rewriting with maximal AC-matching without constructing an enhanced rewrite system.

Implementing rewriting with a set R of rules in a completion procedure, we could construct a global matching procedure considering all of the rules in R at the same time: we can use information discovered when trying to match a term with a rule r, to match the same term with other rules. But as the set of rules often changes during completion, we do not do so: we forget about information involved in previous mismatches (because it is too expensive to construct a global matching procedure for a set of rules which often changes, so we have to construct such a procedure every time the set of rule changes). The pair of a rule and its extension is often considered as a unity, e.g. both rules are constructed together and are eliminated together. If we do so, we should also implement the matching with both of these rules as a single action, because the rules are so closely related that a mismatch with one of them yields valuable information to the match (or mismatch) with the other rule.

Regarding that sometimes a rule which has an extension does not need the extension (e.g. $s(x) + y \Rightarrow s(x + y)$, cf. example 6.65), a completion algorithm might work with a subset of R_{ext}.

Definition 6.73 By R_1 we denote any subset of R_{ext} such that for each rule $\ell \Rightarrow r$ in R_{ext} the terms ℓ and r are joinable modulo AC with the rewrite relation $\Longrightarrow_{R_1^{AC}}$.

We may construct such a set R_1 by first adding extended rules for each rule in R (of course only if there is any such extended rule for the considered rule). Then we can eliminated some rules $\ell \Rightarrow r$, if ℓ and r are reducible to AC-equal terms using rewriting with maximal AC-matching and applying only rules different from $\ell \Rightarrow r$.

Assume $\ell + x \approx r + x$ is the extension of $\ell \Rightarrow r$ and ℓ already contains a context variable. If we can reduce $\ell + x$ by a σ-instance of $\ell \Rightarrow r$ to $r\sigma$ and then join $r\sigma$ and $r + x$, then the clause isomorph to the rule $\ell \Rightarrow r$ is self extending. This method yields a sufficient criterion for a clause to be self extending. The above definition of R_1 covers this method and we may encounter such a set of rules R_1 in an AC-completion process based on AC-closed sets.

Lemma 6.74 Let R be a set of rules and t and s be ground terms. We have:

$$t \Longrightarrow_{R/AC} s \qquad \text{implies} \qquad t \Downarrow_{R_1^{AC}} s'$$

Proof: The proof of lemma 6.70 provides a rule $\ell_1 \Rightarrow r_1 \in R_{ext}$ able to reduce t to s'. By the above definition of R_1 we have $\ell_1 \Downarrow_{R_1^{AC}} r_1$, which implies $t \Downarrow_{R_1^{AC}} s'$. □

To get a coherent rewrite relation, we have to further restrict the elimination of rules from R_{ext}. We admit at most one reduction step at the right side of a rule:

Lemma 6.75 If the rewrite relation $\Longrightarrow_{R/AC}$ is terminating and for every rule $\ell \Rightarrow r \in R_{ext} \setminus R_1$ we have $\ell \Longrightarrow^+_{R_1^{AC}} v =_{AC} v' \Longleftarrow_{R_1^{AC}} r$ (note that the last rewriting is single step rewriting) or $\ell \Longrightarrow^+_{R_1^{AC}} r' =_{AC} r$ (for an example of such a set R_1 see 6.65 and consider each clause in *Nat* as a rewrite rule), then the rewrite relation $\Longrightarrow_{R_1^{AC}}$ is coherent modulo AC.

Proof: Rewriting modulo AC is coherent (modulo AC) by definition. Now consider

$$t' =_{AC} t \Longrightarrow_{R_1^{AC}} s.$$

Then we have

$$t' \Longrightarrow_{R/AC} s'$$

(because $\Longrightarrow_{R_1} \subseteq \Longrightarrow_{R_{ext}} = \Longrightarrow_R$) and applying the previous lemma (regarding that there exists a rule $\ell_1 \Rightarrow r_1 \in R_{ext}$ which is able to reduce t' to s'' and this rule is either contained in R_1 or eliminated the restricted way allowed for this lemma) we get either

$$t' \Longrightarrow^+_{R_1^{AC}} s''$$

or

$$t' \Longrightarrow^+_{R_1^{AC}} v =_{AC} v' \Longleftarrow_{R_1^{AC}} s''.$$

The first case is the coherence modulo AC of $\Longrightarrow_{R_1^{AC}}$. For the latter case we have to eliminate a coherence peak

$$s =_{AC} s'' \Longrightarrow_{R_1^{AC}} v'$$

using induction on the size of coherence peaks occurring in the equational proof (using $\Longrightarrow_{R/AC}$ as on ordering). Due to our restriction on eliminating R_{ext} rules, we get only coherence peaks, e.g. solving the above coherence peak by reducing s to w and v' to w' we get a new proof of the equality of t' and s containing a coherence peak

$$v =_{AC} v' \Longrightarrow_{R_1^{AC}} w,$$

but never a confluence peak (we will get confluence peaks for arbitrary R_1, i.e. admitting more than one reduction step on the right side of a rule). Each of these coherence peaks can be eliminated (we can prove this using induction). So in all cases we get a coherent relation. □

But to simulate chains of $\Longrightarrow_{R/AC}$ we either need the confluence modulo AC of $\Longrightarrow_{R_1^{AC}}$ (see next lemma) or we have to require that for rules $\ell \Rightarrow r$ eliminated from R_{ext} we have $\ell \Longrightarrow^+_{R_1^{AC}} r'$ (i.e. without reducing the right side of the rule). The latter case seems too unnatural to us, so we do not further consider it here.

Lemma 6.76 If $\Longrightarrow_{R_1^{AC}}$ is confluent modulo AC and $\Longrightarrow_{R/AC}$ is terminating, then we have:

$$t \Longrightarrow^*_{R/AC} s \qquad \text{implies} \qquad t \Downarrow^*_{R_{ext}^{AC}} s$$

Proof: Similar to the previous lemma we can show the coherence modulo AC of $\Longrightarrow_{R_1^{AC}}$ using its confluence modulo AC. Then by [Jouannaud/Kirchner 86] (their theorem 5) the relation is Church-Rosser modulo AC, so any equality in $\equiv_{R \cup AC}$ can be decided by rewriting (note that the termination of $\Longrightarrow_{R/AC}$ implies the termination of $\Longrightarrow_{R_1/AC}$). This holds particularly for t and s. □

Without confluence of $\Longrightarrow_{R_1^{AC}}$, we could transform a rewrite chain of $\Longrightarrow_{R/AC}$ only into a (in general) non-rewrite proof using $\Longrightarrow_{R_1^{AC}}$.

7 Redundancy and Completion

We consider refutational theorem proving. To prove that a theorem T holds in a specification S (here this means a set S of clauses), we add the negation of T (we assume $\neg T$ to be a set of clauses too) to S resulting in a specification S_0. Then a theorem prover takes as input this specification S_0 and produces a (possible infinite) sequence $S_1, S_2, \ldots$ of specifications until it finds a specification S_n for which it knows whether S_n is consistent (then T is not a theorem in S) or inconsistent (and T is a valid theorem). For $\mathcal{M}_E$-saturated (and finite) sets we can decide consistency by theorem 5.32 (and for $\mathcal{M}_{Ext}$-saturated sets by theorem 6.49). So a main task for the prover is the construction of such a saturated set starting from S_0 (we call this part *completion procedure*). A minimal requirement for the correctness of a transformation from S_{i-1} to S_i is the preserving of consistency and inconsistency. But often we want to use (parts of) the prover (its completion procedure) also to construct $\mathcal{M}_E$-saturated (resp. $\mathcal{M}_{Ext}$-saturated) sets for a consistent set S_0 to get a saturated set S_n which allows more efficiently to prove other theorems involving S_0 (example: the Knuth-Bendix completion of a set S_0 of unconditional equation; if we find a saturated set of equations (used as rules) S_n, then we can decide validity in S_0 by rewriting with S_n; if we allow transformations which preserve consistency but changes the class of models satisfying our specifications, consistent sets S_n produced with the theorem prover will be useless for further theorem proving). So we here require more: each transformation preserves the class of models (so S_{i-1} and S_i have the same models, if any). We distinguish *addition* and *deletion* transformations:

- Addition: $S_i = S_{i-1} \cup A$, where A is a set of logical consequences from S_{i-1}
- Deletion: $S_i = S_{i-1} \setminus D$, where D is a set of logical consequences of $S_{i-1} \setminus D$

Here a simplification transformation replacing a clause C by a simpler clause C_s is regarded as an abbreviation for the addition of C_s followed by a deletion of C.

If a deletion removes only one clause D at a time, it is sufficient to require $S_{i-1} \setminus D \models D$. But here we have the disadvantage that we cannot look in parallel for different clauses to be deleted. Also if we have done some work to delete a clause (e.g. have a partial prove that it follows from other clauses) we have to reconsider the whole work after the deletion of another clause. Moreover it may destroy work we have done to yield a saturated specification: to show inferences to be redundant in general the system adds clauses which are conclusion of inferences (with premises in S_{i-1}). The addition of these clauses makes the inferences redundant (see corollary 7.13 together with lemmata 7.14 and 7.17; the user and the system might provide other clauses for simplification purposes). But if a clause (which is such a conclusion) is deleted because it follows from arbitrary clauses, the inferences do not remain redundant and have to be reconsidered again. The notion of redundancy (for clauses) itself it also not appropriate: first of all the deletion of redundant clauses does not preserve the class of models (as redundancy is defined using a single interpretation, i.e. a single model) and additionally it is not modular in the sense that a clause redundant in S_i might become non-redundant in a set S_j ($j > i$). Therefore we will introduce a notion of *compositeness* and delete composite clauses. This allows the deletion of composite clauses in parallel, the deletion of composite clauses preserves the redundancy of inferences and a clause composite in S_i will remain composite in all S_j with $j \geq i$.

As we consider completion as a task in theorem provers, we first give some notions about completion. Then we will prove compositeness to be a general method to subsume

elimination and simplification methods known in theorem proving. Again for $\mathcal{R}$ we always substitute $\mathcal{M}_E$ or $\mathcal{M}_{Ext}$.

Definition 7.1 (Redundancy Criterion)
Let S be a set of clauses. A *redundancy criterion Red* is a relation *Red* between inferences and sets of clauses with

1. For any inference π with maximal premise C and $\mathcal{P}$-set S_C which is $\mathcal{R}$-saturated on NG_C we have

$$Red(\pi, S_C) \quad \text{implies} \quad \pi \text{ is redundant in } S_C$$

2.

$$Red(\pi, S) \quad \text{implies} \quad Red(\pi, S_1) \quad \text{for all } S_1 \supseteq S.$$

Definition 7.2 (Deletion Criterion)
Let *Red* be a redundancy criterion. A *deletion criterion del* (w.r.t. *Red*) is a relation between sets of clauses with:

$$del(D, S) \quad \text{implies} \quad S \setminus D \models D$$

and

$$Red(\pi, S) \quad \text{implies} \quad Red(\pi, S \setminus D)$$

If we are only interested in theorem proving (and not in completion for consistent sets of clauses), we might in the above definition replace the condition $S \setminus D \models D$ by $S \setminus D$ *is inconsistent if and only if* S *is inconsistent.*

Definition 7.3 (Completion Derivation)
Let *del* be a deletion criterion. A *completion derivation* is a sequence $S_0, S_1, S_2, \ldots$ of sets of clauses with either:

- Addition: $S_i = S_{i-1} \cup A$, where A is a set of logical consequences from S_{i-1}
- Deletion: $S_i = S_{i-1} \setminus D$ with $del(D, S_{i-1})$

Definition 7.4 (Limit System)
The *limit system* S_∞ of a completion derivation $S_0, S_1, S_2, \ldots$ is defined as the set of all persistent clauses, i.e.

$$S_\infty := \bigcup_j \left(\bigcap_{k \geq j} S_k \right).$$

Definition 7.5 ($\mathcal{P}$-Fairness)
A completion derivation $S_0, S_1, S_2, \ldots$ is called *fair* (w.r.t. an inference system $\mathcal{R}$) if every inference π (in $\mathcal{R}$) with premises in S_∞ satisfies $Red(\pi, S_i)$ for some i and S_∞ is a $\mathcal{P}$-set.

Lemma 7.6 (Correctness of Fair Completion Derivations)
Let $S_0, S_1, S_2, \ldots$ be a fair completion derivation (w.r.t. $\mathcal{R}$). Then S_∞ is an $\mathcal{R}$-saturated $\mathcal{P}$-set and S_∞ and S_0 have the same models (if any).

Proof: As we only delete and add logical consequences, the class of models remains invariant during the whole derivation. For any inference π with premises in S_∞ we have $Red(\pi, S_i)$ for some S_i, so $Red(\pi, S_j)$ for all $j > i$: for additions during completion it follows from definition 7.1, for deletions from definition 7.2. So for π we have $Red(\pi, S_\infty)$. By fairness S_∞ is also a $\mathcal{P}$-set. Assume S_∞ is not $\mathcal{R}$-saturated. Then there are inferences (with premises in S_∞) which are non-redundant in S_∞. The totality of $>_c$ (resp. $>_c^{ext}$) implies that there exists a clause C and a non-redundant inference π with maximal premise C such that C is minimal in the following sense: every inference with premises (in S_∞) smaller than C is redundant. So we may assume S_∞ to be $\mathcal{R}$-saturated on NG_C. We already know (see above) that we have $Red(\pi, S_\infty)$ and S_∞ is an $\mathcal{R}$-saturated $\mathcal{P}$-set on NG_C. But then with definition 7.1 we conclude π to be redundant in S_∞. This contradicts our assumption, so S_∞ is $\mathcal{R}$-saturated. □

7.1 Compositeness

We will define notions of compositeness for $\mathcal{M}_E$ and $\mathcal{M}_{Ext}$. We expect that similar definitions and lemmata are useful for similar inference systems.

Definition 7.7 (C-Boundedness)
Let D be a clause, C be a ground clause, t_{max} a maximal (w.r.t. $>_E$) ground term of C and σ a ground substitution. $D\sigma$ is called *C-bounded* if

- $C >_c D\sigma$ and a term E-equal to t_{max} occurs in the antecedent of C,
- or $C >_c D\sigma$ and there is no variable x in D with $x\sigma =_E t_{max}$.

Definition 7.8 (Bounded $\mathcal{M}_E$-Compositeness)
Let N be a set of clauses. Let C and B be ground clauses. *C is $\mathcal{M}_E$-composite (w.r.t. N) with bound B*, if there exist ground instances $C_1\sigma_1, \ldots, C_k\sigma_k$ of $N \cup E$ such that $C_1\sigma_1, \ldots, C_k\sigma_k \models C$ and $C_j\sigma_j$ is B-bounded for all j with $1 \leq j \leq k$.

Note that in general we cannot use arbitrary instances of E-equations with terms E-equal to the maximal term of B to show an inference or clause to be redundant. But there are special inference systems admitting special orderings on clauses, for which an E-equation is a very small clause, so that its use for redundancy is (nearly) unrestricted. An example is the system $\mathcal{M}_{Ext}$, where we can use all E-equations applicable at terms occurring in C or smaller clauses to prove the clause C to be $\mathcal{M}_{Ext}$-composite (see below).

Definition 7.9 ($\mathcal{M}_E$-Compositeness of Clauses)
Let N be a set of clauses and C be a ground clause. C is $\mathcal{M}_E$-composite (w.r.t. N), if C is $\mathcal{M}_E$-composite (w.r.t. N) with bound C.

A non-ground clause is $\mathcal{M}_E$-composite if all its ground instances are $\mathcal{M}_E$-composite.

Lemma 7.10 Let C be a ground instance of a clause in N and N be $\mathcal{M}_E$-saturated on NG_C and N does not contain the empty clause. if an instance $D\sigma$ of a clause D in N is C-bounded, then $I_C \models D\sigma$.

Proof: By lemma 5.31 $I_C \models D\sigma'$ for a substitution σ' with $\sigma' =_E \sigma$. Let t_{max} be a maximal term of C. If a term E-equal to t_{max} occurs in the antecedent of C, then by lemma 5.30 I_C also satisfies $D\sigma$. If there is no variable x in D such that $x\sigma =_E t_{max}$, by the same lemma

we also get $I_C \models D\sigma$. If there is such a variable in D and there is no term E-equal to t_{max} in the antecedent of C, then $D\sigma$ is not C-bounded, which contradicts our assumptions. So with our assumptions in any case $I_C \models D\sigma$. □

Lemma 7.11 (Redundancy of $\mathcal{M}_E$-Composite Clauses)
Let C be a ground instance of a clause in a set N which is $\mathcal{M}_E$-saturated on NG_C and does not contain the empty clause. If C is $\mathcal{M}_E$-composite (w.r.t. N), then it is redundant.

Proof: There are ground instances $C_1, \ldots, C_k$ of $N \cup E$ which show C to be redundant. By lemmata 7.10 and 5.30 we know that I_C satisfies all these clauses C_j. So I_C is a model satisfying these clauses and by $C_1, \ldots, C_k \models C$ I_C must also satisfy C, which means that C is redundant. □

Definition 7.12 ($\mathcal{M}_E$-Compositeness of Inferences)
A ground inference π with conclusion C and maximal premise B is called *$\mathcal{M}_E$-composite* (w.r.t. N) if either one of its premises is $\mathcal{M}_E$-composite (w.r.t. N), or else C is $\mathcal{M}_E$-composite (w.r.t. N) with bound B.

A non-ground inferences is $\mathcal{M}_E$-composite if all its ground instances are $\mathcal{M}_E$-composite.

Corollary 7.13 (Addition of Clauses for $\mathcal{M}_E$)
Let π be an inference with conclusion D. Then π is composite in $N \cup \{D\}$ (for arbitrary N).

Proof: For every ground instance $\pi\sigma$ and every inference rule of $\mathcal{M}_E$ $D\sigma$ is smaller (w.r.t. $>_c$) than the maximal premise C of $\pi\sigma$. Trivially we have $\{D\sigma\} \models D\sigma$. So every ground instance of π is composite in $N \cup \{D\}$, hence π is composite. □

So simply adding conclusions is the straight forward way in a completion procedure to make inferences composite (so redundant in the limit system).

Lemma 7.14 (Redundancy of Composite Inferences)
Let π be a ground instance of an inference with premises in $N \cup E$. Let C be its maximal premise. Let N be $\mathcal{M}_E$-saturated on NG_C and N does not contain the empty clause. If π is $\mathcal{M}_E$-composite (w.r.t. N), then it is redundant.

Proof: Suppose the maximal premise of π is B and the conclusion is D. If one of the premises is composite, then (by lemma 7.11) it is also redundant and so π is redundant. Otherwise there are ground instances $C_1, \ldots, C_k$ of $N \cup E$ which prove D to be composite. By lemmata 7.10 and 5.30 we know that I_B satisfies all these clauses C_j. So I_B is a model for these clauses and (by $C_1, \ldots, C_k \models D$) I_B must also satisfy D, which means that π is redundant. □

Lemma 7.15 (Compositeness is well-founded)
If a clause C is $\mathcal{M}_E$-composite (w.r.t. N), then for every ground instance $C\sigma$ there exist non-$\mathcal{M}_E$-composite ground instances $C_1, \ldots, C_k$ of $N \cup E$ which prove $C\sigma$ to be composite.

Proof: see [Bachmair/Ganzinger 91c]. □

Lemma 7.16 (Modularity of Compositeness)

(i) If $N \subseteq N_1$, then every inference or clause which is $\mathcal{M}_E$-composite w.r.t. N is also $\mathcal{M}_E$-composite w.r.t. N_1.

(ii) If $N \subseteq N_1$ and all clauses in $N_1 \setminus N$ are $\mathcal{M}_E$-composite w.r.t. N_1, then every inference or clause which is $\mathcal{M}_E$-composite w.r.t. N_1 is also $\mathcal{M}_E$-composite w.r.t. N.

Proof: Part (i) is obvious. For part (ii) assume there exists a ground instance C of a clause (or inference) which is $\mathcal{M}_E$-composite in N_1. By lemma 7.15 there exists a set of non-$\mathcal{M}_E$-composite ground instances $C_1, \ldots, C_k$ of clauses in $N_1 \cup E$ which prove the clause (or inference) C to be $\mathcal{M}_E$-composite. As all ground instances of clauses in $N_1 \setminus N$ are $\mathcal{M}_E$-composite (by the assumption above), the clauses C_j are ground instances of clauses in $N \cup E$ and C is also $\mathcal{M}_E$-composite w.r.t. N. □

Lemma 7.17 $\mathcal{M}_E$-Compositeness of inferences is a redundancy criterion. $\mathcal{M}_E$-Compositeness of clauses is a deletion criterion w.r.t. $\mathcal{M}_E$-compositeness of inferences (as redundancy criterion).

Proof: By lemma 7.14 and lemma 7.16 (part (i)) compositeness of inferences is a redundancy criterion. By lemma 7.16 the compositeness of clauses is a deletion criterion. □

Compositeness for $\mathcal{M}_{Ext}$

Note that the ordering $>_c^{ext}$ (definition 6.14) depends on the set N: the weight of an extended equation in an extended clause might take the values from 0 to 2.

Definition 7.18 (Complexities of Clauses)
Let $C\sigma$ be a ground instance of a clause. Consider $>_c^{ext}$ as an ordering over E-multiset expressions for $\mathcal{M}_{Ext}$ of clauses (i.e. we do not consider clauses and compare their E-multiset expressions (which depend on the set N), but comparing given multiset expressions). For each set $N \cup \{C\}$ there is an E-multiset expression of $C\sigma$. The minimal (w.r.t. $>_c^{ext}$) such multiset expression of $C\sigma$ is called the *minimal complexity of* $C\sigma$. We denote it by $CompMin(C)$. The maximal (w.r.t. $>_c^{ext}$) such multiset expression of $C\sigma$ is called the *maximal complexity of* $C\sigma$. We denote it by $CompMax(C)$. The maximal or minimal complexity of a clause C is denoted as a complexity of a clause C.

Note that for many clauses the complexity is independent from the given set N. Only for clauses which might be extended clauses the minimal and maximal complexity may differ.

Definition 7.19 (Bounded $\mathcal{M}_{Ext}$-Compositeness)
Let N be a set of clauses. Let C be a ground clause and B be a complexity of a clause. C *is* $\mathcal{M}_{Ext}$*-composite (w.r.t.* N*) with bound* B, if there exist ground instances $C_1, \ldots, C_k$ of $N \cup E$ such that $C_1, \ldots, C_k \models C$, $B >_c^{ext} CompMax(C_j)$ for all j with $1 \leq j \leq k$.

Definition 7.20 ($\mathcal{M}_{Ext}$-Compositeness of Clauses)
Let N be a set of clauses and C be a ground clause. C is $\mathcal{M}_{Ext}$-composite (w.r.t. N), if C is $\mathcal{M}_{Ext}$-composite (w.r.t. N) with bound $CompMax(C)$.

A non-ground clause is $\mathcal{M}_{Ext}$-composite if all its ground instances are $\mathcal{M}_{Ext}$-composite.

Lemma 7.21 (Redundancy of $\mathcal{M}_{Ext}$-Composite Clauses)
Let $N := NE \cup Ext \cup SExt$ be a set of clauses. Let C be a ground instance of a clause in $N \setminus Ext$. Let N be $\mathcal{M}_{Ext}$-saturated on NG_C, closed under E-extension and N does not contain the empty clause. If C is composite (w.r.t. N), then it is redundant.

Proof: The proof is similar to lemma 7.11. Note that the clause C has an E-multiset expression in the set N which is equal to $CompMax(C)$. All clauses C_j which prove C to be composite have E-multiset expressions in N which are not greater than $CompMax(C_j)$. Hence $CompMax(C) >_c^{ext} CompMax(C_j)$ implies $C >_c^{ext} C_j$. For the remaining part of the proof see the proof of lemma 7.11. □

Definition 7.22 ($\mathcal{M}_{Ext}$-Compositeness of Inferences)
A ground inference π with conclusion C and maximal premise B is called *$\mathcal{M}_{Ext}$-composite* (w.r.t. N) if either a premise $C\sigma$ of π with $C \in N \setminus Ext$ is $\mathcal{M}_{Ext}$-composite (w.r.t. N), or else C is $\mathcal{M}_{Ext}$-composite (w.r.t. N) with bound $CompMin(B)$.

A non-ground inferences is $\mathcal{M}_{Ext}$-composite if all its ground instances are $\mathcal{M}_{Ext}$-composite.

Corollary 7.23 (Addition of Clauses for $\mathcal{M}_{Ext}$)
Let π be an inference with conclusion D. Then π is composite in $N \cup \{D\}$ (for arbitrary N).

Proof: For every ground instance $\pi\sigma$ and every inference rule of $\mathcal{M}_{Ext}$ $D\sigma$ is smaller (w.r.t. $>_c^{ext}$) than the maximal premise C of $\pi\sigma$. For superposition left, equality factoring and equality resolution inferences this fact is obvious. The only problematic case is the superposition right inference with an (possibly) extended clause $C_E = \Gamma \rightarrow \Delta, t \approx s$ as maximal premise:

$$\frac{\Gamma_1 \rightarrow \Delta_1, t_1 \approx s_1 \quad \Gamma \rightarrow \Delta, t \approx s}{\Gamma, \Gamma_1 \rightarrow \Delta, \Delta_1, t[p \leftarrow s_1] \approx s}$$

Obviously all equations except the equations in Δ and Δ_1 are smaller (w.r.t. $>_{eq}^{ext}$) than $t \approx s$ of C_E. Either the latter equation has a weight of 2 (see definition 6.13) and the conclusion is obviously smaller than C_E, or we used the weight 0 for its maximal equation and all terms in Δ are smaller than t. The other clause is smaller than C_E, so either is is also an extended clause with weight 0, or contains only terms smaller than t. In any case also the equations in Δ_1 are smaller than $t \approx s$ and the conclusion is smaller than the maximal premise.

Trivially we have $\{D\sigma\} \models D\sigma$. So every ground instance of π is composite in $N \cup \{D\}$, hence π is composite. □

So simply adding conclusions is the straight forward way also in an $\mathcal{M}_{Ext}$-completion procedure to make inferences composite (so redundant in the limit system).

Lemma 7.24 (Redundancy of Composite Inferences)
Let π be a ground instance of an inference with premises in N. Let C be its maximal premise. Let N be $\mathcal{M}_{Ext}$-saturated on NG_C, closed under E-extension and N does not contain the empty clause. If π is $\mathcal{M}_{Ext}$-composite (w.r.t. N), then it is redundant.

Proof: Let us denote the maximal premise of π by B. If one of the premises is composite, then (by lemma 7.21) it is also redundant and so π is redundant. Otherwise we use the

arguments of the proof of lemma 7.21 to prove that the conclusion follows from smaller (in the given set N) clauses of NG. So there are ground instances $C_1, \ldots, C_k$ of $N \cup E$ smaller than B which show D to be redundant. By lemmata 6.48 and 6.44 we know that I_B satisfies all these clauses C_j. So I_B is a model for these clauses and (by $C_1, \ldots, C_k \models D$) I_B must also satisfy D, which means that π is redundant. □

Similar to lemma 7.15, also $\mathcal{M}_{Ext}$-compositeness is well-founded and satisfies lemma 7.16. So $\mathcal{M}_{Ext}$-compositeness is a deletion criterion (cf. lemma 7.17).

As an example of compositeness we present the following lemma, which is specific to $\mathcal{M}_{Ext}$-completion for $E = AC$:

Lemma 7.25 Let $E = AC$. Let $C = \Gamma \rightarrow \Delta, \ell \approx r$ and $C_{AC} = \Gamma \rightarrow \Delta, \ell + x \approx r + x$ be clauses in N such that C_{AC} is the AC-extended clause of C (for the equation $\ell \approx r$). Every superposition right inference on the term $\ell + x$ of C_{AC}, where we superpose upon $\ell + x$ strictly below the root, is $\mathcal{M}_{Ext}$-composite in N.

Proof: Let D be a clause $\Gamma_1 \rightarrow \Delta_1, t \approx s$ of N. Let

$$\pi\sigma = \frac{D\sigma \quad C_{AC}\sigma}{\Gamma\sigma, \Gamma_1\sigma \rightarrow \Delta\sigma, \Delta_1\sigma, ((\ell + x)[p \leftarrow s])\sigma \approx (r + x)\sigma}$$

be a ground instance of such a superposition right inference (with $p \neq \varepsilon$ and $t\sigma >_{AC} s\sigma$). As p is a non-variable position of $\ell + x$ and $p \neq \varepsilon$, we have $p = 1.p_1$ and p_1 is a position in ℓ. There is also a superposition right inference between $D\sigma$ and $C\sigma$:

$$\frac{D\sigma \quad C\sigma}{\Gamma\sigma, \Gamma_1\sigma \rightarrow \Delta\sigma, \Delta_1\sigma, (\ell[p_1 \leftarrow s])\sigma \approx r\sigma}$$

Let A be the set of ground instances of AC-equations which are needed for $(\ell/p_1)\sigma =_{AC} t\sigma$, then we have

$$A \cup \{D\sigma, C\sigma\} \models \Gamma\sigma, \Gamma_1\sigma \rightarrow \Delta\sigma, \Delta_1\sigma, (\ell[p_1 \leftarrow s])\sigma \approx r\sigma.$$

But this clause implies the conclusion of the first inference $\pi\sigma$ in this proof (the clauses differ only by an additional context $([] + x)\sigma$ for the last equation in the succedent; note that we use implication for models with equality). All clauses in $A \cup \{D\sigma, C\sigma\}$ are smaller than $C_{AC}\sigma$ (for every set N, because $(\ell + x)\sigma >_{AC} \ell\sigma$, so this holds also for minimal and maximal complexities of these clauses), so the inference with superposition on C_{AC} is $\mathcal{M}_{Ext}$-composite in N. □

A main efficiency problem in (semi)automatically theorem proving is the great number of inferences which have to be considered during completion and the inability to show a large subset of these inferences to be irrelevant. The notion of *compositeness* is the key to adopt a lot of known methods developed to show inferences to be irrelevant. Often these methods are originally applied to equational completion or Horn clause specifications only, but are valid also for a wider class of completion problems. The most popular technique shows the joinability of a critical pair by rewriting. It can be extended to rewrite relations for conditional equational specifications (conditional rewriting, or even contextual rewriting, cf. [Bachmair/Ganzinger 90], [Bachmair/Ganzinger 91c]) and even further to rewriting (or

conditional or contextual rewriting) with E-matching or even modulo E. These and also less algorithmic but more general methods (e.g. using *connectedness* instead of joinability) are subsumed by our notion of compositeness, hence can be implemented in a completion procedure based on our inference systems to enhance the performance of the completion component of a theorem prover.

In particular, if we deal with associative and commutative properties, there are often a lot of unifiers and so a lot of inferences. Due to the permutative nature of the AC-theory, often inferences are merely duplicates of other ones or can be ruled out because of symmetry considerations. The number of clauses influences the number of inferences, so we should keep the sets of clauses as small as possible (the same argument applies to completion with $E = \emptyset$). But even for a small set of clauses there can be an enormous number of inferences, which are expensive to generate and expensive to be shown to be composite (more expensive than for $E = \emptyset$). Therefore techniques which show the compositeness of inferences right at construction time have been developed (e.g. avoiding reducible substitutions for the creation of critical pairs; or other critical pair criteria, cf. [Bachmair/Dershowitz 88]). But even further improvements are possible for methods which conclude the compositeness of inferences from the compositeness of other inferences (e.g. if we have superposed on a subterm s_1, we do not need superpositions on the *symmetric* subterm s_2). These techniques are only known (to the author) for the special case $E = AC$ ([Kapur/Musser/Narendran 88], [Zhang/Kapur 89], [Zhang/Kapur 90]).

8 Variants of Inference Systems

8.1 Merging Paramodulation/Perfect Models

In [Bachmair/Ganzinger 90], [Bachmair/Ganzinger 91a] and [Bachmair/Ganzinger 91b] we find a different inference system (compared with 4.6) where

a) the formulation
it exists a ground substitution such that the equation (term) is maximal (w.r.t. an ordering based on an ordering on ground terms)
is replaced by
the equation (term) is maximal (based on an ordering on terms with variables), i.e. all other equations (terms) are smaller or not comparable with the considered equation (term)
and

b) *equality factoring* is replaced by *ordered factoring* and *merging paramodulation.*

If we use the weaker formulation (the second one in (a) above) for our inference system (with equality factoring), we will get some more inferences (see example 8.2). But the merging paramodulation inference in [Bachmair/Ganzinger 91a] has a formulation where not only maximality is required but $u\tau > v\tau$ for two terms perhaps containing variables. This is stronger than maximality and results in less merging paramodulation inferences (for non-ground clauses). It is the reason why the lifting of some ground merging paramodulation inferences yields superposition right inferences (and not again merging paramodulation inferences on the non-ground level). Working with an ordering on ground terms only, we could replace the formulation $u\tau > v\tau$ by $u\tau\sigma_1 > v\tau\sigma_1$ *for all ground substitutions* σ_1. But then we may find situations where a ground merging paramodulation inference is not liftable (see example 8.3). So we will here use a weaker condition and require the relation $u\tau > v\tau$ only for one ground substitution. This admits completeness and lifting is easier as in [Bachmair/Ganzinger 91a].

Working modulo a theory E, such an inference system with merging paramodulation and ordered factoring is complete, if we add inferences like E-closure or consider sets closed under E-extension. We expect that also a version with merging paramodulation and one of the above E-specific enhancements in the formulation of [Bachmair/Ganzinger 91a] (i.e. with maximality and orderings on terms with variables) is complete. We prove the redundancy-completeness for a derivative of $\mathcal{M}_E$. We therefore adopt the definitions for redundancy, saturation and completeness from $\mathcal{M}_E$.

Definition 8.1 Let $\mathcal{MM}_E$ be the inference system as in definition 5.8 with the inference *equality factoring* replaced by the following two inferences:

4.a) *ordered factoring*

$$\frac{\Gamma \to \Delta, t_1 \approx s_1, t_2 \approx s_2}{(\Gamma \to \Delta, t_1 \approx s_1)\sigma}$$

where $\sigma \in \mu CSU_E(t_1 \approx s_1, t_2 \approx s_2)$ and there exists a ground substitution σ_1 such that $t_1\sigma\sigma_1 \approx s_1\sigma\sigma_1$ is maximal (w.r.t. $>_{eq}$) in $(\Gamma \to \Delta, t_1 \approx s_1, t_2 \approx s_2)\sigma\sigma_1$.

4.b) *merging paramodulation*

$$\frac{\Gamma_1 \rightarrow \Delta_1, s_1 \approx s_2 \quad \Gamma_2 \rightarrow \Delta_2, u_1 \approx t_1, u_2 \approx t_2}{(\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2, u_1 \approx t_1[p \leftarrow s_2], u_2 \approx t_2)\sigma}$$

where σ is the composition $\tau\rho$ of two substitutions, τ is in $\mu CSU_E(t_1/p, s_1)$, $\rho \in \mu CSU_E(u_1\tau, u_2\tau)$, t_1/p is not a variable and there exists a ground substitution σ_1 such that

a) $s_1\sigma\sigma_1 >_E s_2\sigma\sigma_1$

b) $s_1\sigma\sigma_1 \approx s_2\sigma\sigma_1$ is strictly maximal (w.r.t. $>_{eq}$) in $(\Gamma_1 \rightarrow \Delta_1, s_1 \approx s_2)\sigma\sigma_1$

c) $u_1\sigma\sigma_1 \approx t_1\sigma\sigma_1$ is strictly maximal (w.r.t. $>_{eq}$) in $(\Gamma_2 \rightarrow \Delta_2, u_1 \approx t_1, u_2 \approx t_2)\sigma\sigma_1$

d) $t_1\sigma\sigma_1 >_E t_2\sigma\sigma_1$

e) $u_1\sigma\sigma_1 >_E t_1\sigma\sigma_1$.

The next example shows one difference between the formulations in a) (see beginning of this section).

Example 8.2 We assume a lexicographic path ordering with decreasing precedence for the operators $a/0$, $b/0$, $c/0$, $f/2$ and $g/2$. Consider the clause

$$C := \rightarrow g(x, b) \approx f(x, x).$$

If we want to superpose with a σ-instance of $g(x, b)$ we need $g(x, b)\sigma\sigma_1 > f(x, x)\sigma\sigma_1$, therefore $b > x\sigma\sigma_1$ and the term substituted for x cannot contain an operator a. Now consider

$$D := \rightarrow g(y, b) \approx z, a \approx c.$$

If we want to superpose on $g(y, b)$ we need $g(y, b)\sigma\sigma_1 \geq a$, so $y\sigma\sigma_1 \geq a$ and the term substituted for y must contain an operator a. For a superposition right inference of C on the term $g(y, b)$ in D we have to substitute x and y by the same variable (say x) and then need a substitution for x which must contain an a and cannot contain an a. This is impossible, so there is no such superposition right inference. For the inference system in [Bachmair/ Ganzinger 91a] there is such an inference with the conclusion $\rightarrow f(x, x) \approx z, a \approx c$.

Example 8.3 We continue the previous example by considering a merging paramodulation inference between the ground clauses $C\sigma$ and $D\sigma$, where $\sigma = \{x \leftarrow c, y \leftarrow c, z \leftarrow a\}$.

$$\frac{\rightarrow g(c, b) \approx f(c, c) \quad \rightarrow g(c, b) \approx a, a \approx c}{\rightarrow f(c, c) \approx a, a \approx c}$$

With our definition of merging paramodulation in definition 8.1 we can lift this inference, because σ is a needed ground substitution fulfilling all conditions a) to e). With the stronger formulation $g(x, b) > y$ (as in [Bachmair/Ganzinger 91a] or in this paper: $g(x, b)\sigma > y\sigma$ for all ground substitutions) we cannot lift this inference to another merging paramodulation inference. For [Bachmair/Ganzinger 91a] there is a superposition inference (previous example) which has a conclusion we can instantiate to the conclusion of the above ground inference (and that is the way they do the lifting of such merging paramodulation inferences). But here there would be no such superposition right inference and we could not lift this inference. So we use the weaker requirement $u_1\sigma\sigma_1 >_E t_1\sigma\sigma_1$ for at least one ground substitution σ_1 and have less superposition inferences but more merging paramodulation inferences in $\mathcal{MM}_E$ (compared to the inference system in [Bachmair/Ganzinger 91a]).

Why using merging paramodulation?

As we see in the following two examples, starting with the same set of (ground) clauses, for the different inference systems $\mathcal{M}_E$ and $\mathcal{MM}_E$ we yield different saturated sets. Also the constructed rewrite system and even the interpretation may differ (even if we assume $E = \emptyset$). In [Bachmair/Ganzinger 91b] the notion of *perfect models* is introduced. These models are only yield by inferences systems with merging paramodulation (see example 8.5).

Example 8.4 We consider four ground terms a to d with the total ordering $a > b > c > d$. We start with a set of two clauses:

$$\begin{array}{ll} \{ & \rightarrow a \approx b, a \approx d, \\ & \rightarrow b \approx c \qquad \} \end{array}$$

If we saturate this set w.r.t. our inference system $\mathcal{M}_E$ (with equality factoring) we get one additional clause:

$$b \approx d \rightarrow a \approx b$$

Because $b \approx d$ is not satisfied in the interpretation of this clause, the clause itself is satisfied in its interpretation and not productive. The rewrite systems $R_{TOP} = R_{fact}$ constructed for the interpretation of these three clauses (which form an $\mathcal{M}_E$-saturated set) consists of two rewrite rules:

$$R_{fact} = \{a \Rightarrow b, b \Rightarrow c\}$$

If we now saturate the above set of two clauses w.r.t. the inference system in [Bachmair/ Ganzinger 91a] or w.r.t. $\mathcal{MM}_E$, we get also one additional (non-redundant) clause (with a merging paramodulation inference):

$$\rightarrow a \approx c, a \approx d$$

Now the rule set $R_{TOP} = R_{merge}$ contains a different rule:

$$R_{merge} = \{a \Rightarrow c, b \Rightarrow c\}$$

But in this example we have the same interpretation in both cases, $R^*_{fact} = R^*_{merge}$. In the next example we get different interpretations.

Example 8.5 We consider the same ground terms with the same ordering as in the previous example 8.4. We start with an other set of two clauses:

$$\begin{array}{ll} \{ & \rightarrow a \approx b, a \approx c, \\ & \rightarrow b \approx d \qquad \} \end{array}$$

If we saturate this set w.r.t. our inference system $\mathcal{M}_E$ (with equality factoring) we get one additional clause

$$b \approx c \rightarrow a \approx b,$$

which is not productive.

$$R_{fact} = \{a \Rightarrow b, b \Rightarrow d\}$$

is the constructed rewrite system. If we now saturate the above set of two clauses w.r.t. the inference system in [Bachmair/Ganzinger 91a] or w.r.t. $\mathcal{MM}_E$, we get also one additional (non-redundant) clause (with a merging paramodulation inference):

$$\rightarrow a \approx d, a \approx c$$

Now the rule set $R_{TOP} = R_{merge}$ is different from R_{fact}:

$$R_{merge} = \{a \Rightarrow c, b \Rightarrow d\}$$

And also

$$R^*_{fact} = \{a \approx b, b \approx d, a \approx d\} \neq \{a \approx c, b \approx d\} = R^*_{merge}.$$

With the ordering on interpretations in [Bachmair/Ganzinger 91b] R^*_{merge} is smaller than R^*_{fact}. R^*_{merge} is the perfect (minimal) model.

Completeness with merging paramodulation

The lifting of ordered factoring inferences is obvious (cf. lemma 4.11) and the lifting of ground merging paramodulation inferences is more obvious than in [Bachmair/Ganzinger 91a] (it may not result in superposition right inferences between non-ground clauses). But we include a remark in the proof of the following lemma how to prove lifting of merging paramodulation inferences in inference systems similar to [Bachmair/Ganzinger 91a].

Lemma 8.6 (Lifting Lemma 3)
Let N be a set of clauses, $C := \Gamma_1 \rightarrow \Delta_1, s_1 \approx s_2$ and $D := \Gamma_2 \rightarrow \Delta_2, u_1 \approx t_1, u_2 \approx t_2$ be clauses of N and σ_2 be a ground substitution. For any merging paramodulation inference

$$\frac{C\sigma_2 \quad D\sigma_2}{F},$$

in which we superpose at position p into the term $t_1\sigma_2$ of $D\sigma_2$, and p is not at or below a variable position of D, there exist ground substitutions τ, ρ and σ_3 such that

$$\frac{C \quad D}{B}$$

is a merging paramodulation inference with the unsubstituted clauses and

- $\tau \in \mu CSU_E(s_1, t_1/p)$
- $\rho \in \mu CSU_E(u_1\tau, u_2\tau)$
- $\tau\rho\sigma_3 =_E \sigma_2$, so $C\tau\rho\sigma_3 =_E C\sigma_2$ and $D\tau\rho\sigma_3 =_E D\sigma_2$
- $B\sigma_3 =_E F$ and all E-equality steps apply within subterms s_B of $B\sigma_3$ with $u_1\sigma_2 \geq_E s_B\tau\rho\sigma_3$.

Proof: We have $(t_1/p)\sigma_2 =_E s_1\sigma_2$. So there are substitutions τ and τ_1 such that $\sigma_2 =_E \tau\tau_1$ and $\tau \in \mu CSU_E(t_1/p, s_1)$. Because $u_1\sigma_2 =_E u_2\sigma_2$ we get $u_1\tau\tau_1 =_E u_2\tau\tau_1$ and therefore substitutions ρ and ρ_1 with $\tau_1 =_E \rho\rho_1$ and $\rho \in \mu CSU_E(u_1\tau, u_2\tau)$. With transitivity of E-equality we get $\sigma_2 =_E \sigma_1 := \tau\rho\rho_1$. Now we define $\sigma := \tau\rho$ and look at the above definition

8.1. Because all comparisons in conditions a) to e) for merging paramodulation inferences are done by E-compatible orderings, with substitutions σ and σ_1 all these conditions are satisfied and the inference

$$\frac{C \quad D}{(\Gamma_1, \Gamma_2 \rightarrow \Delta_1, \Delta_2, u_1 \approx t_1[p \leftarrow s_2], u_2 \approx t_2)\tau\rho}$$

is the desired merging paramodulation inference (with $\sigma_3 := \rho_1$).

Remark: If we want to consider an inference system similar to the system in [Bachmair/ Ganzinger 91a] (but working modulo E), where we do not use comparisons after instantiating the clauses with σ_1 but looking for maximal terms (equations) and requiring $u_1\tau >_E t_1\tau$ for merging paramodulations, we may use the following information:

Let the ordering $>_E$ be stable under substitutions. For the substitution $\tau\rho$

- the equation $(s_1 \approx s_2)\tau\rho$ is maximal in $C\tau\rho$,
- $s_2\tau\rho$ is not greater than $s_1\tau\rho$,
- $(u_1 \approx t_1)\tau\rho$ is maximal in $D\tau\rho$ and
- $t_2\tau\rho$ is not greater than $t_1\tau\rho$

(otherwise there cannot be a ground substitution as ρ_1: e.g. for stable orderings $t_2\tau\rho >_E t_1\tau\rho$ implies $t_2\tau\rho\rho_1 >_E t_1\tau\rho\rho_1$). With the same argument (using the ground substitution $\rho\rho_1$) we cannot have $t_1\tau >_E u_1\tau$. So either we have also $u_1\tau >_E t_1\tau$ (and the above inference is also a merging paramodulation inference in that inference system) or $u_1\tau$ and $t_1\tau$ are incomparable and the above inference (needed for the lifting) is a superposition right inference. □

The interpretation and its properties remain the same as in section 5.1. The definitions and lemmata in section 5.2 apply also to the inference system $\mathcal{MM}_E$, except that we have to replace lemma 5.29 and 5.31, because these are the only lemmata which use equality factoring inferences. We will first prove a stronger version of lemma 5.29: in some cases the right sides of the productive equations are irreducible (see the difference to the former inference system with equality factoring: in example 8.5 a rule $a \Rightarrow b$ is produced by a clause $\rightarrow a \approx b, a \approx c$ and b is reducible, but the interpretation of this set saturated with respect to $\mathcal{MM}_E$ does not contain such a rule).

Lemma 8.7 (Properties of Non-Redundant Clauses)
Let $F\sigma := \Gamma\sigma \rightarrow \Delta\sigma, t\sigma \approx s\sigma$ be a non-redundant ground instance of a clause in N and and assume that for all substitutions σ' with $\sigma' =_E \sigma$ we have $I_{F\sigma} \not\models \Gamma\sigma \rightarrow \Delta\sigma'$. Let the equation $t\sigma \approx s\sigma$ be maximal (w.r.t. $>_{eq}$) in $F\sigma$ with $t\sigma >_E s\sigma$, and let $t\sigma'$ be irreducible by $\Longrightarrow_{R_{F\sigma}}$ (for all $\sigma' =_E \sigma$). If N is $\mathcal{MM}_E$-saturated on $NG_{F\sigma} \cup \{F\sigma' \mid \sigma' =_E \sigma\}$ and if $I_{F\sigma}$ satisfies all ground instances of E-equations between terms smaller than $t\sigma$, the following holds:

(i) σ is irreducible (w.r.t. $\Longrightarrow_{R_{F\sigma}}$), i.e. $x\sigma$ is irreducible for all $x \in vars(F)$

(ii) F is productive.

(iii) For all $(\Gamma\sigma)'$ with $(\Gamma\sigma)' =_E \Gamma\sigma$ and all ground clauses $C >_c F$ or $C =_E F\sigma$,

$$(\Gamma\sigma)' \subseteq I_C \ .$$

(iv) For all $\Delta\sigma'$ with $\sigma' =_E \sigma$ and all ground clauses $C >_c F$ or $C =_E F\sigma$,

$$\Delta\sigma' \cap I_C = \emptyset \ .$$

(v) if $\Delta\sigma$ contains a term $t_2\sigma$ E-equal to $t\sigma$, i.e. $\Delta\sigma = t_2\sigma \approx s_2\sigma, \Delta_2\sigma$, then $s\sigma$ is irreducible by $\Longrightarrow_{R_{F\sigma}}$ (i.e. the right side of the productive equation is irreducible).

Proof: The proof is similar to the one of lemma 5.29. We use induction on the ordering $>_c$, so let us assume that (i)–(v) hold for every suitable instance F_1 of N with $F\sigma >_c F_1$. Since $F\sigma$ is non-redundant we have $\Gamma\sigma \subseteq I_{F\sigma}$ and $\Delta\sigma \cap I_{F\sigma} = \emptyset$.

(iii) see 5.29.

(i) see 5.29.

(ii) If $F\sigma$ is not productive, then an equation $t_2\sigma \approx s_2\sigma$ E-equal to (but different from) the maximal equation $t\sigma \approx s\sigma$ of $F\sigma$ occurs in the succedent of $F\sigma$. In other words, the equation $t\sigma \approx s\sigma$ is not strictly maximal and the clause is not productive, for this reason. A clause $F_1 = \Gamma\sigma' \rightarrow \Delta\sigma'$ smaller than $F\sigma$ can be obtained as conclusion of an instance of an *ordered factoring* inference with premise $\Gamma \rightarrow \Delta, t \approx s$. This clause is false in $I_{F\sigma} = I_{F\sigma'}$. This contradicts the required saturation. So from now on we may assume that no other equation E-equal to $t\sigma \approx s\sigma$ occurs in the succedent of $F\sigma$ and hence that $F\sigma$ is productive.

(iv) $I_{F\sigma} \not\models \Gamma\sigma \rightarrow \Delta\sigma$, so $I_{F\sigma} \cap \Delta\sigma = \emptyset$. By our assumptions we also have $I_{F\sigma} \cap \Delta\sigma' = \emptyset$ and so by lemma 5.10 also $I_{(F\sigma)'} \cap \Delta\sigma' = \emptyset$.

Suppose an $\Delta\sigma'$ contains an equation $u\sigma' \approx v\sigma'$ which is satisfied by an I_C (with $C >_c F\sigma$). We may assume that $u\sigma >_E v\sigma$, for otherwise, if $u\sigma =_E v\sigma$, then $t\sigma >_E u\sigma$ (if $t\sigma =_E u\sigma$ the equation $t\sigma \approx s\sigma$ were not maximal in $F\sigma$) and $u\sigma' \approx v\sigma'$ is also satisfied by $I_{F\sigma}$ (by the assumption the needed E-equalities are available). By construction all interpretations are Church-Rosser systems, so $u\sigma'$ and $v\sigma'$ are reducible by R_C to $E_{F\sigma}$-equal terms (the terms cannot grow by reduction, so the needed instances of E-equations from E_C are also satisfied in $I_{F\sigma}$). The equation is not satisfied by $I_{F\sigma}$, so $u\sigma'$ is reduced with a rule of $R_C \setminus R_{F\sigma}$, hence the left side of the reducing rule is E-equal to $t\sigma$ or even greater. As $\Delta\sigma'$ contains no term greater (w.r.t. $>_E$) than $t\sigma$, we have $u\sigma' =_E t\sigma$. With (ii) we already know that $F\sigma$ is productive and other clauses cannot produce rules whose left side is E-equal to $t\sigma$ and which are not contained in $irred\_rules(t\sigma \approx s\sigma, R_{F\sigma})$ (this is the set of rules produced by $F\sigma$). All rules produced by $F\sigma$ have the same right side $s\sigma$. The equation $u\sigma' \approx v\sigma'$ (remember that $u\sigma =_E t\sigma$) is rewritten to $s\sigma \approx v\sigma'$ (note that with restrictions 5.23 we know that $u\sigma'$ is reduced at the root, hence to $s\sigma$) and the sides $s\sigma$ and $v\sigma'$ are reducible to $E_{F\sigma}$-equal terms. Because $t\sigma >_E s\sigma$ we can only use the rules of $R_{F\sigma}$ to reduce $s\sigma$ or $v\sigma'$, so $I_{F\sigma} \models s\sigma \approx v\sigma'$ (and $I_{F\sigma} \models s\sigma'' \approx v\sigma''$). We have $s\sigma \geq_E v\sigma$ (maximality of the equation) and by part (ii) (see proof above) even $s\sigma >_E v\sigma$. We conclude that $s\sigma$ is reducible at a (non-variable, see part (i)) position p by a rule $(\ell\sigma)' \Rightarrow r\sigma$ in $R_{F\sigma}$ produced by $C_1\sigma$ with $C_1 = \Gamma_1 \rightarrow \Delta_1, \ell \approx r$ (we here assume that we use the same substitution for F and C_1; we can always construct this situation by proper renaming of variables). Being more precise we have to remark here that the rules produced by $F\sigma$ have right side $(s\sigma)_{min}$. But all E-equalities to show $s\sigma =_E (s\sigma)_{min}$ are satisfied by $I_{F\sigma}$. So by the Church-Rosser property of interpretations we know that $s\sigma$ is reducible if and only if $(s\sigma)_{min}$ is reducible. For simplicity we omitted the index min. We consider the *merging paramodulation* inference

$$\pi = \frac{\Gamma_1\sigma \rightarrow \Delta_1\sigma, \ell\sigma \approx r\sigma \quad \Gamma\sigma \rightarrow \Delta\sigma, t\sigma \approx s[(\ell\sigma)']\sigma}{\Gamma\sigma, \Gamma_1\sigma \rightarrow \Delta\sigma, \Delta_1\sigma, t\sigma \approx s[p \leftarrow r]\sigma}$$

which can be lifted (and then instantiated) to an E-equal inference with maximal premise $F\sigma'' = \Gamma\sigma'' \rightarrow \Delta\sigma'', t\sigma'' \approx s\sigma''$. By our assumptions (and induction for $C_1\sigma$) the conclusion of this inference is false in $I_{F\sigma} = I_{F\sigma''}$. This contradicts the saturation, so there cannot be an equation in $\Delta\sigma'$ which is satisfied in I_C.

(v) By part (i) $s\sigma$ is not reducible at a variable position. With the same inference as in the paragraph above, we get a contradiction from the reducibility of $s\sigma$ at a non-variable position. So $s\sigma$ has to be irreducible. □

We do not give a repetition of lemma 5.31. The lemma uses a degenerated equality factoring inference in case b2). If we replace this inference by an ordered factoring inference, the proof remains valid (and becomes simpler, because we do not get an additional E-equality in the antecedent of the conclusion, cf. lemma 5.31 case b2).

8.2 Hierarchic Specifications

In [Bachmair/Ganzinger/Waldmann ed] hierarchic specification with first-order clauses are considered (in the many-sorted case). Their main definitions (e.g. base term, abstracted term, simple substitution, ...) can be adopted when working modulo E. Their inference system $\mathcal{E}$ (and similar their system $\mathcal{P}$, cf. section 8.1) can be extended for $E \neq \emptyset$ (we yield the inference rules of 4.5). But to get their results, we have to require that the E-part of a hierarchic specification is properly separated into a base part and an extended part (and not spread arbitrarily between the base specification and the extending specification):

A *specification* SP is a quadruple (Σ, Ω, Ax, E), where Σ is a set of sorts, Ω is a set of operator symbols over Σ, Ax is a set of axioms (i.e. first-order clauses over (Σ, Ω)) and E is a set of unconditional equations (over (Σ, Ω)).

A *hierarchic specification* SP is a pair (SP_1, SP_2), where $SP_1 = (\Sigma_1, \Omega_1, Ax_1, E_1)$ is called the *base specification*, $SP_2 = (\Sigma_2, \Omega_2, Ax_2, E_2)$ is called the *body* of SP. We require $\Sigma_1 \subseteq \Sigma_2$, $\Omega_1 \subseteq \Omega_2$, and (in addition to the other requirements of [Bachmair/Ganzinger/Waldmann ed])

- $E_1 \subseteq E_2$,
- $ops(E_2 \setminus E_1) \cap \Omega_1 = \emptyset$, where $ops(C)$ is the set of all operator symbols occurring in a clause of the set C of clauses,
- E_2 is a set of unconditional equations as in definition 4.1,
- there is no non-base ground term t which is $(E_2 \setminus E_1)$-equal to a ground base term s.

Further definitions are adopted from [Bachmair/Ganzinger/Waldmann ed].

With this definition we can consider completion and theorem proving in a modular fashion, e.g. use a theorem prover for the base specification SP_1 as a subroutine in a theorem prover for the whole specification SP or saturate the axioms of SP without considering inferences with premises in Ax_1. But we can neither forget E-equations ($E_1 \subseteq E_2$), when considering saturation of Ax_2, nor later add E-equations which influence the base specification (so we require $ops(E_2 \setminus E_1) \cap \Omega_1 = \emptyset$). We will only give short examples for both restrictions:

Example 8.8 (Omitting Restriction $E_1 \subseteq E_2$)

$$SP_1 = (\Sigma_1 = \{s\}, \Omega_1 = \{*{:}s \times s \rightarrow s\}, Ax_1 = \emptyset, E_1 = \{x * y \approx y * x\})$$

We extend the specification by

$$SP_2 = (\Sigma_1, \Omega_2 = \{0, 1\colon \rightarrow s;\ i\colon s \rightarrow s;\ *\colon s \times s \rightarrow s\}, Ax_2, E_2 = \emptyset),$$

where

$$\begin{aligned} Ax_2 = \ \{ \ & 0 \approx 1 \rightarrow , \\ & \rightarrow 0 * x \approx 0, \\ & \rightarrow i(x) * x \approx 1\} \ . \end{aligned}$$

Ax_2 is saturated modulo $E_2 = \emptyset$ and does not contain the empty clause. But the specification is inconsistent if we add the equations of E_1. By this example, we need E-completion techniques even if we extend a base specification and the extension does not introduce new E-equations. Often only the base specification will introduce E-equations (e.g. AC-operators are typically found in base specifications like natural numbers, integers or booleans). But even in this situation we need the methods described in this paper to work (e.g. saturate) with the extension part of a hierarchic specification.

Example 8.9 (Omitting Restriction $ops(E_2 \setminus E_1) \cap \Omega_1 = \emptyset$)

$$SP_1 = (\Sigma_1 = \{s\}, \Omega_1 = \{0, 1\colon \rightarrow s;\ i\colon s \rightarrow s;\ *\colon s \times s \rightarrow s\}, Ax_1, E_1 = \emptyset),$$

where

$$\begin{aligned} Ax_1 = \ \{ \ & 0 \approx 1 \rightarrow , \\ & \rightarrow 0 * x \approx 0, \\ & \rightarrow i(x) * x \approx 1\} \ . \end{aligned}$$

We extend the specification by

$$SP_2 = (\Sigma_1, \Omega_1, Ax_2 = \emptyset, E_2 = \{x * y \approx y * x\}),$$

i.e. add the commutativity of $*$. As there are no clauses in Ax_2, SP is saturated modulo $E = E_1 \cup E_2 = E_2$ and does not contain the empty clause. But the specification is inconsistent. If we want a notion of saturation which also covers this case, we have to reconsider inferences with clauses in Ax_1 (here, in particular superposition inferences of Ax_1-clauses with unification modulo $E = E_2$). This would destroy the modularity of our (completion) method. Therefore we require $ops(E_2 \setminus E_1) \cap \Omega_1 = \emptyset$.

The effect of requiring $ops(E_2 \setminus E_1) \cap \Omega_1 = \emptyset$ is as follows:

Lemma 8.10 Let t and s denote two terms occurring in a clause in SP_1. The substitution σ is a simple E_1-unifier of t and s if and only if it is a simple E_2-unifier of t and s.

Proof: From $E_1 \subseteq E_2$ we conclude that E_1-unifiers are also E_2-unifiers. Now assume σ to be a simple substitution which is an E_2-unifier of t and s, i.e. $t\sigma =_{E_2} s\sigma$. Both terms (t and s) do not contain operator symbols of $\Omega_2 \setminus \Omega_1$. The substitution is simple, so cannot introduce such operator symbols, hence $t\sigma$ and $s\sigma$ are Ω_1-terms. As equations of $E_2 \setminus E_1$ do not contain Ω_1-operators, we cannot apply those equation to show $t\sigma =_{E_2} s\sigma$, hence have already $t\sigma =_{E_1} s\sigma$. So σ is already an E_1-unifier. □

Orderings for Hierarchic Specifications:
Term orderings used for hierarchic specifications need the following property:

> Every ground base term is smaller than every term that contains a non-base operator symbol.

This contradicts the E_2-compatibility if there exists a term t containing at least one non-base operator symbol and t is E_2-equal to a term s without non-base operator symbols (above we require $t >_E s$, from compatibility and $t =_{E_2} s$ we conclude $s >_E t$, hence $t >_E t$ which contradicts the irreflexivity and well-foundedness of the ordering). E_1 equations are variable preserving ($E_1 \subseteq E_2$ and E_2 satisfies definition 4.1). So there cannot exist terms t and s as above with $t =_{E_1} s$ and no further restrictions for E_1 are needed. For the other part of E_2 (i.e. for $E_2 \setminus E_1$) we require this explicitly. Note that any pure AC-theory (without ACU-operators; with non-base ACU-operators f, we can find base terms which are ACU-equal to non-base terms, e.g. $f(baseterm, e) =_{ACU} baseterm$; such equalities may destroy the compatibility of an ordering for hierarchic specifications, see above), where non-base operators are AC-operators (otherwise we may not fulfill the restriction $ops(E_2 \setminus E_1) \cap \Omega_1 = \emptyset$), is an appropriate candidate for $E_2 \setminus E_1$.

For hierarchic specifications satisfying all our requirements it is possible to have E-compatible (well-founded) orderings, where every base term is smaller than any term containing a non-base operator.

For lexicographic and recursive path orderings every base term is smaller than any term containing a non-base operator, if we use a precedence, where every non-base operator is greater than every base operator.

Orderings based on polynomial interpretations ([Cherifa/Lescanne 87]), where operator symbols are interpreted as polynomials and the interpretation of a ground term is always a natural number can not guarantee the above property. That is a disadvantage, because polynomial orderings are often used as AC-compatible orderings (and $E = AC$ is the most popular example for working modulo E). But there is a different method to use polynomial interpretations to construct AC-compatible (and even total) orderings ([Narendran/ Rusinowitch 91]): here operator symbols are interpreted as variables and the interpretation of a ground term is polynomial over such variables (see section 9.2). This method is also appropriate for hierarchic specifications (see lemma 9.15).

By a last example, we will sketch a useful application of hierarchic specifications.

Example 8.11 Consider the extension of a boolean specification by arithmetics.

$$SP_{bool} = (\{bool\}, \{and, or, \ldots\}, \ldots, AC(\{and, or\}))$$

$$SP_{bool+others} = (\{nat, bool, \ldots\}, \{and, or, \ldots, +, *, >, \ldots\}, \ldots, AC(\{and, or, +, *\}))$$

We here can expect to use SP_{bool} in a modular way, e.g. the result of completing $SP_{bool+others}$ without considering inferences of clause in SP_{bool} will be inconsistent, if and only if it contains the empty clause. Note that the set of AC-operator symbols contains base and non-base operators, hence neither E_1 nor $E_2 \setminus E_1$ is empty.

8.3 Basic Paramodulation and Superposition

In [Bachmair *et al.* 92] a class of restrictions for calculi similar to our inference system is introduced. Inspired by the *basic* strategy for narrowing, in their calculus paramodulation inferences are forbidden at terms introduced by substitutions from previous inference steps. They represent the objects of their inference systems as pairs $C \cdot \sigma$, where C is a clause (the *skeleton*) and σ is a substitution. Such a pair is called a *closure*. A closure $C \cdot \sigma$ represents a clause $C\sigma$ and is called a *ground closure*, if $C\sigma$ is a ground clause. We identify a clause D with the closure $D \cdot id$, where id is the identity substitution. The closure $C \cdot \sigma\rho$ is called an *instance* of $C \cdot \sigma$ (by substitution ρ). The inferences are formulated with closures as premises and conclusion, e.g.

basic superposition left

$$\frac{(\Gamma_1 \to \Delta_1, s_1 \approx s_2) \cdot \rho \quad (\Gamma_2, t_1 \approx t_2 \to \Delta_2) \cdot \rho}{(\Gamma_1, \Gamma_2, t_1[u \leftarrow s_2] \approx t_2 \to \Delta_1, \Delta_2) \cdot \theta}$$

where

- θ is the composition of ρ and σ, where σ is a most general unifier of $s_1\rho$ and $(t_1/u)\rho$
- $(s_1 \approx s_2)\rho$ is strictly maximal in the first premise
- $(t_1 \approx t_2)\rho$ is maximal in the second premise
- t_1/u is not a variable.

Note that the unifier σ (and also the substitution ρ) does not influence the skeleton part of the conclusion. As we would need a non-variable occurrence of the skeleton part to get further superpositions upon the above conclusion, we exclude superpositions upon positions introduced via substitutions in previous inference steps. That is the essential idea of basic superposition.

As done for the non-basic inference rules, we can enhance the above inference rule to work modulo E by using E-unification and an E-compatible ordering for the maximality constraints (note that above we have used the notion of maximality of non-ground terms; similar we could use the existence of a ground substitution σ_1 making the considered terms or equations maximal after instantiation with σ_1; cf. the discussion of these alternatives in section 8.1). Similarly we can extend the other inference rules of [Bachmair *et al.* 92] to work modulo E. To get a refutation complete calculus, we again have to use either E-extended clauses or E-closure inferences. As we never use a superposition upon an instance $(\to e_1 \approx e_2)\sigma$ of an E-equation (except for $\sigma = id$) and the E-closure inference may be regarded as a superposition upon a non-variable subterm of an E-equation, the E-closure inferences are already basic superpositions, if we represent their conclusion as a pair of skeleton and substitution.

Using E-extended clauses, the only superposition we need are upon the root of a maximal side of an extended equation (corollary 6.50), hence we can import other subterms into the substitution part of an extended clause. So the AC-extended clause for

$$(\Gamma \to \Delta, \ell_1 + \ell_2 \approx r) \cdot \sigma$$

will be

$$(\Gamma \to \Delta, x + y \approx r + x) \cdot \{x \leftarrow \ell_1 + \ell_2\}\sigma.$$

For completeness proofs we reason on the level of ground clauses, ground closures and ground terms only. To avoid lifting problems, it is again essential to consider inferences with reduced (ground) closures only. Note that in every lemma of our completeness proof, where we use inferences between ground clauses, we always first show that there is no reducible variable occurring in one of the premises.

We conjecture that it is possible to extend the completeness proof in [Bachmair *et al.* 92] to work modulo E yielding a proof similar to the completeness proof for $\mathcal{M}_E$ or $\mathcal{M}_{Ext}$. The conjecture is based on the similarity between their construction of an interpretation and the interpretation for $\mathcal{M}_E$ and $\mathcal{M}_{Ext}$. In particular, considering reduced instances, the essential properties are:

- We consider reducibility w.r.t. a system of ground rewrite rules.
- Only the strictly maximal equation of a ground instance in the succedent can produce rules.

But for the case $E = AC$ we have to be careful in combining optimizations. If we combine AC-closed sets (see definition 6.62), the restriction of superposing upon maximal AC-superterms only (lemma 6.61) and basic superposition, we get an incomplete system.

Example 8.12 Consider an AC-operator $+$ and a singleton set of clauses $\{ \rightarrow x + 0 \approx x\}$ which is AC-closed. After adding a goal $0 + (0 + 0) \approx 0 \rightarrow$, the set remains AC-closed. There is only one basic superposition left inference, where we superpose upon a maximal AC-superterm:

$$\frac{(\rightarrow x + 0 \approx x) \cdot id \quad (0 + (0 + 0) \approx 0 \rightarrow) \cdot id}{(x \approx 0 \rightarrow) \cdot \{x \leftarrow 0 + 0\}}$$

We are unable to reduce $0 + 0$ to 0. So either we have to superpose upon non-maximal AC-superterms or we have to give the above clause a special treatment. We recommend the latter one, because the clause is a real exception. We cannot construct a similar example, if the context variable x occurs on the right directly below a $+$ sign. So for clauses like $\rightarrow s(y) + x \approx s(y + x)$ (which does not need an extended clause) and for all extended clauses, we do not need a special treatment. By this special treatment we mean to replace the above inference by a new one with conclusion $(x_1 + x_2 \approx 0 \rightarrow) \cdot \{x_1 \leftarrow 0, \ x_2 \leftarrow 0\}$.

Using subsumption in a calculus with basic superposition, one has to ask for the intersection of two substitution parts. Representing a closure $C \cdot \sigma$ (or parts of it) by $C\sigma$ and underlining the parts introduced by substitution σ, what is the intersection of $\underline{(a + b)} + c$ and $b + \underline{(a + c)}$? We conjecture that the intersection can contain $\underline{a}$ (and not only a) and that the intersection of $\underline{(a + c)} + b$ and $\underline{(a + b)} + c$ will contain $\underline{a + b}$. The use of flattened terms (which is always possible when superposing upon maximal AC-superterms) may help to define the notion of intersection for $E = AC$. For general E it will be difficult to give a definition.

We expect that our work and the method in [Bachmair *et al.* 92] can be combined. The combination will be straightforward except some nasty minor details as in the two paragraphs above. As the work on basic superposition is currently in progress, we regard it as a future work to fill the small gaps needed in the combination of the two extensions of a paramodulation calculus.

9 E-Compatible Orderings

An early branch of research to provide orderings for E-completion tries to avoid the use of E-compatible orderings. They only consider the E-termination of $\Longrightarrow_R$ (i.e. the termination of $\Longrightarrow_{R/E}$) for a given R (and the termination of $\Longrightarrow_R$ can be verified using an arbitrary reduction ordering). E.g. [Jouannaud/Munoz 84] gives a method to check the E-termination of a rewrite system R (similar to check the confluence by convergence of critical pairs). So it considers E-compatibility in an indirect way, i.e. regarding $\Longrightarrow_R$ as an ordering over terms, it gives a criterion to decide, whether $\Longrightarrow_R$ can be used to characterize an E-compatible ordering. But a method to construct such an E-compatible ordering is not presented in [Jouannaud/Munoz 84].

But the main notion of [Jouannaud/Munoz 84], the *E-commutation*, was later applied to orderings $>$ not given by a rewrite relation ([Bachmair/Dershowitz 86], [Porat/Francez 86]). This way, E-commutation can be used to prove that a certain method to define an ordering, yields orderings useful for E-completion.

Definition 9.1 (E-Commutation)
A relation $>$ is *E-commuting*, if for every terms t, t' and s (with $t =_E t'$) with $t > s$ there exists an s' (with $s' =_E s$) such that $t' > s'$.

If $>$ is E-commuting, we can guarantee that there is no infinite chain $t_1 > t_2 =_E t_2' > t_3 =_E t_3' > \ldots$ ([Jouannaud/Munoz 84]). $>$ is not E-compatible in general, but very close to it:

Lemma 9.2 Let $>$ be an E-commuting and well-founded ordering. Then there are no terms t, t', s and s' such that $t > s$ but $s' > t'$. So $t > s$ implies that either $t' > s'$ or these terms are incomparable.

Proof: Assume $s > t =_E t_1 > s_1 =_E s$. By the E-commuting property for $s > t$, there is a term t_2 (with $t_2 =_E t$) and $s_1 > t_2$. But then using E-commutation for $t_1 > s_1$, we get a term s_2 with $s_2 =_E s_1 =_E s$ and $t_2 > s_2$. So we can construct an infinite chain $t_1 > s_1 > t_2 > s_2 > \ldots$, which contradicts the well-foundedness. □

So using E-commutation instead of E-compatibility we gain only the incomparability of some terms. In particular, for total orderings the difference between E-compatibility and E-commutation disappears. But even for non-total orderings we can construct an E-compatible ordering $>_E$ based on an E-commuting and well-founded ordering $>$:

$$t >_E s \qquad \text{if and only if}$$

$$t \neq_E s \text{ and there exist terms } t' \text{ and } s' \text{ with } t' > s'$$

As we cannot have $t > s > t'$ (see the lemma above), the ordering is irreflexive. Similar $>_E$ inherits other properties from $>$. So work done on orderings using E-commutation is useful to construct E-compatible orderings.

Nearly all papers constructing E-compatible (or E-commuting) orderings, which give an example for E, use $E = AC$. Often papers are even specialized to the case $E = AC$ (we will cite some in the next section). So the only theory E for which there are really useful results is $E = AC$.

9.1 AC-Compatible Orderings

From methods to construct simplification orderings, methods have been derived to get AC-compatible (or at least AC-commuting) orderings. This holds for

- the Knuth-Bendix ordering ([Steinbach 89b])
- recursive path orderings ([Bachmair/Plaisted 85], [Gnaedig/Lescanne 86])
- and other path and decomposition orderings ([Steinbach 89a]).

All these orderings are based on a precedence relation $>_{prec}$ on the operator symbols and use *flattening* of terms (e.g. a term $(a + (b + c)) + d$ is flattened to $+(a, b, c, d)$). Because of flattening, we have to have minimal precedence for AC-operator symbols to yield a monotonous ordering (monotonicity here means: $t > s$ implies $f(\ldots t \ldots) > f(\ldots s \ldots)$). With the help of distributivity transformations, we can relax this restriction to have two comparable AC-operator symbols. Proofs about these orderings often use AC-commuting properties. A slightly different approach can be found in [Gnaedig 87], here commuting is replaced by "cooperation" and flattening by "decanting" (an operation similar to flattening but preserving the original height of the terms, e.g. $(a + (b + c)) + d$ is replaced by $+(+(+(a, b, c, d)))$). This paper also proves that $E = AC$ is the maximal theory, where flattening can be used to construct E-compatible orderings.

All above mentioned techniques share the same disadvantage, they are limited to two comparable AC-operator symbols. This is a severe restriction, in particular, if we consider total orderings based on total precedence relations (so there can be at most two AC-operators in our signature). An AC-compatible ordering based on a modification of the lexicographic path ordering without requiring minimal precedence of AC-operators is recently developed in [Bachmair 91]. But as mentioned with an example in the conclusion of [Bachmair 91], this method does not provide orderings total on ground terms.

[Cherifa/Lescanne 87] gives another approach to AC-compatible orderings using polynomial interpretations. Besides the disadvantage that in general polynomial interpretations for the operator symbols of the considered signature are more difficult and less intuitive to find than the precedence relation $>_{prec}$, they are in general not total on ground terms (more precise: total on ground term AC-congruence classes) and not useful in hierarchic specifications (a lemma similar to 9.15 does not hold).

But AC-compatible orderings $>_{pol}$ based on polynomial interpretations have one advantage: they are completable, i.e. there exists a total (here total means that we always can compare two non-AC-equal ground terms) and AC-compatible ordering $>^t_{AC}$ such that $>_{pol} \subseteq >^t_{AC}$:

Lemma 9.3 Let $>_{AC}$ be an AC-compatible ordering total on non AC-equal ground terms. Let $\geq$ be an AC-compatible ordering and $>$ its strict (and partial) counterpart. Let $>_{lex}$ denote the following lexicographic combination of $>$ and $>_{AC}$:

$$t >_{lex} s \qquad \text{if and only if}$$

- $t > s$ or
- $t \geq s$, $s \geq t$ and $t >_{AC} s$.

The ordering $>_{lex}$ is AC-compatible. If the relation $\geq$ is total on ground terms, then $>_{lex}$ is total on non-AC-equal ground terms.

Proof: The AC-compatibility follows from the AC-compatibility of the combined orderings. Now we show the totality, so let t and s be ground terms with $t \neq_{AC} s$. If we have $t > s$ or $s > t$, then we are done. Otherwise, because of the totality of $\geq$, we have $t \geq s$ and $s \geq t$ and either $t >_{lex} s$ or $s >_{lex} t$ because of the totality of $>_{AC}$. □

Polynomial interpretations (in the sense of [Cherifa/Lescanne 87]) of ground terms are natural numbers. So $\geq_{pol}$ (for ground terms) is essentially the total ordering $\geq$ over natural numbers and so we can complete $>_{pol}$ by combination with an ordering $>_{AC}$ (such an ordering is described in the next section).

Things are different for recursive path orderings: extending the precedence of the operator symbols to a total relation yields a total ordering, but due to the severe restrictions on the precedence of AC-operator symbols, there may not be a total precedence yielding an AC-compatible ordering. So total precedences do not complete recursive path orderings for AC-completion. On the other hand, we cannot (in general) define a total (on ground terms) relation $\geq$ based on a (partial) recursive path ordering $>$. So in general recursive path orderings cannot be completed to total AC-compatible orderings.

But also for the above lexicographic combination with orderings like $>_{pol}$ we need a total and AC-compatible ordering $>_{AC}$, so we consider the most recent approach to AC-compatible orderings in greater detail:

9.2 Total AC-Compatible Orderings

Narendran and Rusinowitch presented a method to construct an AC-compatible and total ordering on ground terms ([Narendran/Rusinowitch 91]). For each operator h of the signature there is a variable X_h. The set of all such variables is called V_Σ. We interpret ground terms as polynomials (with non-negative coefficients) over this set of variables. So an interpretation I is a homomorphic mapping from ground terms to polynomials over V_Σ, i.e. an n-ary operator h is interpreted as an polynomial h_I with n variables and we have $I[h(t_1, \ldots, t_n)] = h_I(I[t_1], \ldots, I[t_n])$. Then we construct an ordering for terms from a total ordering on the set of variables. To avoid subscripts we ambiguously write h for both, the operator h and the variable X_h. We will improve the results in [Narendran/Rusinowitch 91] by giving an interpretation with smaller and less complicated polynomials and outline the extension of their method to compare non-ground terms.

Definition 9.4 An interpretation I is *AC-compatible*, if and only if

$$t =_{AC} s \qquad \text{implies} \qquad I[t] = I[s].$$

Lemma 9.5 An interpretation I (i.e. a mapping from ground terms to polynomials over V_Σ) is AC-compatible, if and only if it is of the following form (for AC-operators h):

$$I[h(t_1, t_2)] = F_1(h) \cdot I[t_1] \cdot I[t_2] + F_2(h) \cdot (I[t_1] + I[t_2]) + F_3(h)$$

where F_i is a polynomial over the variable h and F_1, F_2 and F_3 satisfy the following equation for all AC-operators h:

$$F_1(h) \cdot F_3(h) + F_2(h) = (F_2(h))^2$$

Proof: Let the homomorphic mapping I be defined as

$$I[h(t_1, \ldots, t_n)] = h_I(I[t_1], \ldots, I[t_n])$$

where h_I is a polynomial with arity n.

For AC-compatibility we require for every AC-operator h

$$I[h(t_1, t_2)] = I[h(t_2, t_1)]$$

for commutativity and (for associativity)

$$I[h(t_1, h(t_2, t_3))] = I[h(h(t_1, t_2), t_3)].$$

From the first equation we conclude that h_I has to be symmetric in its arguments. The second equation implies that h_I cannot contain an argument to a power greater than one: The equation is equivalent to

$$h_I(I[t_1], h_I(I[t_2], I[t_3])) = h_I(h_I(I[t_1], I[t_2]), I[t_3])$$

and if h_I raises its first argument to a power k, then in the above we have $(I[t_1])^k$ on the left, but $(I[t_1])^{2k}$ on the right (and the same holds for the second argument and $I[t_3]$).

So (because of symmetry and every exponent is not greater than one) it is of the above form:

$$I[h(t_1, t_2)] = F_1(h) \cdot I[t_1] \cdot I[t_2] + F_2(h) \cdot (I[t_1] + I[t_2]) + F_3(h)$$

The restriction for the polynomials F_i follows from the equation for associativity (using this representation of I) by a small computation. It is the same restriction (and computation) as for polynomial interpretations in [Cherifa/Lescanne 87] (they give an ordering for terms with variables and map those term variables to variables of polynomials; but the restrictions on the coefficients for the interpretation of AC-operators (which are in their case natural numbers and in our case polynomials over h) are the same). □

Example 9.6 In [Narendran/Rusinowitch 91] we find the following interpretation:

$$\begin{aligned} F_1(h) &= (h+1)(h^2+2h) \\ F_2(h) &= (h+1)^2 \\ F_3(h) &= (h+1) \end{aligned}$$

We can generalize this to

$$\begin{aligned} F_1(h) &= (h+1)^{(n-1)}[(h+1)^n - 1] \\ F_2(h) &= (h+1)^n \\ F_3(h) &= (h+1) \end{aligned}$$

for $n \geq 2$. The above instance with $n = 2$ is the only interpretation given in [Narendran/Rusinowitch 91], but it is easy to extend their methods to $n > 2$.

Note that the AC-compatibility of an interpretation depends only on the interpretation of the AC-operators, so can be seen as a property local to the interpretation of AC-operator symbols, whereas the following property is a global one, i.e. depends on the interpretation of all operator symbols.

Definition 9.7 An interpretation I is *root injective*, if and only if $I[t] = I[s]$ implies that neither $root(t)$ nor $root(s)$ is an AC-operator or else both are the same AC-operator ($root(t) = root(s)$).

Narendran and Rusinowitch use a stronger property than our root injectivity: $I[t] = I[s]$ implies $root(t) = root(s)$. We will show that our restriction is sufficient. The example 9.6 satisfies the stronger property.

Definition 9.8 An interpretation I is *occurrence preserving*, if and only if $I[t] = I[s]$ implies that every function symbol has the same number of occurrences in s and t.

Again we need only a weaker definition:

Definition 9.9 An interpretation I *has finite preimages*, if and only if every polynomial has only finitely many (if any) preimages, i.e. for every $p \in I[\mathcal{T}_\Sigma]$ there are at most finitely many terms $t_1, \ldots, t_n$ such that $I[t_i] = p$.

Lemma 9.10 If an interpretation I is occurrence preserving, then it has finite preimages.

Proof: Let p be a polynomial. Every term t with $I[t] = p$ is built of the same multiset of operator symbols. This multiset is finite, so there are only finitely many terms $t_1, \ldots, t_n$ with this multiset of operator symbols. The preimage of p under I is a subset of $\{t_1, \ldots, t_n\}$, hence finite. □

But there is a much weaker characterization for I having finite preimages:

Definition 9.11 Let I be an interpretation such that for every n-ary operator symbol h and every n-tuple $(t_1, \ldots, t_n)$ the interpretation $I[h(t_1, \ldots, t_n)]$ contains an occurrence of X_h (with exponent greater than 0), i.e. the interpretation does not forget about operator symbols occurring in the interpreted term. Then we call I an *admissible* interpretation.

Lemma 9.12 Every admissible interpretation I has finite preimages.

Proof: Let p be a polynomial and t be a term with $I[t] = p$. Let $k \cdot m$ ($k \in \mathbb{N}$) be a monomial, where the variable associated with the operator h occurs with exponent i. We then say the h-weight of $k \cdot m$ is $k \cdot i$. If we sum up the h-weights of each monomial in p, we get a finite number j. I is a homomorphic mapping and admissible, therefore there are at most j occurrences of h in t. By the same method, we get a finite bound on the number of occurrences for each of the finitely many operator symbols in t. We can define a finite maximal multiset M of operator symbols, such that each term s with $I[s] = p$ is built of operators occurring in a submultiset of M. So there can only be finitely many terms with the same interpretation. □

Definition 9.13 (Ordering over Polynomials)
A polynomial p is a finite multiset $\{mon_1, \ldots mon_n\}$ of monomials mon_i. A monomial mon consists of a factor k (a non-negative integer: $k \in \mathbb{N}$) and a variable part $f_1^{i_1} \cdot \ldots \cdot f_m^{i_m}$, where each exponent i_j is greater than 0. We assume a total ordering $>_{prec}$ on the set V_Σ of variables (as we identify names of variables and names of operator symbols from Σ, we also use the same name for the ordering on operator symbols and the ordering on the variables associated with operator symbols).

First we define an ordering $>_{mon}$ on monomials by

$$k_1 \cdot f_1^{i_1} \cdot \ldots \cdot f_m^{i_m} >_{mon} k_2 \cdot g_1^{j_1} \cdot \ldots \cdot g_n^{j_n} \qquad \text{if and only if}$$

- $n = m = 0$ *and* $k_1 > k_2$ *or*
- $f_1 >_{prec} g_1$ or
- $f_1 = g_1$ and $i_1 > j_1$ or
- $f_1 = g_1$, $i_1 = j_1$ and $k_1 \cdot f_2^{i_2} \cdot \ldots \cdot f_m^{i_m} >_{mon} k_2 \cdot g_2^{j_2} \cdot \ldots \cdot g_n^{j_n}$.

We here assume the f_i and g_j to be listed decreasingly w.r.t. $>_{prec}$. Now we define $>_I$ to be the multiset extension of $>_{mon}$ to finite multisets of monomials, so to polynomials.

Lemma 9.14 If $>_{prec}$ is total on V_Σ and well-founded, then $>_I$ is a total and well-founded ordering over polynomials.

For this ordering $>_I$ (and the orderings we derive from it, e.g. $>_{AC}$, definition 9.16) we can prove a property which is know from (lexicographic or recursive) path orderings:

Lemma 9.15 Let I be an admissible interpretation. Let t and s be ground terms. Let f be an operator occurring in t such that for every operator g occurring in s the variable X_f associated with f is greater than the variable X_g associated with g. Then we have $I[t] >_I I[s]$.

Proof: We consider only interpretations such that $I[t]$ contains at least one monomial, in which we find an occurrence of f (with exponent greater than 0). (Remember: for notational convenience we do not distinguish f and X_f; in the previous sentence X_f is meant). This monomial is greater than every monomial in $I[s]$, hence $I[t] >_I I[s]$. □

Assume there is a (total) precedence relation $>_{prec}$ on the operator symbols and we use this precedence relation to compare the associated variables. Further assume that an ordering $>_{AC}$ is based on an admissible interpretation and such a precedence relation. If an operator occurring in a term t is greater than every operator in a term s, then we have $t >_{AC} s$.

We can use this property for hierarchic specifications: Assume the set of operators is divided into base operators and non-base operators. Assume that every non-base operator is greater than every base operator. A ground term containing a non-base operator is greater than any ground term built of base operators only.

Definition 9.16 Let I be an admissible, AC-compatible and root injective interpretation. Let $t = f(t_1, \ldots, t_n)$ and $s = g(s_1, \ldots, s_m)$. Then $t >_{AC} s$ if and only if

- $I[t] >_I I[s]$ or
- $I[t] = I[s]$ and
  - if f is an AC-operator then $subterms_{AC}(f, t) >_{AC(mult)} subterms_{AC}(f, s)$
  - if f is not an AC-operator then $subterms(t) >_{AC(lex)} subterms(s)$

where $>_{AC(mult)}$ (resp. $>_{AC(lex)}$) is the multiset (resp. lexicographic) extension of $>_{AC}$, $subterms(t)$ is the multiset of immediate (direct) subterms (sons) of the root of t and

$$subterms_{AC}(f, h(t_1, \ldots, t_n)) = \begin{cases} \bigcup_{i=1}^{i=n} subterms_{AC}(f, t_i) & \text{if } f = h \\ \{h(t_1, \ldots, t_n)\} & \text{otherwise} \end{cases}$$

Lemma 9.17 The relation $>_{AC}$ is

1. irreflexive and transitive
2. well-founded
3. monotonic
4. total on the set of AC-congruence classes (for all t, s we have either $t >_{AC} s$ or $s >_{AC} t$ or $t =_{AC} s$)
5. AC-compatible

Proof: The proof is very similar to the corresponding one in [Narendran/Rusinowitch 91], but using root injectivity instead of the stronger property $I[t] = I[s]$ *implies* $root(t) = root(s)$ and the fact that I is admissible instead of occurrence preserving.

1) Irreflexivity is trivialy obtained. Suppose now $t >_{AC} s >_{AC} u$. If at least one comparison of the interpretations yields a $>_I$, the transitivity of $>_{AC}$ follows from the transitivity of $>_I$. Now, if $I[t] = I[s] = I[u]$ then, due to the root injectivity of I, all three terms have the same AC-operator as root symbol or no terms has an AC-operator at the top. The transitivity can be concluded using induction on the size (= height) of terms.

2) Suppose there exists an infinite antichain $t_1 >_{AC} t_2 >_{AC} \ldots$. Since $>_I$ is well-founded, there exists a $k \in \mathbb{N}$ such that for all $j \geq k$, we have $I[t_j] = I[t_k]$. Because I has finite preimages (we require I to be admissible; use lemma 9.12 to show that I has finite preimages), there are only finitely many terms which have the same interpretation $I[t_k]$. So there are two indices i_1 and i_2 with $k \leq i_1 < i_2$ and $t_{i_1} = t_{i_2}$. By transitivity we have $t_{i_1} >_{AC} t_{i_2}$, which contradicts irreflexivity, hence there is no such chain and the ordering $>_{AC}$ is well-founded.

3) The interpretation I is monotonic in the following sense:
$I[t] >_I I[s]$ implies $I[f(\ldots t \ldots)] >_I I[f(\ldots s \ldots)]$ and $I[t] = I[s]$ implies $I[f(\ldots t \ldots)] = I[f(\ldots s \ldots)]$.
For $t >_{AC} s$ and $I[t] >_I I[s]$ we immediately get $f(\ldots t \ldots) >_{AC} f(\ldots s \ldots)$. For $t >_{AC} s$ and $I[t] = I[s]$ we conclude $f(\ldots t \ldots) >_{AC} f(\ldots s \ldots)$ from the comparison of their (AC-) subterms.

4) As the ordering is defined by recursively comparing certain subterms, the totality can be proved using induction on the size (=height) of terms, assuming that for terms of smaller size they are either comparable or AC-equal.

5) Follows from the AC-compatibility of I and with induction on the size (=height) of terms following the definition of $>_{AC}$. □

The example 9.6 gives an interpretation that is appropriate to define an ordering $>_{AC}$. There are inappropriate interpretations:

Example 9.18 Let I_2 be an interpretation with

$$\begin{aligned} F_1(h) &= (h+1) \\ F_2(h) &= (h+1) \\ F_3(h) &= h \end{aligned}$$

It is AC-compatible and occurrence preserving but not root injective: With simple computations we get:

$$I_2[f(t_1, t_2)] + 1 = (f+1)(I_2[t_1] + 1)(I_2[t_2] + 1)$$

Let f and g be two different AC-operators. So $t = f(g(a,a),a)$ and $s = g(f(a,a),a)$ (terms with different root symbols) have the same interpretation:

$$(f+1)(g+1)(I_2[a]+1)^3 - 1$$

In a well-founded ordering defined as above using I_2 these terms are not comparable (i.e. the ordering is not total): We have $t >_{AC} s$ only if the multiset $\{g(a,a),a\}$ is greater than $\{g(f(a,a),a)\}$. This leads to $g(a,a) >_{AC} g(f(a,a),a)$ and $a >_{AC} f(a,a)$ which violates the well-foundedness. For the inverse comparison we obtain the same situation with the roles of f and g exchanged.

Lemma 9.5 gives a restriction for the interpretation of AC-operator symbols. We are free to use very simple interpretations for other symbols:

Definition 9.19

$$I_3[f(t_1,\ldots,t_n)] = (f+1)\cdot I_3[t_1]\cdot\ldots\cdot I_3[t_n]$$

if f is not an AC-operator, otherwise (for AC-operators f)

$$I_3[f(t_1,t_2)] = F_1(f)\cdot I_3[t_1]\cdot I_3[t_2] + F_2(f)\cdot(I_3[t_1]+I_3[t_2]) + F_3(f)$$

with F_i as in example 9.6.

Lemma 9.20 The interpretation I_3 is root injective.

Proof: Assume $I_3[t] = I_3[s]$. The definition of root injectivity only requires to consider cases, where at least one of the root symbols is an AC-operator (if both root operators are not AC-operators, there are terms with different root symbol but same interpretation, so the stronger property in [Narendran/Rusinowitch 91] is not satisfied). So assume there are terms $t = f(t_1,t_2)$ and $s = g(t_1,\ldots,t_n)$, f is an AC-operator and g is different from f. From the fact that the interpretation of t has to be dividable by $(g+1)$, we can construct a contradiction as in [Narendran/Rusinowitch 91]. The only thing we need is that the interpretation of subterms with root operator g_i are dividable by (g_i+1), which is valid in our interpretation. Considering this fact, the case where both root symbols are AC-operators has exactly the same proof as in [Narendran/Rusinowitch 91]. □

Lemma 9.21 The interpretation I_3 is occurrence preserving.

Proof: The maximal monomial in $I_3[t]$ is $\prod_{f\in Ops} f^{exp(f)}$ where Ops is the multiset of symbols in t and $exp(f)$ is 3, if f is an AC-operator, otherwise 1. □

So with the help of interpretation I_3 we may define an ordering as in definition 9.16. There are even less complex interpretations, e.g.

$$I_4[f(t_1,\ldots,t_n)] = f\cdot I_4[t_1]\cdot\ldots\cdot I_4[t_n]$$

if f is not an AC-operator, otherwise (for AC-operators f)

$$I_4[f(t_1,t_2)] = f\cdot I_4[t_1]\cdot I_4[t_2] + (f+1)(I_4[t_1]+I_4[t_2]) + (f+1)$$

So for AC-operators we have an interpretation with

$$\begin{aligned} F_1(h) &= h \\ F_2(h) &= (h+1) \\ F_3(h) &= (h+1)\,. \end{aligned}$$

Another possible and even less complex interpretation is obtained with

$$\begin{aligned} F_1(h) &= h \\ F_2(h) &= 1 \\ F_3(h) &= 0\,. \end{aligned}$$

Both interpretations are AC-compatible and occurrence preserving. But it is still unknown, if they are appropriate to define an AC-compatible ordering, because the root injectivity of such interpretations is an open question.

If we really want to use such an AC-compatible and total ordering in a computer implementation, interpretations like I_4 are nevertheless useful: An ordering using I_4 which has the root injectivity build in its definition, e.g.

$$t = f(t_1, \ldots, t_n) >_{AC} s = g(s_1, \ldots, s_m) \qquad \text{if and only if}$$

- $I_4[t] >_I I_4[s]$ or
- $I_4[t] = I_4[s]$, $f = g$ or neither f nor g is an AC-operator and
  - if f is an AC-operator then $subterms_{AC}(f,t) >_{AC(mult)} subterms_{AC}(f,s)$
  - if f is not an AC-operator then $subterms(t) >_{AC(lex)} subterms(s)$

loses only the totality (all other properties of lemma 9.17 hold). It can be used in a lexicographic combination with a total AC-compatible (and inefficient to compute) ordering. (For the totality of the lexicographic combination we need the totality of $\geq_{AC}$ in the first component (cf. lemma 9.3), hence have to define $t \geq_{AC} s$ by $t >_{AC} s$ or $I[t] = I[s]$.)

The author has not found a counterexample for the root injectivity (and so for the totality) of the above interpretation I_4. So we conjecture that only in rare cases (perhaps never) the inefficient second component of the lexicographic combination of orderings has to be computed.

Comparing Terms with Variables

Now we outline an extension of the above method to compare terms with variables.

As above, we assign a variable X_f to each operator symbol f (and identify X_f with f). We treat (term) variables as additional operator symbols, so assign them a (polynomial) variable of a set of variables disjoint to V_Σ, say V_V. Now interpretations are polynomials over the set $V = V_V \cup V_\Sigma$ of variables. Note that ground terms are interpreted as polynomials over V_Σ. We extend a total ordering $>^t_{prec}$ on V_Σ to a partial ordering on V by

$$op_1 >_{prec} op_2 \qquad \text{if and only if} \qquad op_1 >^t_{prec} op_2.$$

So polynomial variables assigned to term variables (i.e. variables of V_V) are always incomparable to other variables of V. To compare polynomials containing variables of V_V, we

have to list the variable part of the monomial based on an order of variables in V, where variables in V_Σ are listed in an order embedded in $>_{prec}$ and variables in V_V are listed prior to variables of V_Σ in an arbitrary but fixed order, e.g. a monomial can be represented as $2xyfga$ (for $f >_{prec} g >_{prec} a$) but not as $2fxgya$. So we implicitly use a total precedence of all variables in V which extends $>_{prec}$ by an arbitrary precedence for variables in V_V such that each variable in V_V is greater than every variable in V_Σ.

Definition 9.22 (Ordering over Polynomials in V)
A polynomial p is a finite multiset $\{mon_1, \ldots mon_n\}$ of monomials mon_i. A monomial mon consists of a factor F (a non-negative integer: $F \in \mathbb{N}$), a V_V part $v_1^{i_1} \cdot \ldots \cdot v_m^{i_m}$ with $v_i \in V_V$ and a V_Σ part $f_1^{i_{m+1}} \cdot \ldots \cdot f_n^{i_{n+m}}$ with $f_j \in V_\Sigma$, where each exponent i_j is greater than 0. First we define an ordering $>_{mon}$ over monomials by

$$F_1 \cdot v_1^{i_1} \cdot \ldots \cdot v_m^{i_m} \cdot f_1^{i_{m+1}} \cdot \ldots \cdot f_n^{i_{m+n}} >_{mon} F_2 \cdot w_1^{j_1} \cdot \ldots \cdot w_k^{j_k} \cdot g_1^{j_{k+1}} \cdot \ldots \cdot g_l^{j_{k+l}}$$

if and only if

$$m = k,\ v_r = w_r\ (1 \leq r \leq k), i_r \geq j_r\ (1 \leq r \leq k)$$

and

- $n = m = 0$ and $F_1 > F_2$ or
- $f_1 >_{prec} g_1$ or
- $f_1 = g_1$ and $i_{m+1} > j_{k+1}$ or
- $f_1 = g_1$, $i_{m+1} = j_{k+1}$ and $F_1 \cdot f_2^{i_{m+2}} \cdot \ldots \cdot f_n^{i_{n+m}} >_{mon} F_2 \cdot g_2^{j_{k+2}} \cdot \ldots \cdot g_l^{j_{k+l}}$.

Now we define $>_I$ to be the multiset extension of $>_{mon}$ to finite multisets of monomials, so to polynomials.

Now we will define an interpretation I by mixing the interpretation techniques of [Cherifa/Lescanne 87] and [Narendran/Rusinowitch 91]. Using this interpretation with definition 9.16, we will obtain an ordering which is

- a stable ordering for terms with variables,
- AC-compatible,
- total on ground terms (more precise: total on ground term AC-congruence classes),
- and useful for hierarchic specifications (i.e. lemma 9.15 holds).

Definition 9.23 Let $Pol(p_1, \ldots, p_n)$ denote the set of polynomials over $p_1, \ldots, p_n$, where each p_i occurs at least once in each polynomial (with an exponent greater than 0). We define an interpretation scheme for interpretations I by:

- If h is not an AC-operator, then

$$I[h(t_1, \ldots, t_n)] = (h + 1) \cdot Pol(I[t_1], \ldots, I[t_n]).$$

So for constants h (and variables) the interpretation is always $(h + 1)$.

- For exactly one AC-operator g we have:

$$I[g(t_1, t_2)] = (g + 1) + I[t_1] + I[t_2]$$

- For every AC-operator f with $f \neq g$ we have:

$$I[f(t_1, t_2)] = F_1(f) \cdot I[t_1] \cdot I[t_2] + F_2(f) \cdot (I[t_1] + I[t_2]) + F_3(f)$$

with F_i as in example 9.6.

Besides treating variables as new constant symbols (and extending the total precedence $>^t_{prec}$ to a partial one) the above interpretation differs from interpretation I_3 in two ways:

- More general polynomials are allowed to interpret non-AC-operators. We mix here the interpretation techniques of [Cherifa/Lescanne 87] and [Narendran/Rusinowitch 91]. (We assume that even more polynomial over h and $I[t_1], \ldots, I[t_n]$ are appropriate; we will prove this only for the above one.) With the above interpretation, we can orient rules like $h(x) \Rightarrow m(x, x, x)$, i.e. where a variable occurs more often in the smaller term (use the interpretation $I[h(x)] = (h + 1) \cdot (I[x])^3$ and $I[m(x, y, z)] = (m + 1) \cdot I[x] \cdot I[y] \cdot I[z]$ together with the precedence $h >_{prec} m$).

- The interpretation of AC-operators is more determined. But at least one AC-operator symbol g can have a small interpretation. Such different interpretations are needed to orient equations between terms containing two AC-operators, e.g. distributivity in set theory ($x \cap (y \cup z) \Rightarrow (x \cap y) \cup (x \cap z)$). We hope that even more interpretations of AC-operators are leading to AC-compatible and root injective interpretations, then we can also handle specifications with more AC-operators and critical equations between different AC-operator pairs (e.g. distributivity for sets and for natural numbers).

Lemma 9.24 Any interpretation defined using the above interpretation scheme is admissible and AC-compatible.

Proof: By definition it is admissible. For any AC-operator f with $f \neq g$ use lemma 9.5 for the F_i defined by [Narendran/Rusinowitch 91] and stated in example 9.6. This yields the AC-compatibility for the AC-operator f. For the AC-operator g we use the same lemma for the following definition of F_i:

$$\begin{aligned} F_1(g) &= 0 \\ F_2(g) &= 1 \\ F_3(g) &= (g + 1) \end{aligned}$$

Note that we have

$$I[g(t_1, t_2)] = F_1(g) \cdot I[t_1] \cdot I[t_2] + F_2(g) \cdot (I[t_1] + I[t_2]) + F_3(g).$$

□

Lemma 9.25 Any interpretation I defined using the above interpretation scheme is root injective.

Proof: Let op be an operator symbol. By σ_{op} we denote the morphism from terms to terms replacing each symbol in V by op.

Let h be a non-AC-operator and f be an AC-operator different from the AC-operator g.

- Part 1: Let t_1 be a term with root symbol h, then $I[t_1]$ can be divided by $(h+1)$. Let t_2 be a term with root symbol f, then $I[t_2]$ can be divided by $(f+1)$. Both facts are obvious from the definition of I. Also $I[t_2]$ cannot be divided by $(g+1)$ or $(h+1)$ (see [Narendran/Rusinowitch 91], lemma 2).
- Part 2:
  Let $t_3 := g(t_4, t_5)$ be a term with root symbol g, then $I[t_3]$ cannot be divided by $(h+1)$ or $(f+1)$ (also not by $(g+1)$ but we do not need this fact): If $I[t_3]$ is dividable by $(h+1)$, the same holds for $I[g(\sigma_h(t_4), \sigma_h(t_5))]$. But the latter one can be written as $(g+1)+(h+1)*r$ (for an appropriate polynomial r), so can obviously not be divided by $(h+1)$, hence also $I[t_3]$ cannot be divided by $(h+1)$. The same arguments apply to $(f+1)$ instead of $(h+1)$.
- Part 3:
  We will prove that in any case, where two interpretations with different root symbols and at least one root symbol is an AC-operator have the same interpretation, we yield a contradiction. For $I[t_1] = I[t_2]$ observe that the first interpretation is dividable by $(h+1)$, the second one is not (part 1). For $I[t_1] = I[t_3]$ observe that the first interpretation is dividable by $(h+1)$ (part 1), the second one is not (part 2). For $I[t_2] = I[t_3]$ observe that the first interpretation is dividable by $(f+1)$ (part 1), the second one is not (part 2).

□

Using these two lemmata we can verify that an ordering based on I (definition 9.16) has indeed the promised properties (use lemma 9.17). To show stability use induction on the size of terms and the fact that $t =_{AC} s$ implies $t\sigma =_{AC} s\sigma$.

9.3 ACU-Compatible Orderings

Definition 9.26 (ACU-Operator)
An operator f is called an *ACU-operator (with unit e)* if the following equations hold:

$$\begin{array}{rcll} f(f(x,y),z) & = & f(x,f(y,z)) & \text{(associativity)} \\ f(x,y) & = & f(y,x) & \text{(commutativity)} \\ f(x,e) & = & x & \text{(unit } e\text{)} \end{array}$$

Examples for such operators are the arithmetic AC-operators $+$ (with unit 0) and $*$ (unit 1), the boolean operators *and* (unit *true*) and *or* (unit *false*) or the set operator *union* (unit *emptySet*).

Definition 9.27 (ACU-Theory)
If a set E of equations consists of equations used in the previous definition and every operator occurring in E-equations is an ACU-operator, we say E is an *ACU-theory.*

We may relax this to sets of equations, where there are some operators which are only AC-operators (without a unit) or even only commutative operators. In the following we assume a fixed ACU-theory and by AC we mean the subset of all associativity and commutativity axioms for all ACU-operators.

Definition 9.28 Let R_{ACU} be the rewrite system containing the rules $f(x,e) \Rightarrow x$ and $f(e,x) \Rightarrow x$ for every ACU-operator f (with unit e).

Lemma 9.29
$\Longrightarrow_{R_{ACU}/AC}$ is terminating. $\Longrightarrow_{R_{ACU}}$ is Church-Rosser modulo AC for $\equiv_{ACU}$. Let $NF(t)$ denote the normal form of t with respect to $\Longrightarrow_{R_{ACU}}$. So two terms t and s are ACU-equal, if and only if their normal forms $NF(t)$ and $NF(s)$ are AC-equal.

We define an ordering $>_{ACU}$ based on an AC-compatible ordering $>_{AC}$ (see previous section for AC-compatible orderings).

Definition 9.30 We define an ordering $>_{ACU}$ based on $>_{AC}$ by

$$t >_{ACU} s \qquad \text{if and only if} \qquad NF(t) >_{AC} NF(s)$$

Lemma 9.31 If $>_{AC}$ is a well-founded reduction ordering on the AC congruence classes (of ground terms), it satisfies $t >_{AC} s$ whenever s is embedded in t and each unit e in ACU is minimal in this ordering, then $>_{ACU}$ is an ACU-compatible reduction ordering on the ACU congruence classes (of ground terms). If the ordering $>_{AC}$ is total on the AC congruence classes, then $>_{ACU}$ is total on the ACU congruence classes.

Proof:

1. total: Two terms are ACU-equal if and only if their normal forms (w.r.t. R_{ACU}) are AC-equal (lemma 9.29). So $>_{ACU}$ is total because $>_{AC}$ has this property.
2. well-founded: $>_{ACU}$ is embedded in $>_{AC}$.
3. transitive: Because $>_{AC}$ is transitive.
4. monotonic: (Also called stability with contexts):
   We have to show $t >_{ACU} s$ implies $g(\ldots,t,\ldots) >_{ACU} g(\ldots,s,\ldots)$ for all operators g and all terms hidden by the $\ldots$ notation. We speak of a context $c = g(t_1,\ldots,[],\ldots,t_n)$ and replace the hole [] by t or s. If t is not a unit, we have $NF(c[t]) = NF(c)[NF(t)]$ (for units this is wrong: $NF(f(a,e)) = a \neq NF(f(a,[]))[NF(e)] = f(a,e)$). If s is a unit, then $NF(c[s])$ is embedded in $NF(c)[NF(s)]$ and we require the latter one to be greater in $>_{AC}$. From $t >_{AC} s$ we conclude that t is not a unit (minimality of units). Now the ordering $>_{ACU}$ inherits its monotonicity from $>_{AC}$:

$$NF(c[t]) = NF(c)[NF(t)] >_{AC} NF(c)[NF(s)] >_{AC} NF(c[s]).$$

□

An ordering $>_{AC}$ required in this lemma is constructed in the previous section. But to get a total ordering, the minimality of units restricts ourselves to theories with at most one unit. We need a concept of incomparability to increase the number of units. One direction is to introduce sorts (types) and use many sorted logic. But as units are not related to sorts but to operators (e.g. the natural numbers have two units, unit 0 for addition and 1 for multiplication) this is still not satisfying. Introducing incomparable terms via constructors might be a better concept.

Often completion systems work on non-ground clauses and use orderings over terms with variables. These orderings have to be stable under substitutions, i.e. $t >_{ACU} s$ implies

$t\sigma >_{ACU} s\sigma$. This leads to more restrictions on $>_{ACU}$, e.g. we cannot have a simplification ordering, because $x + s >_{ACU} s$ (for an ACU-operator $+$ with unit 0) would imply $x\sigma + s\sigma >_{ACU} s\sigma$, hence (for $x\sigma = 0$) we get $0 + s\sigma =_{ACU} s\sigma >_{ACU} s\sigma$, which contradicts the irreflexivity and well-foundedness of $>_{ACU}$. So we have to look for ACU-completion methods which avoid the use of ACU-compatible orderings in such cases. A trivial solution is of course to use AC-completion techniques and considering clauses like $\rightarrow x + 0 \approx 0$ as normal N-clauses, not as E-clauses. For AC-completion such unit equations can efficiently be handled as rules $x+0 \Rightarrow x$, i.e. we need not use such equations in a nondeterministic way, reading it from left to right and from right to left. The main advantage of ACU-completion is merely the reduced number of unifiers (in particular for terms with variables). That is different from associativity and commutativity: obviously commutativity cannot be used as a rule and an ordering compatible with commutativity C and well-founded cannot orient associativity A, because we yield an infinite decreasing chain

$$(x + y) + z \Longrightarrow_A x + (y + z) =_C (y + z) + x \Longrightarrow_A y + (z + x) =_C$$

$$(z + x) + y \Longrightarrow_A z + (x + y) =_C (x + y) + z \ldots$$

(and a similar chain for orienting associativity the other way). So AC-completion has a stronger relevance to efficiency of theorem proving than a further extension of it to ACU-completion.

But there are also ACU-completion methods (e.g. based on ACU-unification), which try to use ACU-compatible orderings on the ground level only (and using AC-compatible orderings for other purposes, e.g. for orientation of equations into rules). The ACU-compatible orderings are very similar to the one constructed above (cf. [Jouannaud/Marché 90]). On the non-ground level they consider rules $\ell \Rightarrow r$ reduced w.r.t. $\Longrightarrow_{R_{ACU}}$ (definition see above) and with $\ell >_{AC} r$ (not $\ell >_{ACU} r$). They apply only instances $(\ell \Rightarrow r)\sigma$ of these rules such that $x\sigma \neq_{ACU} 0$ for all variables $x \in vars(\ell)$ and x is an immediate AC-subterm of an ACU-operator $+$ with unit 0 (see [Baird/Peterson/Wilkerson 89]). Hence we have $NF(\ell\sigma) >_{AC} NF(r\sigma)$ for these substitutions σ. But there are still no examples, for which ACU-completion yields a remarkable gain of efficiency compared with AC-completion (see [Baird/Peterson/Wilkerson 89]).

9.4 Theories with Projections

Definition 9.32 (Projection)
An operator p satisfying the equation $p(p(x)) \approx p(x)$ is called a *projection operator.* A theory E containing at least one such operator is called a *theory with projections.*

For simplicity we here consider only theories E with exactly one projection operator p.

Definition 9.33 Let E be divided into two sets E' and E_p, where $E_p = \{p(p(x)) \approx p(x)\}$ and $>$ be a reduction ordering compatible with E'. Let R_p be the singleton rule set $\{p(p(x)) \Rightarrow p(x)\}$ and $NF(t)$ the normal form of the term t with respect to R_p (there is always exactly one unique normal form). For terms t and s we define

$$t >_p s \qquad \text{if and only if} \qquad NF(t) > NF(s)$$

Lemma 9.34 If $>$ is a reduction ordering total on the E'-congruence classes (of ground terms) satisfying

$$p(t) > s \quad \text{implies} \quad t > s \quad \text{for all terms } t \text{ with } root(t) \neq p$$

then $>_p$ is an E-compatible reduction ordering total on the E-congruence classes (of ground terms).

Proof:

1. total: Two terms are E-equal if and only if their normal forms (w.r.t. R_p) are E'-equal. So $>_p$ is total because $>$ has this property.
2. well-founded: $>_p$ is embedded in $>$.
3. transitive: Because $>$ is transitive.
4. monotonic: (Also called stability with contexts):
   We have to show $t >_p s$ implies $g(\ldots,t,\ldots) >_p g(\ldots,s,\ldots)$ for all operators g and all terms hidden by the $\ldots$ notation. We speak of a context $c = g(t_1,\ldots,[],\ldots,t_n)$ and replace the hole $[]$ by t or s. If c is not $p([])$ the ordering inherits its monotonicity from $>$, because $NF(c[t]) = NF(c)[NF(t)]$. Otherwise we assume $c = p([])$ and give a case analysis:
   (a) $root(t) = root(s) = p$:
   $NF(p(t)) = NF(t)$, $NF(p(s)) = NF(s)$, so $NF(t) > NF(s)$ implies $NF(p(t)) > NF(p(s))$ which means $p(t) >_p p(s)$.
   (b) $root(t) = p$, $root(s) \neq p$:
   $p(t') := NF(p(t)) = NF(t)$, $NF(p(s)) = p(NF(s))$, so $NF(t) > NF(s)$ implies $p(t') > NF(s)$. Assume $p(NF(s)) > p(t')$. Then with the property we require for $>$ we get $NF(s) > p(t')$ violating $p(t') > NF(s)$. So we cannot have $p(NF(s)) > p(t')$ and yield $p(t') > p(NF(s))$ (totality of $>$) which is (via definition) $p(t) >_p p(s)$.
   (c) $root(t) \neq p$, $root(s) = p$:
   $NF(p(t)) = p(NF(t)) \geq NF(t) > NF(s)$ implies $NF(p(t)) > NF(s) = NF(p(s))$.
   (d) $root(t) \neq p$, $root(s) \neq p$:
   $NF(p(t)) = p(NF(t))$, $NF(p(s)) = p(NF(s))$ and monotonicity of $>$ implies $p(NF(t)) > p(NF(s))$.

□

Example 9.35 Let $E = \{p(p(x)) \approx p(x)\}$ and $>$ be an recursive path ordering (with total precedence). If p is the minimal operator, we get an E-compatible ordering $>_p$. If we consider other E, containing only some ground instances of the above E, e.g. $E = \{p(p(a)) \approx p(a)\}$, then some more precedences become acceptable.

The construction of E-compatible orderings for similar theories E containing equations like $not(not(x)) \approx x$ can be done similar by orienting the equation to a rule and comparing normal forms. In this case we need the requirement $not(t) > s$ implies $t > not(s)$.

10 Related Work

10.1 Extended Paramodulation

For $E = AC$ it is possible to extend the notion of superposition (also called paramodulation) to get a complete inference system without the use of AC-extended clauses. In [Rusinowitch/Vigneron 91] a calculus is presented introducing the notion of ordered AC-paramodulation and ordered extended AC-paramodulation.

[Rusinowitch/Vigneron 91] admits arbitrary predicate symbols, so a general resolution inference is needed. We only need equality resolution, which may be regarded as resolution with the clause $\rightarrow x \approx x$. For completeness this clause has to be contained in specifications considered by Rusinowitch and Vigneron. To get a better comparison to our work, we will neglect the differences resulting from considering general resolution and discuss the calculus of [Rusinowitch/Vigneron 91] restricted to the predicate $\approx$.

As a second minor difference Rusinowitch and Vigneron use ordered factoring for positive and negative literals, whereas we use factoring only for positive equations (i.e. equations in the succedent). Therefore they only paramodulate into strictly maximal negative equations. We have superposition left inferences also into the antecedent of a clause, where there are two identical (or AC-equal) maximal equations in the antecedent.

Having no inferences similar to equality factoring or merging paramodulation in [Rusinowitch/Vigneron 91], results in a major difference: they have to paramodulate into the smaller side of an equation. As shown by an example from Bachmair and Ganzinger ([Bachmair/Ganzinger 90], [Bachmair/Ganzinger 91c]) paramodulation into the maximal side of a positive equation only is not sufficient, hence Rusinowitch and Vigneron have to paramodulate into non-maximal terms of positive equations. We conjecture that it is superfluous for negative equations. But nevertheless the calculus in [Rusinowitch/Vigneron 91] defines paramodulations into the smaller side of a negative equation, too. Considering only Horn clauses (or even equational specifications) we need no inferences like merging paramodulation or equality factoring, but for the system of Rusinowitch and Vigneron the paramodulation into smaller sides of equations (even in the antecedent) remains. This is a disadvantage of their calculus.

To treat AC-properties without extended clauses, they introduce two paramodulation inferences which are not only extended by using AC-unification instead of unification (this is the only difference to standard systems for their resolution and factoring inference rule), but also by simulating paramodulations with extended clauses.

First we discuss the following inference rule (the second paramodulation inference in their paper):

OAC-ext-para:

$$\frac{(s \approx t) \vee D_1 \quad (\ell \approx r) \vee D_2}{((f(t,x) \approx f(r,y)) \vee D_1 \vee D_2)\sigma}$$

$$\text{if} \begin{cases} root(s\sigma) = root(\ell\sigma) = f, f \text{ is an } AC\text{-operator} \\ f(s,x)\sigma =_{AC} f(\ell,y)\sigma, \text{ where } x \text{ and } y \text{ are new variables} \\ (f(s,x) \approx f(t,x))\sigma \text{ is strictly maximal in the instantiated first clause} \\ (f(\ell,y) \approx f(r,y))\sigma \text{ is strictly maximal in the instantiated second clause} \\ f(s,x)\sigma \text{ is strictly maximal in } (f(s,x) \approx f(t,x))\sigma \\ f(\ell,y)\sigma \text{ is strictly maximal in } (f(\ell,y) \approx f(r,y))\sigma \end{cases}$$

Whenever we superpose an extended equation of an extended clause upon an extended equation at the root of the maximal side of this equation of another extended clause, then they have an *OAC-ext-para* inference. Note that this is a superposition on the right, because we extend only positive equations, i.e. equations in the succedent. Note also that this is the only kind of inferences between two extended clauses we need.

Sometimes they get more inferences: They require strict maximality of the extended equations $(f(s,x) \approx f(t,x))\sigma$ and $(f(\ell,y) \approx f(r,y))\sigma$; we require this maximality, too. But sometimes we do not get such an inference, because we do not need the corresponding extended clause: for an equation $\ell \approx r$ in the succedent of a clause C to need an extension we require the maximality of the equation we want to extend ($\ell \approx r$; see definition of closedness under AC-extension, definition 6.56). There are situations, where $(\ell+x \approx r+x)\sigma$ is maximal in $C\sigma$, but $(\ell \approx r)\sigma$ is not and no extended clause is introduced. So we have criteria to avoid the use of extended clauses, but the *OAC-ext-para* rule does not know such criteria.

Rusinowitch and Vigneron present a second paramodulation inference rule (the first paramodulation inference rule in their paper):

OAC-para:

$$\frac{(s \approx t) \vee D_1 \quad L \vee D_2}{(L_{new} \vee D_1 \vee D_2)\sigma}$$

if
- $(L/p)\sigma =_{AC} s\sigma$ or $(L/p)\sigma =_{AC} f(s,x)\sigma$, where f is an AC-operator ($= root(s\sigma)$) and x is a new variable
- $(s \approx t)\sigma$ or $(f(s,x) \approx f(t,x))\sigma$ is strictly maximal in the instantiated first clause
- $L\sigma$ is strictly maximal in the instantiated second clause
- $s\sigma$ resp. $f(s,x)\sigma$ is strictly maximal in $(s \approx t)\sigma$ resp. $(f(s,x) \approx f(t,x))\sigma$
- $L_{new} = L[p \leftarrow t]$ or $L_{new} = L[p \leftarrow f(t,x)]$
- Moreover, if L is not a positive equational literal, p is a non-variable occurrence of L and if L is a positive equational literal $\ell \approx r$ then p is either a non-variable occurrence of ℓ or possibly ε if $root(s\sigma)$ is an AC-operator

This inference rule covers a lot of superposition inferences of our system:

- superposition (left or right) between two unextended clauses
- superposition (left or right) of an extended clause upon an unextended clause
- superposition right of an unextended clause upon an extended equation of an extended clause

Note that by corollary 6.50 and lemma 7.25 other inferences with extended clauses are not needed in our system. The above definition of *OAC-para* again admits some inferences we do not consider, i.e. paramodulation into the smaller side of an equation and paramodulation with $f(s,x)$ where we would not need such an extended clause.

At the whole their system admits more inferences than ours, in particular, when considering only Horn clauses or equations. They also do not introduce simplification or elimination methods and it is not clear how to understand (and implement) their calculus without knowing about AC-extensions and how to apply their results to theories E with $E \neq AC$. In this respect the techniques described in this thesis appear to be superior.

10.2 Rewrite Methods for First-Order Theorem Proving

There are theorem proving methods based on a canonical rewrite system BR (with unconditional rules) for the boolean ring ([Bachmair/Dershowitz 87b], [Hsiang/Dershowitz 83], [Hsiang 85], [Hsiang 87]). Arbitrary Boolean terms are reduced to terms containing only the operators *and* (represented by juxtaposition) and *exclusive_or* (represented by $+$). As these operators are both associative and commutative, we need rewriting with AC-matching for the rules in BR. The system BR applied with AC-matching is Church-Rosser modulo AC for the Boolean ring.

Specifications are sets of (unconditional) equations and rules over Boolean terms, hence the equality sign $=$ (in the above mentioned papers) corresponds to logical equivalence of Boolean terms. A clause $A \rightarrow B$ corresponds to the Boolean term $AB + A + 1$ and to the boolean equation $AB + A = 0$ (or to the rule $AB + A \Rightarrow 0$). If one is accustomed to work with clauses, this representation is harder to read and the inferences on Boolean rules are more difficult to understand, e.g. a superposition (right) inference

$$\frac{A \rightarrow a \approx b \quad B \rightarrow a \approx c}{A, B \rightarrow b \approx c}$$

corresponds to a para-superposition inference (in the EN-strategy of [Hsiang 87])

$$\frac{A(a \approx b) + A \Rightarrow 0 \quad B(a \approx c) + B \Rightarrow 0}{(B(b \approx c) + B)A \Rightarrow 0}.$$

We think the former representation is more natural and more intuitive covering the idea of computing with the equations $a \approx b$ and $b \approx c$. Moreover clauses are often represented by terms containing more literals (e.g. $A, B \rightarrow C, D$ becomes $ABCD+ABC+ABD+AB \Rightarrow 0$). Especially the representation of implication (which is often used in specifications, e.g. for logic programming) is too sophisticated.

Similar to resolution, the rewrite based proof methods have to be enhanced to handle the equality predicate $\approx$ efficiently (do not confuse the equality predicate $\approx$ between arbitrary terms with $=$, the equivalence of Boolean terms).

[Bachmair/Dershowitz 87b] only covers the case, where $\approx$ is given by a canonical set of unconditional rewrite rules.

[Hsiang 87] extends his method, called the *N-strategy*, with a special treatment of $\approx$ (in particular his para-superposition inference incorporates paramodulation). The extended method is called EN-strategy.

Hsiang uses an ordering to decrease the number of possible inferences only in one situation: when superposing $s \approx t$ upon another (Boolean) term B with substitution σ, he requires $t\sigma \not\succeq s\sigma$. ([Bachmair/Dershowitz 87b] uses an ordering similarly as done in our calculus, but they do not extend their method to handle arbitrary $\approx$-literals specially.) Similar to the method in the previous section, Hsiang even can (and sometimes has to) paramodulate into the smaller term of an equation.

The rewrite based methods use rewriting also for simplification purposes, but a general criteria for simplification, such as our notion of compositeness, is not presented.

Saturated (w.r.t. our inference system) sets for consistent specifications can often be used to yield more efficient theorem proving methods (cf. [Bachmair/Ganzinger 91c]). Saturation w.r.t. the EN-strategy is only applied in refutational theorem proving contexts.

The rewrite based methods, if at all extended to handle $\approx$, have not been improved to work modulo AC (or modulo a general E). The only implicit use of AC-properties is restricted to the built-in treatment of the Boolean AC-operators *and* and *exclusive_or*.

10.3 Resolution Based Systems

Resolution (together with factoring) is a complete method to reason with clausal specifications. But to express equality relations, we have to add some nasty axioms like symmetry and transitivity of equality. Because resolution in specifications containing these axioms is far too inefficient, paramodulation is preferred. E.g. our inference system especially developed for equality literals is based on paramodulation and avoids the application of congruence axioms for equality. [Stickel 85] introduces *theory resolution*, which extends resolution in a very general way, e.g. paramodulation is covered as *partial theory resolution*. Just as well resolution with E-unification is an example of theory resolution for the theory E. But combining these two branches (paramodulation considered as partial theory resolution and resolution with E-unification) is incomplete in general ([Eisinger/Ohlbach 91]). To get a complete calculus, we have to separate the equality axioms from the other clauses such that the equality symbol does not occur in clauses not contained in E. Results published in the framework of theory resolution combining these two branches are not known. Our calculus combines paramodulation and working modulo E. Since the only predicate symbol is $\approx$, we do not really consider resolution.

Hölldobler ([Hölldobler 88], [Hölldobler 89]) also extends resolution to resolution modulo E, but suffers from the same drawback: he cannot arbitrarily mix equality literals and non-equality literals in the same clause. Besides that, he only considers resolution for logical programming, i.e. restricted to Horn clauses.

11 Conclusions

We have presented calculi for theorem proving modulo E. Inference rules are defined for full first-order clauses, hence theorem proving with Horn clauses and unconditional equational specifications is covered as a special case. The calculi are extensions and modifications of inference systems for $E = \emptyset$. We summarize the main influences of E, i.e. the main differences to methods for $E = \emptyset$:

- We use E-unification.
- Orderings are E-compatible.
- There are E-specific inferences (E-closure inferences) or E-specific clauses (E-extended clauses).

The main part of our work is the completeness proof for the calculi $\mathcal{M}_E$ and $\mathcal{M}_{Ext}$. For an $\mathcal{M}_E$-saturated (resp. $\mathcal{M}_{Ext}$-saturated and closed) set N of clauses not containing the empty clause, we define a model satisfying all clauses in N and all equations of E via an equality Herbrand interpretation. The interpretation consists of a ground rewrite system R_{TOP} and a set E_{TOP} of ground instances of E-equations such that the rewrite relation $\Longrightarrow_{R_{TOP}}$ is a Church-Rosser system modulo E_{TOP}. Other methods for proving completeness results have already been adopted to methods modulo E, e.g. [Bachmair 88] for working with proof orderings or [Rusinowitch/Vigneron 91] for semantic trees and $E = AC$. The method of constructing models via Church-Rosser rewrite systems has not been extended to problems modulo E in the literature so far. The reader interested in this method should first study the interpretation and proofs for $\mathcal{M}_E$, because the introduction of extended clauses complicates the definition of the interpretation and the completeness proof for $\mathcal{M}_{Ext}$. We outlined the use of these techniques to systems with merging paramodulation and to hierarchic specifications. We conjecture that a lot of other similar theorem proving methods can be proved to be complete with our interpretation method. But as for any other method of proving completeness the construction of interpretations becomes much more complicated and needs a lot of technical details, when we do the step from an empty set E to $E \neq \emptyset$.

All our results hold for the case $E = AC$. Some refinements for this special case have been discussed, e.g. reducing the number of AC-extended clauses and implementing rewriting modulo AC more efficiently using the relation $\Longrightarrow_{R^{AC}}$. Without such refinements (e.g. providing the finiteness of sets closed under AC-extension) AC theorem proving remains intractable.

We have shown, how our inference systems can be incorporated into completion algorithms. We presented the notion of "compositeness" providing a redundancy criterion appropriate for completion systems working modulo E. Compositeness subsumes most elimination and simplification techniques known for completion and theorem proving.

In the last section we discussed E-compatible orderings. The only known approach so far to obtain total and AC-compatible reduction orderings ([Narendran/Rusinowitch 91]) is improved here and investigated in greater detail.

In summary, this paper covers all ingredients for the design and implementation of a first-order theorem prover modulo AC. With this paper we provide the basis for the implementation of such a prover in terms of a reliable theoretical background.

References

[Anantharaman/Hsiang 89]
Siva Anantharaman and Jieh Hsiang. Automated Proofs of the Moufang Identities in Alternative Rings. LIFO, Université d'Orléans, Orléans (France), 1989.

[Anantharaman/Hsiang/Mzali 89]
Siva Anantharaman, Jieh Hsiang, and Jalel Mzali. SbReve2: A Term Rewriting Laboratory with (*AC*-) Unfailing Completion. In *Proceedings 3rd Conference on Rewriting Techniques and Applications*, pp. 533–537, Chapel Hill, April 1989. Springer Verlag, LNCS 355.

[Anantharaman/Mzali 89]
Siva Anantharaman and Jalel Mzali. Unfailing Completion Modulo a Set of Equations. Research Report No. 470, LRI-Orsay, Orsay (France), 1989.

[Bachmair 87]
Leo Bachmair. *Proof methods for equational theories.* PhD thesis, University of Illinois at Urbana-Champaign, 1987.

[Bachmair 88]
Leo Bachmair. Proof methods for equational theories. Extended revision of the PhD thesis, SUNY at Stony Brook, New York, 1988.

[Bachmair 91]
Leo Bachmair. Associative-Commutative Reduction Orderings. Report MPI-I-91-209, Max-Planck-Institut für Informatik, Saarbrücken (Germany), 1991.

[Bachmair *et al.* 92]
Leo Bachmair, Harald Ganzinger, Christopher Lynch, and Wayne Snyder. Basic Paramodulation and Superposition. 11th Conference on Automated Deduction; *to appear*, 1992.

[Bachmair/Dershowitz 86]
Leo Bachmair and Nachum Dershowitz. Commutation, Transformation and Termination. In *Proceedings of the 8th Conference on Automated Deduction (CADE).* Springer Verlag, LNCS 230, 1986.

[Bachmair/Dershowitz 87a]
Leo Bachmair and Nachum Dershowitz. Completion for rewriting modulo a congruence. In *Proceedings 2nd Conference on Rewriting Techniques and Applications*, pp. 192–203, Bordeaux (France), May 1987. Springer Verlag, LNCS 256.

[Bachmair/Dershowitz 87b]
Leo Bachmair and Nachum Dershowitz. Inference Rules for Rewrite-Based First-Order Theorem Proving. In *Proceedings of the Symposium on Logic in Computer Science (LICS)*, pp. 331–337, Ithaca, New York, June 1987. IEEE.

[Bachmair/Dershowitz 88]
Leo Bachmair and Nachum Dershowitz. Critical Pair Criteria for Completion. *Journal of Symbolic Computation*, Vol. 6, pp. 1–18, 1988.

[Bachmair/Dershowitz/Hsiang 86]
L. Bachmair, N. Dershowitz, and J. Hsiang. Orderings for equational proofs. In *Proc. Symp. Logic in Computer Science*, pp. 346–357, Boston, Massachusetts, 1986.

[Bachmair/Ganzinger 90]
Leo Bachmair and Harald Ganzinger. On Restrictions of Ordered Paramodulation with Simplification. In Mark E. Stickel, editor, *Proceedings 10th International Conference on Automated Deduction (CADE)*, pp. 427–441, Kaiserlautern (Germany), July 1990. Springer Verlag, Lecture Notes in Artificial Intelligence 449.

[Bachmair/Ganzinger 91a]
Leo Bachmair and Harald Ganzinger. Completion of First-Order Clauses with Equality by Strict Superposition. In *Proceedings of the 2nd International Workshop on Conditional and Typed Rewriting Systems*, pp. 162–180, Montreal (Canada), 1991. Springer Verlag, LNCS 516.

[Bachmair/Ganzinger 91b]
Leo Bachmair and Harald Ganzinger. Perfect Model Semantics for Logic Programs with Equality. Proceedings of the International Conference on Logic Programming (ICLP 91), 1991.

[Bachmair/Ganzinger 91c]
Leo Bachmair and Harald Ganzinger. Rewrite-Based Equational Theorem Proving With Selection and Simplification. Report 208, Max-Planck-Institut für Informatik, Saarbrücken (Germany), to appear also in *Journal of Computation and Logic*, 1991.

[Bachmair/Ganzinger/Waldmann ed]
Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. First-Order theorem proving for hierarchic specifications. Technical Report, Max-Planck-Institut für Informatik, Saarbrücken (Germany), 1992, submitted.

[Bachmair/Plaisted 85]
Leo Bachmair and David A. Plaisted. Associative Path Orderings. In *Proceedings of the 1st Conference on Rewriting Techniques and Applications (RTA)*, pp. 241–254. Springer Verlag, LNCS 202, 1985.

[Baird/Peterson/Wilkerson 89]
Timothy B. Baird, Gerald E. Peterson, and Ralp W. Wilkerson. Complete Sets of Reductions Modulo Associativity, Commutativity and Identity. In Nachum Dershowitz, editor, *Proceedings 3rd Conference on Rewriting Techniques and Applications*, pp. 29–44, Chapel Hill, April 1989. Springer Verlag, LNCS 355.

[Bertling 90]
Hubert Bertling. *Knuth-Bendix Completion of Horn Clause Programs for Restricted Linear Resolution and Paramodulation.* PhD thesis, Universität Dortmund (Germany), 1990.

[Bertling/Ganzinger/Schäfers 89]
Hubert Bertling, Harald Ganzinger, and Renate Schäfers. CEC: A System for Conditional Equational Completion – User Manual (Version 1.5). Report M.1.3-R-18.0, PROSPECTRA-Project, Universität Dortmund, West Germany, 1989.

[Bockmayr 90]
Alexander Bockmayr. *Beiträge zur Theorie des logisch-funktionalen Programmierens.* PhD thesis, Univ. Karlsruhe, 1990.

[Chang/Lee 73]
Chin-Liang Chang and Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, 1973.

[Cherifa/Lescanne 87]
B.A. Cherifa and Pierre Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computer Programming*, Vol. 9, No. 2, pp. 137–160, October 1987.

[Dershowitz 82]
Nachum Dershowitz. Orderings for Term Rewrite Systems. *Theoretical Computer Science*, Vol. 17, pp. 279–301, 1982.

[Dershowitz 85]
Nachum Dershowitz. Computing with Rewrite Systems. *Information and Control*, Vol. 65, pp. 122–157, 1985.

[Dershowitz 87]
Nachum Dershowitz. Termination of Rewriting. *Journal of Symbolic Computation*, Vol. 3, No. 1&2, pp. 69–115, 1987.

[Dershowitz 89]
Nachum Dershowitz. Completion and Its Applications. In Hassan Ait-Kaci and Maurice Nivat, editors, *Resolution of Equations in Algebraic Structures, Volume 2: Rewriting Techniques*, pp. 31–85. Academic Press, 1989.

[Dershowitz/Jounannaud 89]
Nachum Dershowitz and Jean-Pierre Jounannaud. Rewrite Systems. Rapports de Recherche No. 478, Laboratoire de Recherche en Informatique (LRI), Orsay (France), 1989.

[Drosten 89]
Klaus Drosten. *Termersetzungssysteme.* Springer Verlag, 1989.

[Eisinger/Ohlbach 91]
Norbert Eisinger and Hans Jürgen Ohlbach. Deduction Systems Based on Resolution. Report MPI-I-91-217, Max-Planck-Institut für Informatik, Saarbrücken (Germany), 1991.

[Fages 87]
F. Fages. Associative commutative unification. *Journal of Symbolic Computation*, Vol. 3, pp. 257–275, 1987.

[Fortenbacher 88]
Albrecht Fortenbacher. *Effizientes Rechnen in AC-Gleichungstheorien.* PhD thesis, Universität Karlsruhe, 1988.

[Gallier 87]
Jean H. Gallier. *Logic for Computer Science: Foundations of Automatic Theorem Proving.* John Willey & Sons, 1987.

[Ganzinger 88a]
Harald Ganzinger. A completion procedure for conditional equations. In Kaplan and Jouannaud, editors, *Proc. 1st Int. Workshop on Conditional Term Rewriting Systems, June 1987*, pp. 62–83. Springer Verlag, LNCS 308, 1988.

[Ganzinger 88b]
Harald Ganzinger. Completion with history-dependent complexities for generated equations. In Sannella and Tarlecki, editors, *Proc. Workshop on Abstract Data Types, 1987*, pp. 73–91. Springer Verlag, LNCS 332, 1988.

[Ganzinger 91]
Harald Ganzinger. A completion procedure for conditional equations. *Journal of Symbolic Computation*, Vol. 11, pp. 51–81, 1991.

[Ganzinger/Schäfers 90]
Harald Ganzinger and Renate Schäfers. System Support for Moduar Order-Sorted Horn Clause Specifications. In *Proceedings 12th International Conference on Software Engeneering*, pp. 150–159, Nice (France), March 1990. IEEE.

[Gnaedig 87]
Isabelle Gnaedig. Investigations on Termination of Equational Rewriting. Rapports de Recherche No. 732, Institute National de Recherche en Informatique et en Automatique (INRIA), France, 1987.

[Gnaedig/Lescanne 86]
Isabelle Gnaedig and Pierre Lescanne. Proving Termintion of Associative Commutative Rewriting Systems by Rewriting. In *Proceedings of the 8th Conference on Automated Deduction (CADE)*, pp. 52–61. Springer Verlag, LNCS 230, 1986.

[Hofbauer/Kutsche 89]
Dieter Hofbauer and Ralf-Detlef Kutsche. *Grundlagen des maschinellen Beweisens.* Vieweg, 1989.

[Hölldobler 88]
Steffen Hölldobler. SLDE-Resolution. Report of "Universität der Bundeswehr", Munich, 1988.

[Hölldobler 89]
Steffen Hölldobler. Equational Logic Programming. Springer Verlag, Lecture Notes in Artificial Intelligence 353, 1989.

[Hsiang 85]
Jieh Hsiang. Refutational Theorem Proving using Term-Rewrite Systems. *Artificial Intelligence*, Vol. 25, pp. 255–300, 1985.

[Hsiang 87]
Jieh Hsiang. Rewrite Methods for Theorem Proving in First Order Theories with Equality. *Journal of Symbolic Computation*, Vol. 3, pp. 133–151, 1987.

[Hsiang/Dershowitz 83]
Jieh Hsiang and Nachum Dershowitz. Rewrite Methods for Clausal and Non-Clausal Theorem Proving. In *Proceedings 10th International Conference on Automata, Languages and Programming (ICALP)*, Barcelona (Spain), 1983. Springer Verlag, LNCS 154.

[Huet 80]
G. Huet. Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems. *Journal of the ACM*, Vol. 37, No. 4, pp. 797–821, October 1980.

[Huet 81]
G. Huet. A Complete Proof of the Correctness of the Knuth and Bendix Completion Algorithm. *Journal of Computation and System Science*, Vol. 23, pp. 11–21, 1981.

[Jouannaud/Kirchner 86]
Jean-Pierre Jouannaud and Helene Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal of Computing*, Vol. 15, No. 4, pp. 1155–1194, 1986.

[Jouannaud/Marché 90]
Jean-Pierre Jouannaud and Claude Marché. Completion modulo Associativity, Commutativity and Identity (AC1). In A. Miola, editor, *Proceedings of the International Symposium on Design and Implementation of Symbolic Computation Systems (DISCO)*, Capri (Italy), 1990. Springer Verlag, LNCS 429.

[Jouannaud/Munoz 84]
Jean-Pierre Jouannaud and Miguel Munoz. Terminaton of a set of rules modulo a set of equations. In *Proceedings of the 7th Conference on Automated Deduction (CADE)*, pp. 175–191. Springer Verlag, LNCS 170, 1984.

[Jouannaud/Waldmann 86]
Jean-Pierre Jouannaud and Bernard Waldmann. Reductive conditional term rewriting systems. In *Proc. 3rd TC2 Working Conference on the Formal Description of Prog. Concepts*. North-Holland, August 1986.

[Kaplan 84]
Stephane Kaplan. Conditional rewrite rules. *Theoretical Computer Science*, Vol. 33, pp. 175–193, 1984.

[Kaplan 87]
Stephane Kaplan. A compiler for conditional term rewriting. In *Proceedings 2nd Conference on Rewriting Techniques and Applications*, pp. 25–41, Bordeaux (France), May 1987. Springer Verlag, LNCS 256.

[Kaplan/Remy 87]
Stephane Kaplan and J.L. Remy. Completion algorithms for conditional rewriting systems. In *MCC Workshop on Resolution of Equations in Algebraic Structures*, May 1987.

[Kapur/Musser/Narendran 88]
Deepak Kapur, David R. Musser, and Paliath Narendran. Only Prime Superpositions Need be Considered in the Knuth-Bendix Completion Procedure. *Journal of Symbolic Computation*, Vol. 6, pp. 19–36, 1988.

[Knuth/Bendix 70]
D. E. Knuth and P. B. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pp. 263–297. Pergamon Press, Oxford, 1970.

[Lai 90]
Mike Lai. Using a Term Rewriting Tool to Derive Commutativity for Rings Satisfying $x^5 = x$. In *2nd International Workshop on Conditional and Typed Rewriting Systems, Extended Abstracts*, Montreal (Canada), June 1990. CENPARMI Concordia University, Montréal.

[Loveland 78]
Donald W. Loveland. *Automated Theorem Proving: A Logical Basis*, volume 6 of *Fundamental Studies in Computer Science*. North-Holland, New York, 1978.

[Marché 91]
Claude Marché. On AC-termination and ground AC-completion. In Ronald V. Book, editor, *4th International Conference on Rewriting Techniques and Applications (RTA)*, pp. 411–422, Como (Italy), April 1991. Springer Verlag, LNCS 488.

[Martin/Nipkow 90]
Ursula Martin and Tobias Nipkow. Ordered Rewriting and Confluence. In Mark E. Stickel, editor, *Proceedings 10th International Conference on Automated Deduction (CADE)*, pp. 366–380, Kaiserlautern (Germany), July 1990. Springer Verlag, Lecture Notes in Artificial Intelligence 449.

[Narendran/Rusinowitch 91]
Paliath Narendran and Michaël Rusinowitch. Any ground associative-commutative theory has a finite canonical system. In Ronald V. Book, editor, *4th International Conference on Rewriting Techniques and Applications (RTA)*, pp. 423–434, Como (Italy), April 1991. Springer Verlag, LNCS 488.

[Nieuwenhuis 91]
Robert Nieuwenhuis. First-Order Completion Techniques. Research Report LSI-UPC, 1991.

[Peterson/Stickel 81]
Gerald E. Peterson and Mark E. Stickel. Complete sets of reductions for some equational theories. *Journal of the ACM*, Vol. 28, No. 2, pp. 233–264, April 1981.

[Porat/Francez 86]
Sara Porat and Nissim Francez. Full-Commutation and Fair-Termination in Equational (and Combined) Term-Rewriting Systems. In *Proceedings of the 8th Conference on Automated Deduction (CADE)*, pp. 21–41. Springer Verlag, LNCS 230, 1986.

[Rusinowitch 87]
Michaël Rusinowitch. Theorem Proving with Resolution and Superposition: An Extension of Knuth and Bendix Procedure as a Complete Set of Inference Rules. CRIN Report 87-R-128, Centre de Recherche en Informatique de Nancy, Vandoeuvre les Nancy (France), 1987.

[Rusinowitch/Vigneron 91]
M. Rusinowitch and L. Vigneron. Automated Deduction with Associative Commutative Operators, 1991.

[Steinbach 89a]
Joachim Steinbach. Path and Decomposition Orderings for Proving *AC*-Termination. SEKI Report SR-89-18, Universität Kaiserslautern, Kaiserslautern (Germany), 1989.

[Steinbach 89b]
Joachim Steinbach. Proving Termination of Associative-Commutative Rewrite Systems Using the Knuth-Bendix Ordering. SEKI Report SR-89-13, Universität Kaiserslautern, Kaiserslautern (Germany), 1989.

[Stickel 81]
Mark E. Stickel. A unification algorithm for associative commutative functions. *Journal of the ACM*, Vol. 28, No. 3, pp. 423–434, July 1981.

[Stickel 84]
Mark E. Stickel. A Case Study of Theorem Proving by the Knuth-Bendix Method: Discovering that $x^3 = x$ implies ring commutativity. In *Proceedings 7th International Conference on Automated Deduction (CADE)*, pp. 248–258. Springer Verlag, LNCS 170, 1984.

[Stickel 85]
Mark E. Stickel. Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*, Vol. 1, pp. 333–353, 1985.

[Stickel 86]
Mark E. Stickel. A Prolog Technology Theorem Prover: Implementation by an extended Prolog Compiler. In *Proceedings 8th International Conference on Automated Deduction (CADE)*, pp. 573–587. Springer Verlag, LNCS 230, 1986.

[Wertz 89]
Ulrich Wertz. Vervollständigung einer Gleichungsspezifikation modulo einer Gleichungskongruenz. Diploma thesis (in german), University of Dortmund, 1989.

[Zhang 88]
Hantao Zhang. *Reduction, Superposition and Induction: Automated Reasoning in an Equational Logic.* PhD thesis, University of Iowa, Technical Report 88-06, 1988.

[Zhang/Kapur 88]
Hantao Zhang and Deepak Kapur. First-Order Theorem Proving Using Conditional Rewrite Rules. In *Proceedings 9th International Conference on Automated*

Deduction (CADE), pp. 1–20, Argonne (USA), May 1988. Springer Verlag, LNCS 310.

[Zhang/Kapur 89]
Hantao Zhang and Deepak Kapur. Consider Only General Superpositions in Completion Procedures. In Nachum Dershowitz, editor, *Proceedings 3rd Conference on Rewriting Techniques and Applications*, pp. 513–527, Chapel Hill, April 1989. Springer Verlag, LNCS 355.

[Zhang/Kapur 90]
Hantao Zhang and Deepak Kapur. Unnecessary Inferences in Associative-Commutative Completion Procedures. *Mathematical Systems Theory*, Vol. 23, pp. 176–206, 1990.