J. Symbolic Computation (1991) **11**, 51–81

A Completion Procedure for Conditional Equations†

HARALD GANZINGER‡

Max-Planck-Institut für Informatik, Im Stadtwald, D-6600 Saarbrücken, Germany

(*Received* 15 *September* 1987)

The paper presents a new completion procedure for conditional equations. The work is based on the notion of reductive conditional rewriting and the procedure has been designed to handle in particular non-reductive equations that are generated during completion. The paper also describes techniques for simplification of conditional equations and rules, so that the procedure terminates on more specifications. The correctness proofs which form a substantial part of this paper employ recursive path orderings on the proof trees of conditional equational logic, an extension of the ideas of Bachmair, Dershowitz & Hsiang to the conditional case.

1. Introduction

In this paper we present a completion procedure for conditional equations. Emphasis is laid upon a rigorous proof of the correctness of the procedure as well as on developing techniques that make the procedure useful in practice. The completion procedure is, adopting the ideas of Bachmair, Dershowitz & Hsiang (1986) and Bachmair (1987) presented as a set of inference rules. It is shown that any application of an inference rule leads to less complex proofs of equational theorems. If the inference rules are applied according to a fair strategy—which, however, need not exist in all cases—the final system of conditional rules will be reductive, canonical, and will generate the same congruence as the initially given set of conditional equations.

Proofs will be represented as terms in a signature of proofs. A recursive path ordering on proof terms will allow to order proofs according to their complexity. The proof of correctness of the completion procedure is based on the idea that proofs for equational theorems which are not rewrite proofs can be transformed into simpler proofs at some stage of the completion process. As the ordering of proofs is well-founded, a rewrite proof must be obtained eventually. Proof terms may contain variables and hence represent proofs of equations under hypotheses, i.e. contexts in which other equations can be assumed to hold true.

The concept of conditional rewriting which we use is the one proposed by Kaplan (1984*a*, *b*), and further developed by Jouannaud & Waldmann (1986). Conditions in rules are not restricted to formulas of a lower level in the specification hierarchy as in Remy & Zhang (1984) and Zhang & Remy (1985). It is only required that terms in conditions be smaller, according to some given reduction ordering, than the left side of the rule. Then, the recursive calls of the rewrite relation for testing the applicability of a rule always terminate and the rewrite relation becomes decidable.

† A preliminary version of this paper has appeared in the Proceedings of the First International Workshop on Conditional Term Rewriting, Orsay, June 1987, Springer Lecture Notes in Computer Science, 1988.

‡ This work was partially supported by the ESPRIT-project PROSPECTRA, ref. no. 390, at the University of Dortmund, Germany.

0747-7171/91/010051 + 31 $03.00/0

The restriction to reductive rules is sufficiently strong to allow for efficient rewriting (Kaplan, 1987). On the other hand it is usually the case that non-reductive equations are generated during completion as critical pairs between reductive rules. This problem has been reported by many authors, Kaplan (1984*a*), Jouannaud & Waldmann (1986), Ganzinger (1987), Kaplan & Remy (1987), Kounalis & Rusinowitch (1987) and Orejas (1987) among others. For a completion procedure to be useful in practice it is of principal importance that it has strong enough techniques to handle these critical pairs. In this paper we describe two concepts for such techniques.

The first concept is devoted to the *elimination* of equations and rules. The idea is that an equation $C \Rightarrow s = t$ can be discarded if there is also another proof of the same conditional equation, different from the one which led to the construction of the equation. In addition this proof has to be simpler with respect to the complexity measure on proofs. To that end, proofs of equations $s = t$ under *hypotheses* C are represented as terms with variables (representing the "unknown" proofs for the equations in C) in our algebra of proofs. Particularly useful techniques for obtaining such simpler proofs are certain efficient variations of rewriting $s = t$ modulo the (skolemized) conditions in C of an equation, and subsumption. It will be demonstrated by means of examples that practical procedures must provide adequate combinations of these principal techniques.

The second concept is that of *superposing rules on conditions of non-reductive conditional equations.* Hereby, the set of *solutions* of the condition is enumerated. For particular classes of solutions specific instances of the equation will be generated. Any of the instances can then be treated specifically. Some instance may become a reductive rule, for other instances the previously mentioned elimination techniques may be applicable, for still other instances one may have to investigate the corresponding class of solutions further by again superposing rules on its condition. If this narrowing-like process comes to an end, the original equation need not be considered any more when enumerating the equational theory of the specification. A main difficulty in practice is, however, to detect loops in this process when it does not terminate. It is well-known (Réty, 1988) that naive approaches to (conditional) narrowing almost never terminate.

With these two concepts, the procedure presented in this paper is a considerable improvement over the ones given by Kaplan (1984*b*), Jouannaud & Waldmann (1986) and Kaplan & Remy (1987). Another improvement over previous work is that our procedure can be used to process *modules* of a complex specification separately. Upon combination of two preprocessed and hence canonical systems of rewrite rules only the inferences between axioms in different modules need to be computed. The situation here is thus the same as in the unconditional case. Another similarity to the unconditional case is a technique in which rules are numbered to achieve fair computation of critical pairs and to avoid recomputation of critical pairs for those rules that are simplified on their condition and right-hand side at a later stage in the completion process. This technique has been proven correct by Huet (1981) for the unconditional case and implemented in most existing completion procedures.

The approach taken in this paper shows, in its treatment of non-reductive equations, quite some similarities with recent work by Kounalis & Rusinowitch (1987). In fact, the idea of superposing rules on non-reductive conditions of equations has been taken from their work. One difference between both approaches is that our procedure will fail when it encounters a non-orientable *unconditional* equation which it cannot eliminate. On the other hand, if our procedure terminates it will have constructed a *reductive* set of conditional rewrite rules which is confluent on *all* terms, and not just ground-confluent

as in the case in Kounalis & Rusinowitch (1987). In the latter approach rewriting in the completed system is less efficient as it can require reductivity proofs for the substitution instances of an equation during rewriting. An unfailing variant of our procedure (which can be designed and proven correct according to the ideas described in Hsiang & Rusinowitch (1987) and Bachmair (1987) for the unconditional case) would combine the advantages of both approaches in the Horn clauses case.

Conditional equations are a particular case of first-order clauses with equality. In fact, the work of Kounalis & Rusinowitch (1987) is just a restriction to Horn clauses of the first-order theorem prover described by Rusinowitch (1987). More recent work on this subject is by Bachmair & Ganzinger (1990, 1991) who further optimize the inference system, provide a general concept of redundancy as a basis for simplification and elimination of clauses, and extend the idea of Knuth-Bendix-completion to full first-order clauses. This latter work subsumes most of the results of the present paper. Nevertheless, the treatment here is of interest due to the use of proof transformations and a more elaborate ordering on clauses. Proof transformations for the related case of refutation theorem proving in first-order logic which are based on a different proof algebra are studied by Bachmair (1989). Other related relevant work in this area is by Nieuwenhuis & Orejas (1991).

The procedure described in this paper has been implemented in the CEC system (Ganzinger & Schäfers, 1990) and been successfully applied in practice. Some examples will be shown in section 6.

2. Basic Notions and Notations

We consider terms over many-sorted signatures. A signature $\Sigma = (S, \Omega)$ consists of a set of sorts S and a family Ω of sets of operator symbols with arity in S^*. T_Σ denotes the set of all Σ-terms, $(T_\Sigma)_s$ is the set of terms of sort $s \in S$. By X we denote a fixed set of sorted variables containing denumerably infinitely many variables for each sort. $T_\Sigma(X)$ is the set of all terms that may contain variables from X. Given a term or formula t, $var(t)$ denotes the set of variables occurring in t.

Substitutions are denoted by σ, σ', etc. and their application to a term t by $t\sigma$. Substitutions σ with domain $x_1, \ldots, x_n$ are also written as $[x_1\sigma/x_1, \ldots, x_n\sigma/x_n]$. The identity substitution (with empty domain) is denoted by Λ. If o is an occurrence in t, then t/o denotes the subterm of t at o and $t[o \leftarrow t']$ denotes the result of subterm replacement at o using t'. $t[o_1 \leftarrow t_1, \ldots, o_n \leftarrow t_n]$, for independent occurrences o_i in t, is a shorthand for $t[o_1 \leftarrow t_1] \ldots [o_n \leftarrow t_n]$.

If $t \in T_\Sigma(X)$, by $\tilde{t}$ we denote the term obtained from t by considering the variables as new constants.

We assume a reduction ordering $>$ on $T_\Sigma(X)$ to be given. A reduction ordering is a well-founded ordering which is compatible with operators and stable under substitutions. If $>$ is a reduction ordering and st is the strict subterm ordering, then the transitive closure $>_{st}$ of $(> \bigcup st)$ is a Noetherian order on $T_\Sigma(X)$ which is stable under substitutions and satisfies the subterm property, i.e. terms are greater than any of their proper subterms.

We also assume that (for each sort) we have an auxiliary constant [ ] in Σ such that [ ] is smaller than any other non-variable term in $T_\Sigma(X)$ wrt. the given reduction ordering.

A *context* N is a term with exactly one occurrence of [ ], indicating a hole into which other terms can be inserted. $N[s]$ denotes the replacement of the hole in N by s. The distance of the hole from the root is called the depth of a context. Contexts of depth 0 simply consist of a hole. These are also called *empty contexts.*

A conditional equation over Σ is a formula of form

$$t_1 = t_1' \wedge \cdots \wedge t_n = t_n' \Rightarrow t_0 = t_0',$$

where $n \geq 0$ and $t_i, t_i' \in T_\Sigma(X)_{s_i}$. A conditional equation in which the order of terms in the conclusion is relevant is called a conditional rewrite rule. We will use the arrow $\rightarrow$ to explicitly distinguish rules from equations. For conditional rewrite rules, an additional requirement is $var(t_0') \subseteq var(t_0)$ and, for $i > 0$, $var(t_i) \subseteq var(t_0)$ and $var(t_i') \subseteq var(t_0)$, i.e. each variable that occurs in the rule must already occur in the left side of the conclusion.

In this paper, we assume the reader to be familiar with the basic properties of reduction orderings and in particular the recursive path ordering. A recursive path ordering of terms is obtained by lifting a well-founded ordering $>$ on the (possibly infinite) set of operators of the given signature to paths in terms. The reader may consult Dershowitz & Manna (1979), Huet & Oppen (1986) and Dershowitz (1987) for definitions and basic results. We briefly repeat some of the basic properties needed below.

Recursive path orderings $>$ are simplification orderings. That is they are compatible with operators, stable under substitutions, and satisfy the subterm property. Such orderings are in particular well-founded. In this paper we will make use of the following fact about recursive path orderings. Suppose two terms s and t are given such that there is a subterm s' of s for which $var(t) \subseteq var(s')$ and $s'(\varepsilon) > t(o)$, for each non-variable occurrence o in t. Then, $s > t$. ($s'(\varepsilon)$ is the root operator in s', $t(o)$ is the operator at o in t.) This property about operator precedences is not preserved under substitutions. Nevertheless, $s\sigma > t\sigma$ will follow from the stability of the recursive path ordering under substitutions.

The operators in proof terms will be ordered by making use of multiset orderings, cf. Dershowitz & Manna (1979) for details. For multisets we have $M > N$, if N can be obtained by replacing one or more elements in M by any finite number of smaller elements. The multiset ordering is Noetherian on finite multisets, provided the ordering on elements is.

Kaplan's concept of conditional rewriting introduces the notion of a simplifying rewrite rule. It has been generalized to the notion of reductive rewrite rules by Jouannaud & Waldmann (1986). A rule is reductive, if $t_0 > t_0'$ and, for $i > 0$, $t_0 > t_i$ and $t_0 > t_i'$, i.e. if the term on the right side and each term that occurs in the condition is to be smaller than the left side of the equation.

Let R be a set of Σ-rules and $t, t' \in T_\Sigma(X)$. The rewrite relation $t \rightarrow_R t'$ is given as the least fixpoint of the following recursive definition: $t \rightarrow_R t'$ iff there exists a rule

$$t_1 = t_1' \wedge \cdots \wedge t_n = t_n' \Rightarrow l \rightarrow r$$

in R, an occurrence o in t, a substitution $\sigma : X \rightarrow T_\Sigma(X)$ such that $t/o = l\sigma$, $t' = t[o \leftarrow r\sigma]$, and for each $i \leq n$ there exists a term s_i such that $t_i\sigma \rightarrow_R^* s_i$ and $t_i'\sigma \rightarrow_R^* s_i$. (We will subsequently write $t_i\sigma \downarrow_R t_i'\sigma$ to denote this converging of t_i and t_i' under R.)

In Kaplan (1984*b*) and Jouannaud & Waldmann (1986) it is shown that, in the case of finite and reductive R, $\rightarrow_R$ is decidable and finitely terminating. (This is proved mainly

by the observation that $N[l\sigma] >_{st} t_i\sigma$ and $N[l\sigma] >_{st} t_i'\sigma$, for any context N and substitution σ.) For terminating R, local confluence is equivalent to global confluence and then $\equiv_R = \downarrow_R$. In this paper, conditional rewrite rules are always assumed to be reductive. A conditional equation, however, may have conditions of arbitrary complexity.

3. Proof Terms for Conditional Equational Logic

3.1. THE SIGNATURE OF PROOFS

Proofs are terms in which the operators represent applications of logical inference rules. By the "propositions-as-types" paradigm, the sort of a proof term is the proved theorem. Hence, for equational logic, the sorts of proof terms are unconditional Σ-equations $u=v$, for $u, v \in T_\Sigma(X)_s$, $s \in S$.

Given sets E and R of conditional equations and conditional rewrite rules respectively, the following set of operators (inference rules) is complete for $\equiv_{E\cup R}$:

(*a*) *Substitution instances of rules and equations*:

$$apply_{\eta,\sigma} : c_1\sigma \times \cdots \times c_k\sigma \to (s\sigma = t\sigma),$$

and

$$applyR_{\eta,\sigma} : c_1\sigma \times \cdots \times c_k\sigma \to (t\sigma = s\sigma),$$

for $\eta \equiv C \Rightarrow s \to t \in R$ or $\eta \equiv C \Rightarrow s = t \in E$, where $C \equiv c_1 \wedge \cdots \wedge c_k$, $k \geq 0$. Hence, $apply_{\eta,\sigma}$ and $applyR_{\eta,\sigma}$ are k-ary operators that map proofs for the condition instances $c_i\sigma$ to a proof for $s\sigma = t\sigma$ and $t\sigma = s\sigma$, respectively. The signature of proof terms has a particular operator for each (*head*) *application and reverse application* of a rule or equation. The signature of proof terms is therefore infinite, even if the set of rules and equations is finite. We will also use the notation

$$\frac{P_1 \ldots P_k}{s\sigma \to t\sigma} \quad \text{for } apply_{C\Rightarrow s\to t,\sigma}(P_1, \ldots, P_k),$$

$$\frac{P_1 \ldots P_k}{t\sigma \leftarrow s\sigma} \quad \text{for } applyR_{C\Rightarrow s\to t,\sigma}(P_1, \ldots, P_k),$$

$$\frac{P_1 \ldots P_k}{s\sigma \leftrightarrow t\sigma} \quad \text{for } apply_{C\Rightarrow s=t,\sigma}(P_1, \ldots, P_k), \quad \text{and}$$

$$\frac{P_1 \ldots P_k}{t\sigma \leftrightarrow s\sigma} \quad \text{for } applyR_{C\Rightarrow s=t,\sigma}(P_1, \ldots, P_k).$$

This notation obviously abstracts from the details about the used rule or equation and substitution, and hence will not always be sufficiently precise. Where needed we will provide additional information separately, e.g. attach the rules or equations as subscripts. In the case of an empty condition we will simply write $s\sigma\ \rho\ t\sigma$, for $\rho \in \{\to, \leftarrow, \leftrightarrow\}$.

(*b*) *Reflexivity and Transitivity*:

$$\_;\_\ldots\_;\_ : (s_0 = s_1) \times (s_1 = s_2) \times \cdots \times (s_{n-1} = s_n) \to (s_0 = s_n),\ n \geq 2, \text{ or } n = 0.$$

This operator allows to form sequences of proofs. We have chosen a variadic operator to abstract from the obvious associativity property of the binary ";". The case $n = 0$ represents the reflexivity axioms $s_0 = s_0$. We will also use the notation

$$\frac{P_{11} \ldots P_{1k_1} P_{21} \ldots P_{2k_2} \ldots \ldots P_{n1} \ldots P_{nk_n}}{s_0 \rho_0 s_1 \rho_1 s_2 \ldots s_{n-1} \rho_{n-1} s_n}$$

for

$$\frac{P_{11} \ldots P_{1k_1}}{s_0 \rho_0 s_1}; \quad \frac{P_{21} \ldots P_{2k_2}}{s_1 \rho_1 s_2}; \quad \ldots; \quad \frac{P_{n1} \ldots P_{nk_n}}{s_{n-1} \rho_{n-1} s_n},$$

where $\rho_i \in \{\to, \leftarrow, \leftrightarrow\}$. Another notation is

$$\frac{Q_1 \ldots Q_k}{s_0 \rho^* s_n} \quad \text{for} \quad \frac{Q_1 \ldots Q_k}{s_0 \rho s_1 \rho \ldots \rho s_n}, \quad n \geq 0, \rho \in \{\to, \leftarrow, \leftrightarrow\}.$$

(*c*) *Compatibility with contexts*:
Contexts N of depth 1 take proofs for $s = t$ into proofs of $N[s] = N[t]$:

$$N[\_] : (s = t) \to (N[s] = N[t]).$$

We will write

$$\frac{P_1 \ldots P_k}{N[s_1] \rho_1 N[s_2] \rho_2 \ldots \rho_{n-1} N[s_n]} \quad \text{for} \quad N\left[\frac{P_1 \ldots P_k}{s_1 \rho_1 s_2 \rho_2 \ldots \rho_{n-1} s_n}\right].$$

Formally, contexts of depth >1 have to be viewed as nested applications of contexts of depth 1.

(*d*) *Symmetry*:

$$i_{s,t} : (s = t) \to (t = s).$$

The subscripts s, t will usually be omitted.

Given E and R, we will denote by $\mathscr{C}(E, R)$ this signature of proof rules of the equational calculus.

As we have operators for applying rules and equations both ways, the operator for the symmetry of equality is redundant. It will be used as an auxiliary operator for defining transformations of proof terms. We also could have avoided the introduction of context operators if we had extended the notion of rule and equation application to applications in context. This is done in Bachmair *et al.* (1986), Bachmair (1987) and Küchlin (1986). In these papers, moreover, proofs P and inverse proofs $i_{s,t}(P)$ are identified. Hence, there is no need for the i-operators at all. We have felt that our slightly more complex notion of proof orderings to be developed later justifies our slightly more redundant signature of proof terms. The redundancies will be removed by proof normalization rules to be given below.

For an example, suppose we have the following set E of equations

1 $(0 < s0) = tt$
2 $(0 < 0) = ff$
3 $(sx < y) = (x < py)$
4 $(px < y) = (x < sy)$
5 $(0 < x) = ff \Rightarrow (0 < px) = ff$

and the set R of rules

6 $(0 < x) = tt \Rightarrow (0 < sx) \to tt$
7 $spx \to x$
8 $psx \to x$

In this case,

$$\frac{\dfrac{(0 < s0) \leftrightarrow_1 tt}{(0 < ss0) \to_6 tt}}{(p0 < sssp0) \to_7 (p0 < ss0) \leftrightarrow_4 (0 < sss0) \to_6 tt}$$

is a proof for $(p0 < sssp0) = tt$, in which we have explicitly indicated the used equations and rules. The inference notation of proof terms also abstracts from the precise interleaving between the context operators and the other proof operators. For normalized proof terms there is always one unique way of inferring this interleaving:

DEFINITION 2.1. A proof term is said to be *normalized* if it is in normal form with respect to the following proof transformations (we use the arrow $\longrightarrow$ to distinguish the rewriting of proof terms for the rewriting $\to$ of object terms; π and π_i are variables for proofs):

$$i(K[\pi]) \longrightarrow K[i(\pi)]$$
$$K[\pi_1; \pi_2; \ldots; \pi_n] \longrightarrow K[\pi_1]; K[\pi_2]; \ldots; K[\pi_n]$$
$$i(\pi_1; \pi_2; \ldots; \pi_n) \longrightarrow i(\pi_n); \ldots; i(\pi_2); i(\pi_1)$$
$$i(apply(\pi_1, \ldots, \pi_n)) \longrightarrow applyR(\pi_1, \ldots, \pi_n)$$
$$i(applyR(\pi_1, \ldots, \pi_n)) \longrightarrow apply(\pi_1, \ldots, \pi_n)$$
$$i(i(\pi)) \longrightarrow \pi$$

Normalization removes patterns of form $i(P)$, for a non-variable term P, and distributes contexts over sequences of rule and equation applications. The proof normalization rules are confluent and terminating.

DEFINITION 3.2. A (normalized) proof term is called a *rewrite proof* if it has the form

$$\frac{P_1 \ldots P_n}{s \to^* u \leftarrow^* t},$$

with rewrite proofs P_i, $1 \leq i \leq n$, $n \geq 0$.

Hence, rewrite proofs contain no applications of equations and no peaks $\leftarrow\!\!\rightarrow$ in rewrite rule applications. If $s \to_R t$, this step of conditional rewriting is represented by a proof

term of form $\frac{P_1 \ldots P_n}{s \to t}$ over $\mathscr{E}(E, R)$, with a tuple P_i of rewrite proofs for the appropriately substituted conditions of the used rule.

3.2. COMPLEXITY OF PROOFS

In order to define a well-founded ordering on proof terms, we will now introduce a complexity measure c on the proof operators. We will then, for any two proof operators F and G, define $F > G$ iff $c(F) > c(G)$. This precedence of operators will be well-founded. Therefore, the lifting of $>$ on operators to a recursive path ordering $>_{\mathscr{E}}$ on the proof terms will yield a simplification ordering on proof terms with variables, i.e. on $T_{\mathscr{E}}(X)$.†

We have assumed to be given a reduction ordering $>$ on $T_\Sigma(X)$. Moreover, let min be an element smaller than any term in $T_\Sigma(X)$ (including contexts). If C is a set of unconditional equations, by $\mathscr{T}(C)$ we denote the multiset of terms on the left and right sides of the equations in C. By $\cup$ we denote the union of multisets.

DEFINITION 3.3. We define

$$c(apply_{C \Rightarrow s \to t, \sigma}) = c(apply\, R_{C \Rightarrow s \to t, \sigma}) = (\{s\sigma\}, \{s\}, \mathscr{T}(C) \cup \{t\}),$$
$$c(apply_{C \Rightarrow s = t, \sigma}) = c(apply\, R_{C \Rightarrow s = t, \sigma}) = (\mathscr{T}(C\sigma) \cup \{s\sigma, t\sigma\}, \mathscr{T}(C) \cup \{s, t\}, \varnothing),$$
$$c(K[\_]) = (\varnothing, \varnothing, \{K\}),$$
$$c(\_; \ldots; \_) = (\varnothing, \varnothing, \varnothing),$$
$$c(i_{s,t}) = (\{\min\}, \varnothing, \varnothing).$$

These triples are compared lexicographically, using the lifting of $>_{st}$ to multisets of $T_\Sigma(X)$-terms for the first component, the lifting of the subsumption ordering $\gg$ on terms to multisets of terms for the second component and the ordering on multisets of terms induced by $>_{st}$ for the third component.

For an equation, the multiset of all substituted terms of the equation is the dominating component of the complexity of any application of the equation in a proof. For a rule, the substituted left side of the rule is the major complexity measure for its application. Because of the reductivity of rules, the left side dominates any other term in the rule anyway. Hence, the actual substitution plays a major role in the complexity of an application. In the following, if η is a conditional equation or rule, $c(\eta)$ will always denote the complexity $c(apply_{\eta,\Lambda})$ of an *application* of η under the identity substitution Λ.

Note that normalization of proof terms simplifies these wrt. $>_{\mathscr{E}}$. Any of the normalization rules $L \to R$ satisfies $L >_{\mathscr{E}} R$.

3.3. PROOFS OF CONDITIONAL EQUATIONS

Proofs of conditional equations can be written as proof terms with variables to represent the "assumed proofs" for the conditions. We denote by X_C, for any given multiset C of unconditional equations, a C-sorted family of sets of proof variables. Proof variables π of sort $s = t$ are also written as $\pi : s = t$. This representation of proofs for conditional

† More precisely, we compare equivalence classes of proof terms where two terms are equivalent if corresponding operators have the same complexity. This detail is relevant to show the reductivity of the proof normalization rules for the removal of $i_{s,t}$-operators and is needed nowhere else.

equations is complete as $C \Rightarrow s = t$ is valid in all models of $E \cup R$ iff $\tilde{s} \equiv_{E \cup R \cup \tilde{C}} \tilde{t}$. (Remember, $\tilde{s}$ and $\tilde{C}$ denotes the replacement of variables by constants in s and C, respectively.) For proof terms $P \in T_{\mathscr{G}(E,R)}(\{\pi_1 : e_1, \ldots, \pi_n : e_n\})_e$ we will also use the sequent-like notation

$$\pi_1 : e_1, \ldots, \tau_n : e_n \vdash_{E,R} P : e$$

to indicate the variables occurring in P, and abbreviate $\pi_i : e_i$ simply by e_i, where the names of the variables do not matter. As an example,

$$(0 < sx) = tt \vdash (p0 < x) \leftrightarrow_4 (0 < sx) = tt$$

represents a proof of $(0 < sx) = tt \Rightarrow (p0 < x) = tt$. The last step of the proof uses the hypothesis.

We will later have to transform proofs for conditional equations into proofs for substituted equations. Let σ be a Σ-substitution. If Q is a proof, by $\sigma(Q)$ we denote the proof obtained from P by the following rules:

$$\sigma(N[P]) = N\sigma[\sigma(P)]$$

$$\sigma(apply[R]_{D \Rightarrow lp\tau,\sigma'}(P_1, \ldots, P_k)) = apply[R]_{D \Rightarrow lp\tau,\sigma'\sigma}(\sigma(P_1), \ldots, \sigma(P_k))$$

$$\sigma(i_{s,t}(P)) = i_{s\sigma,t\sigma}(\sigma(P))$$

$$\sigma(P_1; \ldots; P_k) = \sigma(P_1); \ldots; \sigma(P_k)$$

$$\sigma(\pi : s = t) = \pi : s\sigma = t\sigma, \text{ for any proof variable } \pi.$$

Hence, if Q is a proof of $C \Rightarrow s = t$, $\sigma(Q)$ represents a proof of $C\sigma \Rightarrow s\sigma = t\sigma$. $\sigma(Q)$ can be viewed as the result of applying an $\mathscr{G}(E, R)$-signature morphism induced by σ to Q. It should not be mixed up with applications of substitutions τ of proofs for proof variables to a proof P. The latter would be written as $P\tau$.

When, during completion, one wants to eliminate a conditional equation, one is obliged to construct a proof of the equation which has a *bounded* complexity.

A proof $P \in T_{\mathscr{G}(E,R)}(X_C)_{s=t}$ is said to be *γ-bounded*, γ an operator in our algebra of proofs, if for any operator F in P it is $c(\gamma) > c(F)$. P is said to be *η-bounded*, η a conditional equation or rule, if P is $apply_{\eta,\Lambda}$-bounded. Hence, a proof for $C \Rightarrow s = t$ which is bounded by $C \Rightarrow s = t$ is in particular simpler than the proof that just applies $C \Rightarrow s = t$ under the identity substitution. The complexities are generally such that rewriting a term in an equation $C \Rightarrow s = t$ is always a proof that is $C \Rightarrow s = t$-bounded.

The order used to compare of the first component of our complexity triples is stable under substitutions. The two last components do not depend on the substitution with which an equation or rule is applied. Therefore, if $C \vdash_{E,R} Q : s = t$ is bounded by η then, for any σ, $\sigma(Q)$ is bounded by $apply_{\eta,\sigma}$.

4. Inference Rules for Conditional Equational Completion

Below we list the inference rules for conditional completion $\mathscr{CC}$.

(*O*) *Orienting an equation*

$$\frac{E \cup \{C \Rightarrow s = t\}, R}{E, R \cup \{C \Rightarrow s \to t\}}, \quad \text{if } \{s\} > \{t\} \cup \mathscr{T}(C).$$

We may orient an equation, if the left side is greater than each term in the condition as well as the term on the right side wrt. the given reduction ordering.

(*A*) *Adding a conditional equational consequence*

$$\frac{E, R}{E \cup \{C \Rightarrow s = t\}, R}, \quad \text{if } \exists C \vdash_{E,R} P : s = t.$$

Hence we may add an arbitrary conditional equational consequence to the set of equations. Concrete completion procedures will only make restricted use of this inference rule and add certain superposition instances of rules on rules and of rules on equations, cf. below.

(*SE*) *Simplifying an equation*

$$\frac{E \cup \{C \Rightarrow s = t\}, R}{E \cup \{C \Rightarrow u = t\}, R}, \quad \text{if } \exists C \vdash_{E,R} P : s = u \text{ bounded by } C \Rightarrow s = t, \text{ and } s > u.$$

We may simplify the conclusion of a conditional equation, if the complexity of the equation is thereby decreased and if there is a proof of $C \Rightarrow s = u$ which is less complex than applications of the unsimplified equation. Of course it is assumed that the symmetric case in which t is simplified is also covered by this rule.

(*D*) *Deleting a trivial equation*

$$\frac{E \cup \{C \Rightarrow s = t\}, R}{E, R}, \quad \text{if } \exists C \vdash_{E,R} P : s = t \text{ bounded by } C \Rightarrow s = t.$$

an equation may be deleted, if there is a simpler proof for it than an application of the equation itself.

(*SC*) *Simplifying a condition*

$$\frac{E \cup \{C \wedge u = v \Rightarrow s = t\}, R}{E \cup \{C \wedge w = v \Rightarrow s = t\}, R}, \quad \text{if } \exists C \vdash_{E,R} P : u = w$$

bounded by $C \wedge u = v \Rightarrow s = t$ and $u > w$.

$$\frac{E, R \cup \{C \wedge u = v \Rightarrow s \to t\}}{E, R \cup \{C \wedge w = v \Rightarrow s \to t\}}, \quad \text{if } \exists C \vdash_{E,R} P : u = w$$

bounded by $C \wedge u = v \Rightarrow t = t$ and $u > w$.

A condition may be simplified under the assumption that the other condition equations hold true. The symmetric case, in which v is rewritten, is assumed to be also covered by these rules. The difference in complexity bounds for equations and rules is due to the fact that we do not want to recompute superpositions from rules that have been simplified on their conditions.

(*DC*) *Deleting a trivial condition*

$$\frac{E \cup \{C \wedge u = v \Rightarrow s = t\}, R}{E \cup \{C \Rightarrow s = t\}, R}, \quad \text{if } \exists C \vdash_{E,R} P : u = v.$$

$$\frac{E, R \cup \{C \wedge u = v \Rightarrow s \to t\}}{E, R \cup \{C \Rightarrow s \to t\}}, \quad \text{if } \exists C \vdash_{E,R} P : u = v \text{ bounded by } C \wedge u = v \Rightarrow t = t.$$

A condition which is subsumed by the other conditions may always be deleted. The proof for $C \Rightarrow u = v$ may be arbitrarily complex in the case of equations. In the case of rules we require a complexity bound to not have to recompute superpositions with the simplified rule.

(*SRL*) *Simplifying the left side of a rule*

$$\frac{E, R \cup \{C \Rightarrow s \to t\}}{E \cup \{C \Rightarrow u = t\}, R}, \quad \text{if } \exists C \vdash_{E,R} P : s = u \text{ bounded by } C \Rightarrow s \to t, \text{ and } s > u.$$

(*SRR*) *Simplifying the right side of a rule*

$$\frac{E, R \cup \{C \Rightarrow s \to t\}}{E, R \cup \{C \Rightarrow s \to u\}}, \quad \text{if } \exists C \vdash_{E,R} P : t = u \text{ bounded by } C \Rightarrow t \to t, \text{ and } t > u.$$

The bound $C \Rightarrow t \to t$ for P will allow to avoid the recomputation of superpositions with the simplified rules.

It is obvious that any of the above inference rules leaves the congruence $\equiv_{E \cup R}$ invariant. We now demonstrate that proof terms become less complex in proof signatures that are obtained by inference rule application.

LEMMA 4.1. *The ordering $>_p$ is a* proof ordering *for conditional completion, i.e. for any inference $(E, R) \vdash_{\mathscr{CC}} (E', R')$ and any proof term $P \in (T_{\mathscr{E}(E,R)})_{u=v}$ we have $P \in (T_{\mathscr{E}(E',R')})_{u=v}$, i.e. P is also a proof of $u = v$ in the new system, or there exists a proof $P' \in (T_{\mathscr{E}(E',R')})_{u=v}$ of $u = v$ in the new system which is less complex, i.e. $P >_p P'$.*

PROOF. We demonstrate, for any of the *CC*-inference rules, how "old" proofs can be simulated in the new system. The simulations will be represented as sets of rewrite rules $L \to R$ on proof terms such that $L >_p R$. Hence, by application of these proof rewrite rules, proofs become smaller. If a proof cannot be rewritten, it is already a valid proof in the new system $\mathscr{E}(E', R')$. The rules are, as usually, meant as rule schemes in which arbitrary proof terms (with or without variables) may be substituted for the variables. Variables for proofs are denoted by π_i. The inference rules mostly assume, as a side condition, the existence of some bounded proof $\kappa_1 : e_1, \ldots, \kappa_n : e_n \vdash_{E,R} P$ of some equation with condition $C = \{e_1, \ldots, e_n\}$. In these cases, P' will denote the proof $\sigma(P)[\pi_i/\kappa_i; 1 \leq i \leq n]$ and the variables π_i will be of the required type $e_i\sigma$.

ad (*O*).

Suppose $(E \cup \{C \Rightarrow s = t\}, R) \vdash_{\mathscr{CC}} (E, R \cup \{C \Rightarrow s \to t\})$ and $s > t$. The rules

$$\frac{\pi_1 \ldots \pi_n}{s\sigma \leftrightarrow_{C \Rightarrow s=t} t\sigma} \longrightarrow \frac{\pi_1 \ldots \pi_n}{s\sigma \to_{C \Rightarrow s \to t} t\sigma}$$

and

$$\frac{\pi_1 \ldots \pi_n}{t\sigma \leftrightarrow_{C \Rightarrow s=t} s\sigma} \longrightarrow \frac{\pi_1 \ldots \pi_n}{t\sigma \leftarrow_{C \Rightarrow s \to t} s\sigma}$$

replace any application of the equation by an application of the rule. It is $c(apply_{C \Rightarrow s=t,\sigma}) = (\mathscr{T}(C\sigma) \cup \{s\sigma, t\sigma\}, \mathscr{T}(C) \cup \{s, t\}, \varnothing)$. The latter is greater than $(\{s\sigma\}, \{s\}, \mathscr{T}(C) \cup \{t\}) = c(apply_{C \Rightarrow s \to t,\sigma})$. Hence the proof transformation rules are reductive.

ad (*A*).

Nothing needs to be shown, as any old proof term is a valid proof term in the transformed proof signature.

ad (SE).
In this case, old proofs can be rewritten to new proofs by

$$\frac{\pi_1 \ldots \pi_n}{s\sigma \leftrightarrow_{C \Rightarrow s=t} t\sigma} \longrightarrow P'; \frac{\pi_1 \ldots \pi_n}{u\sigma \leftrightarrow_{C \Rightarrow u=t} t\sigma}$$

$$\frac{\pi_1 \ldots \pi_n}{t\sigma \leftrightarrow_{C \Rightarrow s=t} s\sigma} \longrightarrow \frac{\pi_1 \ldots \pi_n}{t\sigma \leftrightarrow_{C \Rightarrow u=t} u\sigma}; i(P').$$

The reductivity of these rules follows from the boundedness of P by $C \Rightarrow s = t$, and hence the boundedness of P' by $apply_{C \Rightarrow s=t,\sigma}$. Therefore the application of the original equation $C \Rightarrow s = t$ under substitution σ is more complex than any operator in P'. Moreover, $c(apply_{C \Rightarrow s=t,\sigma}) > c(apply_{C \Rightarrow u=t,\sigma})$, as $s > u$. Note also that the operator $i(\_)$ has a complexity less than any application of a rule or equation.

ad (D).

$$\frac{\pi_1 \ldots \pi_n}{s\sigma \leftrightarrow_{C \Rightarrow s=t} t\sigma} \longrightarrow P'$$

$$\frac{\pi_1 \ldots \pi_n}{t\sigma \leftrightarrow_{C \Rightarrow s=t} s\sigma} \longrightarrow i(P')$$

achieve the desired transformations.

ad (SC).
We first prove the case in which the condition of an equation is simplified. The following rules can be used to construct new proofs using the simplified equation. π is assumed to be a variable of sort $u\sigma = v\sigma$.

$$\frac{\pi_1 \ldots \pi_n\, \pi}{s\sigma \leftrightarrow_{C \wedge u=v \Rightarrow s=t} t\sigma} \longrightarrow \frac{\pi_1 \ldots \pi_n\, i(P');\pi}{s\sigma \leftrightarrow_{C \wedge w=v \Rightarrow s=t} t\sigma}$$

$$\frac{\pi_1 \ldots \pi_n\, \pi}{t\sigma \leftrightarrow_{C \wedge u=v \Rightarrow s=t} s\sigma} \longrightarrow \frac{\pi_1 \ldots \pi_n\, i(P');\pi}{t\sigma \leftrightarrow_{C \wedge w=v \Rightarrow s=t} s\sigma}$$

Note that

$$\begin{aligned} c(apply_{C \wedge u=v \Rightarrow s=t,\sigma}) &= (\mathcal{T}((C \wedge u = v)\sigma) \cup \{s\sigma, t\sigma\}, \mathcal{T}(C \wedge u = v) \cup \{S, t\}, \varnothing) \\ &> (\mathcal{T}((C \wedge w = v)\sigma) \cup \{s\sigma, t\sigma\}, \mathcal{T}(C \wedge w = v) \cup \{s, t\}, \varnothing) \\ &= c(apply_{C \wedge w=v \Rightarrow s=t,\sigma}). \end{aligned}$$

By construction, P' is less complex than the given application of the original equation.

In the case of simplifying the condition of a rule, the third component in our operator complexities comes into play. We have

$$\begin{aligned} c(apply_{C \wedge u=v \Rightarrow s \to t,\sigma}) &= (\{s\sigma\}, \{s\}, \mathcal{T}(C \wedge u = v) \cup \{t\}) \\ &> (\{s\sigma\}, \{s\}, \mathcal{T}(C \wedge w = v) \cup \{t\}) \\ &= c(apply_{C \wedge w=v \Rightarrow s \to t,\sigma}). \end{aligned}$$

Hence, the same kind of proof transformation is reductive also for the case of a simplified rewrite rule.

ad (*DC*).

We again prove the case in which the condition of an equation is simplified. Here we have the rule

$$\frac{\pi_1 \ldots \pi_n \pi}{s\sigma \leftrightarrow_{C \wedge u=v \Rightarrow s=t} t\sigma} \longrightarrow \frac{\pi_1 \ldots \pi_n}{s\sigma \leftrightarrow_{C \Rightarrow s=t} t\sigma}$$

the reductivity of which is obvious.

ad (*SRL*).

Old proofs can be rewritten to new proofs by

$$\frac{\pi_1 \ldots \pi_n}{s\sigma \rightarrow_{C \Rightarrow s \rightarrow t)} t\sigma} \longrightarrow P'; \frac{\pi_1 \ldots \pi_n}{u\sigma \rightarrow_{C \Rightarrow u=t} t\sigma}$$

$$\frac{\pi_1 \ldots \pi_n}{t\sigma \leftarrow_{C \Rightarrow s \rightarrow t} s\sigma} \longrightarrow \frac{\pi_1 \ldots \pi_n}{t\sigma \leftrightarrow_{C \Rightarrow u=t} u\sigma}; i(P')$$

The reductivity of these rules follows from the fact that both u and t as well as any term in the condition C are simpler than s. Also, the boundedness of P and hence P' is relevant.

ad (*SRR*).

Old proofs can be rewritten to new proofs by

$$\frac{\pi_1 \ldots \pi_n}{s\sigma \rightarrow_{C \Rightarrow s \rightarrow t} t\sigma} \longrightarrow \frac{\pi_1 \ldots \pi_n}{s\sigma \rightarrow_{C \Rightarrow s \rightarrow u} u\sigma}; i(P')$$

$$\frac{\pi_1 \ldots \pi_n}{t\sigma \leftarrow_{C \Rightarrow s \rightarrow t} s\sigma} \longrightarrow P'; \frac{\pi_1 \ldots \pi_n}{u\sigma \leftarrow_{C \Rightarrow s \rightarrow u} s\sigma}$$

As $t > u$, the reductivity of these rules follows from the third component in the complexity measure for rewrite rule applications.

Note that for this proof we need the ordering $>_p$ on proofs to be stable under substitutions. Some of the above rewrite rules are not right-linear. Therefore it is not possible to simply take the multiset of the complexities of operators in a proof term as complexity measure.

A *completion procedure*, i.e. a mechanism that computes sequences $(E_0, R_0), (E_1, R_1), \ldots$ of derivations using the above inference rules for conditional completion, is *complete*, if any proof of an equation can be transformed into a rewrite proof in some derivate (E_i, R_i) of the initial specification (E_0, R_0). To prove this it is sufficient to show that any (normalized) proof $P \in (T_{\mathscr{C}(E_j, R_j)})_{u=v}$ which is not a rewrite proof can be transformed into a less complex proof $Q \in (T_{\mathscr{C}(E_k, R_k)})_{u=v}$, for some k. As $>_p$ is well-founded and a proof ordering for conditional completion, after finitely many such transformations a rewrite proof must be obtained.

If a normalized proof term is not a rewrite proof, it must contain an application of an equation or a subproof of form $s \leftarrow u \rightarrow w$. If the latter situation cannot be simplified, a *critical pair* must be added to the set of equations. The following is the notion of critical pairs in the conditional case.

DEFINITION 4.2. Let two conditional rules $C \Rightarrow M \rightarrow N$ and $D \Rightarrow G \rightarrow H$ be given and assume that their variables have been renamed such that they do not have any common variables. Assume moreover that o is a non-variable occurrence in M such that M/o and

G can be unified with a mgu σ. Then, $(C \wedge D)\sigma \Rightarrow M[o \leftarrow H]\sigma = N\sigma$ is a (*contextual*) *critical pair* between the two rules.

The following lemma is a slight generalization of the lemma by Jouannaud & Waldmann (1986) which proves that for reductive rewrite rules local confluence is equivalent to the convergence of all critical pairs.

LEMMA 4.3. *Let R be reductive. If $s \leftarrow_R w \rightarrow_R t$, then there is also a simpler proof Q of $s = t$ or there exists a contextual critical pair $C \Rightarrow c = d$ between two rules in R such that $s = N[c\sigma]$, $t = N[d\sigma]$ and $C\sigma \subset \downarrow_R$. Moreover, Q does not contain any application of an equation. In particular, Q is a rewrite proof in the case of unconditional rewriting of w.*

PROOF. We will construct a simpler proof of $s = t$ in those cases in which the two rewritings of w do not overlap. In the remaining case the assertion follows immediately from the definition of contextual critical pairs.

(a) *The rewritings have disjoint redexes*
Then the given proof of $s = t$ is of form

$$\frac{P_1}{N[o_2 \leftarrow l_2\sigma_2][r_1\sigma_1] \leftarrow N[o_2 \leftarrow l_2\sigma_2][l_1\sigma_1]}; \frac{P_2}{N[o_1 \leftarrow l_1\sigma_1][l_2\sigma_2] \rightarrow N[o_1 \leftarrow l_1\sigma_1][r_2\sigma_2]}$$

with independent occurrences σ_1 and o_2 in N. The proof

$$\frac{P_2}{N[o_1 \leftarrow r_1\sigma_1][l_2\sigma_2] \rightarrow N[o_1 \leftarrow r_1\sigma_1][r_2\sigma_2]}; \frac{P_1}{N[o_2 \leftarrow r_2\sigma_2][r_1\sigma_1] \leftarrow N[o_2 \leftarrow r_2\sigma_2][l_1\sigma_1]}$$

is less complex as the context terms in the proof algebra which correspond to the reduced contexts $N[o_i \leftarrow r_i\sigma_i]$ are less complex than those corresponding to the initial contexts $N[o_i \leftarrow l_i\sigma_i]$, $i = 1, 2$.

(b) *The rewritings occur one above the other*
Again we assume applications of rules $C_i \Rightarrow l_i \rightarrow r_i$, $i = 1, 2$ with matching substitutions σ_i. Additionally we may assume the following situation.

(i) Rule 1 is applied at an occurrence o in context K.
(ii) l_2 occurs at occurrence ξ_1 of a variable x in l_1 inside w. (The general case in which l_2 occurs inside a bigger term at x is not really more complex for what we have to prove here.)
(iii) There are $n \geq 1$ occurrences ξ_i of x in l_1 and $m \geq 0$ occurrences ζ_i of x in r_1.
(iv) C_1 consists of exactly one equation, say $u = v$. u and v both have exactly one occurrence ω and π, respectively, of x. (The general case does only add notational clutter in the following proof.)
(v) P_1 is the rewrite proof for $u\sigma_1 = v\sigma_1$ used upon application of rule 1. P_2 is the tuple of rewrite proofs for the condition $C_2\sigma_2$.

Then the given proof P is of form

$$\frac{P_1}{K[r_1\sigma_1] \leftarrow K[l_1\sigma_1]}; \frac{P_2}{K'[l_2\sigma_2] \rightarrow K'[r_2\sigma_2]},$$

where

$$K' = K[o \leftarrow l_1\tilde{\sigma}_1][o.\xi_2 \leftarrow l_2\sigma_2, \ldots, o.\xi_n \leftarrow l_2\sigma_2]$$

with $\tilde{\sigma}_1$ as σ_1 except that it substitutes the variable x by a hole [ ].

A proof of $s = t$ in which the reverse application of rule 1 follows m applications of rule 2 would be the sequence Q_1; Q_2; Q_3 of the following proofs:

Q_1 represents m applications $l_2\sigma_2 \rightarrow r_2\sigma_2$ of rule 2 at the occurrences ζ_i of x in r_1 which are the occurrences $o.\zeta_i$ in

$$s \equiv K[o \leftarrow r_1\tilde{\sigma}_1][o.\zeta_i \leftarrow l_2\sigma_2, 1 \leq i \leq m].$$

This rewrites the latter term into the term

$$K[o \leftarrow r_1\tilde{\sigma}_1][o.\zeta_i \leftarrow r_2\sigma_2, 1 \leq i \leq m].$$

Q_2 represents the reverse application of rule 1 at o under a substitution σ_1' which is equal to σ_1 and $\tilde{\sigma}_1$, except that it substitutes x by $r_2\sigma_2$. Hereby the proof of the rewritten condition instance $u\sigma_1' = v\sigma_1'$ is constructed by (reversely) rewriting σ_1' into σ_1, then applying the given proof P_1 of $u\sigma_1 = v\sigma_1$ and finally rewriting σ_1 into σ_1'. (Note that $u\tilde{\sigma}_1[r_2\sigma_2] \equiv u\sigma_1'$, $u\tilde{\sigma}_1[l_2\sigma_2] \equiv u\sigma_1$ and $v\tilde{\sigma}_1[r_2\sigma_2] \equiv v\sigma_1'$, $v\tilde{\sigma}_1[l_2\sigma_2] \equiv v\sigma_1$.) Formally,

$$Q_2 = \frac{u\tilde{\sigma}_1[r_2\sigma_2] \leftarrow u\tilde{\sigma}_1[l_2\sigma_2];\ P_1;\ v\tilde{\sigma}_1[l_2\sigma_2] \rightarrow v\tilde{\sigma}_1[r_2\sigma_2]}{K[r_1\sigma_1'] \leftarrow K[l_1\sigma_1']}.$$

Q_3, finally, represents the $n-1$ reverse applications of rule 2 at the occurrences ξ_i, $i > 1$ in l_1, which are the occurrences $o.\xi_i$, $i > 1$, in

$$K[l_1\sigma_1'] \equiv K[o \leftarrow l_1\tilde{\sigma}_1][o.\xi_i \leftarrow r_2\sigma_2;\ 1 \leq i \leq n].$$

This takes this term into the term

$$t \equiv K[o \leftarrow l_1\tilde{\sigma}_1][o.\xi_1 \leftarrow r_2\sigma_2,\ o.\xi_i \leftarrow l_2\sigma_2;\ 2 \leq i \leq n],$$

as required. When comparing this second proof with the original one, we can make the following observations:

(1) $\dfrac{P_1}{K[r_1\sigma_1] \leftarrow K[l_1\sigma_1]} >_{\#} Q_2$:

This is because $l_1\sigma_1 > l_1\sigma_1'$ and because $(\{l_1\sigma_1\}, \{l_1\}, \{u, v, r_1\})$ is greater than the complexity of any of the applications of rule 2 in the proofs of $u\sigma_1' = u\sigma_1$ and $v\sigma_1 = v\sigma_1'$. This can be seen from $l_1\sigma_1 >_{st} l_2\sigma_2$. The reductivity of the rewrite rules is then needed to conclude that $l_2\sigma_2$ is more complex than any of the redexes in the proofs for $C_2\sigma_2$.

(2) $\dfrac{P_1}{K[r_1\sigma_1] \leftarrow K[l_1\sigma_1]} >_{\#} Q_3$:

To justify this claim we note that any rewriting in Q_3 occurs in terms smaller than $l_1\sigma_1$. Q_3 contains only applications of rules and contexts. For the same reason we obtain

(3) $\dfrac{P_1}{K[r_1\sigma_1] \leftarrow K[l_1\sigma_1]} >_{\#} Q_1$.

Altogether, $P >_{\#} Q_1; Q_2; Q_3$, which was to be shown.

In the unconditional case Q is a rewrite proof. Hence, Lemma 4.3 is the generalization to the case of conditional equations of Bachmair's formulation of the critical pairs lemma (Bachmair, 1987).

The following definition defines fairness of a completion procedure in a way such that its completeness for equational logic is induced. By CP_i we denote the set of critical pairs between any two rules in R_i.

DEFINITION 4.4. A CC-derivation $(E_0, R_0), (E_1, R_1), \ldots$ is called fair, if (1) and (2) are satisfied. Hereby, $R_\infty = \bigcup_i \bigcap_{j \geq i} R_j$ denotes the set of final rules of the process.

1. If $C \Rightarrow c = d \in \bigcap_{j \geq i} E_j$ for some i, and if for a substitution σ there exists a tuple P of rewrite proofs in R_∞ for the equations in $C\sigma$, then there exists an index k and a proof $Q \in (T_{\mathscr{C}(E_k, R_k)})_{c\sigma = d\sigma}$ for $c\sigma = d\sigma$ such that
$$\frac{P}{c\sigma \leftrightarrow_{C \Rightarrow c = d} d\sigma} >_{\#} Q.$$
2. Let $C \Rightarrow c = d \in \bigcap_{j \geq i} CP_j$, for some i, be obtained by superposing rules $A \Rightarrow l \to r$, $B \Rightarrow s \to t \in R_l$ with $l\sigma'$ the superposition term. If there exists a substitution σ, $\sigma = \sigma'\tau$, tuples P_A and P_B of rewrite proofs in R_∞ for the conditions $A\sigma$ and $B\sigma$, then there exists a k and a proof $Q \in T_{\mathscr{C}(E_k, R_k)}$ for $c\tau = d\tau$ such that
$$\frac{P_A \; P_B}{c\tau \leftarrow_{A \Rightarrow l \to r} l\sigma \to_{B \Rightarrow s \to t} d\tau} >_{\#} Q.$$

Note that we do not require that any proof which applies an equation or a peak be simplified. It is sufficient, as to be seen in the following lemma, that such a simplification exists in cases where the conditions of the corresponding applications of equations and rules are proved by rewrite proofs. Note also that we do not require that equations be either simplified or oriented eventually. As we shall see later there are other ways of simplifying applications of equations in proofs.

LEMMA 4.5. *Let $(E_0, R_0), (E_1, R_1), \ldots$ be a fair CC-derivation and P be a normalized proof in $(T_{\mathscr{C}(E_l, R_l)})_{u=v}$ for some equation $u = v$. If P is not a rewrite proof, then there is, for some k, a proof $P' \in (T_{\mathscr{C}(E_k, R_k)})_{u=v}$ such that $P >_{\#} P'$.*

PROOF. Suppose P to be given. If it is not a rewrite proof it contains an application of an equation, i.e. a subterm $\tilde{P}$ of form
$$\frac{Q}{s\sigma \leftrightarrow_{C \Rightarrow s = t} t\sigma}$$
or a peak, i.e. a subterm of form
$$\frac{R \; S}{s \leftarrow w \to t}.$$

We may assume that we have selected these subterms such that they have minimal height, that is the R, Q, S are (possibly empty) tuples of rewrite proofs. If any of the equations or rewrite rules in these proofs are eliminated in some later step m of the completion procedure, from Lemma 4.1 follows the existence of a simpler proof $\tilde{P}'$ for $\tilde{P}$ in $\mathscr{C}(E_m, R_m)$. In case all used rules and equations persist, the application of $C \Rightarrow s = t$ can be simplified by (1) in the definition of fairness. In the case of a persistent peak we may apply the critical pairs Lemma 4.3 and either construct a simpler proof or conclude that the branching is caused by a critical pair between two rules in R_i. A simpler proof $\tilde{P}'$ now exists because of the second fairness requirement. In any case we have managed to transform the critical subterm $\tilde{P}$ of P into a simpler proof $\tilde{P}'$ in $\mathscr{C}(E_m, R_m)$.

Replacing $\tilde{P}$ in P by $\tilde{P}'$ yields, because of the fact that $>_\mu$ is compatible with operators, a proof P'' which is simpler than P. However, it uses rules and equations from both $\mathscr{C}(E_m, R_m)$ and $\mathscr{C}(E_i, R_i)$. To construct a proof P' in $\mathscr{C}(E_k, R_k)$, $k = max(i, m)$, we can apply to $\tilde{P}'$ the transformations given in the proof of Lemma 4.1 according to the inference rules used along the derivation of $\mathscr{C}(E_k, R_k)$ from $\mathscr{C}(E_n, R_n)$, $n = min(i, m)$.

Hence, if $u \equiv_E v$ and if the completion procedure does not *fail* for inputs E and $>$, that is generates a fair CC-derivation, it will generate a pair (E_k, R_k) such that $u \downarrow_{R_k} v$. In particular, the limit $R_\infty = \bigcup_i \bigcap_{j \geq i} R_j$ is canonical and $\equiv_{R_\infty} = \equiv_{E_0 \cup R_0}$. In this case, the equational theory can be decided by rewriting. Note that the final set of equations need not be empty. The final equations however do not contribute to proofs of equations. They could in principle be eliminated. However, we rather prefer to keep them, as in the case of a subsequent enrichment of the specification they have to be reconsidered wrt. to fairness condition (1). An enrichment may increase the set of solutions of the equation's condition. This distinguishes the conditional from the unconditional case and this approach from the ones of Kaplan (1984*b*), Jouannaud & Waldmann (1986), Ganzinger (1987) and Kaplan & Remy (1987).

5. Completion Concepts

In this chapter we present the formal justification of some technical details of the completion procedure in CEC (Gazinger & Schäfers, 1990).

5.1. TECHNIQUES FOR OBTAINING BOUNDED PROOFS

As we have seen in the last section, simplifying conditions and conclusions of conditional equations and rules decreases the complexity of proofs. It also decreases the chances for failure of the completion procedure due to non-reductive rewrite rules.

The simplification inference rules and the inference rule (D) for the elimination of equations are based on bounded proofs of conditional equations. In practice one has to have appropriate techniques to be able to establish proofs of this kind. In the following we will present three such proof techniques that have shown to be useful.

5.1.1. REWRITING WITH EQUATIONS OF CONDITIONS

The first of our three techniques can be used if condition equations can themselves be oriented according to the given reduction ordering. In this case, proofs of conditional equations can be attempted by rewriting using the current set R of (reductive) rules and the (skolemized) condition equations as additional rewrite rules.

PROPOSITION 5.1. *Assume that any equation in a condition C can be oriented, i.e. it holds, w.l.o.g., for any equation $u = v \in C$, $u > v$.*

1. *For any two terms s and t, if $\tilde{s} \rightarrow_{R \cup \tilde{C}} \tilde{t}$, then there exists a $(C \Rightarrow s = \min)$-bounded proof $C \vdash_{\varnothing, R} P : s = t$ for $C \Rightarrow s = t$. Furthermore, this proof is even bounded by $C \Rightarrow s \rightarrow \min$, if $\{s\} > \mathscr{T}(C)$ and if the rule $D \Rightarrow l \rightarrow r$ that is applied to rewrite $\tilde{s}$ is either a member of $\tilde{C}$, or the rule is not applied at the top of $\tilde{s}$ or $\tilde{s} \gg l$.*
2. *For any two terms s and t, if $\tilde{s} \downarrow_{R \cup \tilde{C}} \tilde{t}$, then there exists a $C \Rightarrow s = t$-bounded proof $C \vdash_{\varnothing, R} P : s = t$ for $C \Rightarrow s = t$.*

PROOF. We prove this lemma by Noetherian induction over the complexity of $\tilde{s}$.

Ad 1. Let $D \Rightarrow l \to r$ be the rule that is applied in $\tilde{s} = \tilde{N}[o \leftarrow l\tilde{\sigma}]$, with $\tilde{\sigma}$ the matching substitution and o an occurrence of a hole. In case $D \Rightarrow l \to r$ is an equation $\tilde{u} = \tilde{v} \in \tilde{C}$, the rewriting corresponds to the proof term $N[\pi : u = v]$, with some context N of context operators and a proof variable $\pi : u = v$. This proof obviously has the required properties. Otherwise, the rule is a member of R. Because of the reductivity of R, any term in $D\tilde{\sigma}$ is smaller than $\tilde{s}$. Moreover, $D\tilde{\sigma} \subset \downarrow_{R \cup \tilde{C}}$, otherwise the rule would not be applicable. Using the induction hypothesis for (2) we may now assume the existence of $C \Rightarrow u\sigma = v\sigma$-bounded proofs $C \vdash_{\varnothing, R} P_{u=v} : u\sigma = v\sigma$, for any equation $u = v \in D$. The complexity of any operator in $P_{u=v}$ is hence bounded by $(\mathcal{T}(C) \cup \{u\sigma, v\sigma\}, \mathcal{T}(C) \cup \{u\sigma, v\sigma\}, \varnothing)$, the latter being smaller than $(\mathcal{T}(C) \cup \{s\}, \{s\}, \mathcal{T}(C) \cup \{\min\})$ and hence in particular smaller than $c(C \Rightarrow s = \min)$. Because of the reductivity of rewrite rules, the application of $D \Rightarrow l \to r$ is $C \Rightarrow s = \min$-bounded, too. Therefore,

$$\frac{\ldots P_{u=v} \ldots}{N[l\sigma] \to_{D \Rightarrow l \to r} N[r\sigma]}$$

is $C \Rightarrow s = \min$-bounded. If additionally s is greater than any term in C, we obtain $(\mathcal{T}(C) \cup \{s\}, \{s\}, \mathcal{T}(C) \cup \{\min\}) > (\{l\sigma\}, \{l\}, \mathcal{T}(C) \cup \{\min\})$, if the application is not at the top or if $s \gg l$. In this case, the rewriting is $C \Rightarrow s \to$ min-bounded.

Ad 2. As we may apply (1) to any rewrite step in $\tilde{s} \downarrow_{R \cup \tilde{C}} \tilde{t}$, this assertion is an immediate consequence of the reductivity of the rewrite rules.

For example, $\to_{R \cup \tilde{C}}$-writing is always possible (and then a good choice) in specifications in which conditions are restricted to Boolean conditions, i.e. equations of form $p = tt$ or $p = ff$, if we assume tt and ff to be smaller than any other Boolean term. More general classes of conditional equations of this kind are the ground-normal systems of Bergstra & Klop (1982). In general, one may always at least use those equations in C which can be oriented.

The following example shows some of the power of rewriting with condition equations as additional rules (list notation as in PROLOG):

(1) $x \neq y = tt \Rightarrow$ delete([y|ys], x) $\to$ [y|delete(ys, x)]
(2) has(xs, x) = ff $\Rightarrow$ delete(xs, x) $\to$ xs
(3) $x \neq y = tt \Rightarrow$ has([y|ys], x) $\to$ has(ys, x)

Overlapping (1) and (2) yields the critical pair

$x \neq y = tt$ and has([y|ys], x) = ff $\Rightarrow$ [y|delete(yx, s)] $\to$ [y|ys].

Rule (3) can be applied in $has([\tilde{y}|\tilde{ys}], \tilde{x})$ as $\tilde{x} \neq \tilde{y}$ rewrites to tt, this is the first condition of the critical pair. This simplifies the second condition to $has(ys, x) = ff$. Now, in the next simplification step, $[\tilde{y}|delete(\tilde{ys}, \tilde{x})]$ can be rewritten under $R \cup \tilde{D}$, where D is the simplified condition $D = \{x \neq y = tt,\ has(ys, x) = ff\}$. Having $has(\tilde{ys}, \tilde{x}) \to ff$ available as auxiliary rule in $\tilde{D}$ allows to apply rule (2). This results in rewriting $[\tilde{y}|delete(\tilde{ys}, \tilde{x})]$ to $[\tilde{y}|\tilde{ys}]$. Now, both sides of the conclusion have become identical.

5.1.2 CONTEXTUAL REWRITING

A more general reduction relation, called contextual rewriting in Ganzinger (1987) and Zhang & Kapur (1988) is defined as follows:

DEFINITION 5.2. $s \to_{R,C} t$ iff $s \equiv_C K[l\sigma]$, $s \geq_{st} K[l\sigma]$, $t = K[r\sigma], \ldots u = v \ldots \Rightarrow l \to r \in R$, and $u\sigma \downarrow_{R,C} v\sigma$. In this case, $\downarrow_{R,C}$ stands for $u\sigma \to^*_{R,C} u'$, $v\sigma \to^*_{R,C} v'$, and $u' = v' \in \equiv_C$. (Again, the least fixpoint of this recursive definition is meant.)

$\to_{R,C}$ is a restricted form of rewriting modulo C. The restriction is that one may not produce larger terms by the $\equiv_C$-steps. Note that $\equiv_C$ is decidable if the equations in C are ground (Ackermann, 1954). This is the case in our application. Rewriting by $\to_{R,C}$ also yields bounded proofs:

PROPOSITION 5.3. 1. *For any two terms s and t, if $\tilde{s} \to_{R,\tilde{C}} \tilde{t}$, then there exists a $(C \Rightarrow s = \min)$-bounded proof $C \vdash_{\emptyset,R} P : s = t$ for $C \Rightarrow s = t$. Furthermore, this proof is even $(C \Rightarrow s \to \min)$-bounded, if $\{s\} > \mathcal{T}(C)$ and if the rule $D \Rightarrow l \to r \in R$ that is used to reduce $\tilde{s}$ into $\tilde{t}$ is not applied at the top of $\tilde{s}$ or $\tilde{s} \gg l$.*

2. *For any two terms s and t, if $\tilde{s} \downarrow_{R,\tilde{C}} \tilde{t}$, then there exists a $(C \Rightarrow s = t)$-bounded proof $C \vdash_{\emptyset,R} P : s = t$ for $C \Rightarrow s = t$.*

Proposition 5.3 is a proper generalization of 5.2. In practice one should use an efficient reduction relation between $\to_{R \cup \tilde{C}}$ and $\to_{R,\tilde{C}}$, where $\tilde{C}$ is the subset of (skolemized) orientable equations in C.

This leads us to the following important technique of eliminating equations by contextual reduction:

PROPOSITION 5.4. *An equation $C \Rightarrow s = t$ can be eliminated via inference (D), if $\tilde{s} \downarrow_{R,\tilde{C}} \tilde{t}$.*

5.1.3. SUBSUMPTION

PROPOSITION 5.5. *An equation $C \Rightarrow s = t$ can be eliminated via inference (D), if $D \Rightarrow u = v \in E$ and $(D \Rightarrow u = v)\sigma \equiv (C \Rightarrow s = t)$, for a substitution $\sigma \neq \Lambda$.*

PROOF. σ can be a bijective renaming of variables. In this case we can consider $D \Rightarrow u = v$ and $C \Rightarrow s = t$ to be equal and eliminate one of the copies. (The corresponding proof terms are equivalent modulo complexities.) Otherwise, the second component in our complexity measure for equations makes $D\sigma \Rightarrow u\sigma = v\sigma$ to be a $(C \Rightarrow s = t)$-bounded proof of $C \Rightarrow s = t$. By inference rule (D), $(D \Rightarrow u = v)\sigma$ can be deleted.

Subsumption is a particular case in which the equations in E are used to construct simpler proofs in the course of applying inference (D). In particular if E contains non-reductive equations which will never be oriented into a rewrite rule, the use of non-reductive equations for the simplification of other equations might be crucial for the termination of the completion procedure. In practice, a combination of the above equation elimination techniques as established by the Propositions 5.4 and 5.5 will be needed. Theoretically, when trying to eliminate an equation η, one can enumerate all proofs for η and check as to whether they are η-bounded. Whereas the complexity test for not too big proofs can be reasonably efficiently implemented using the termination proof system for rewrite rules, the problem is to efficiently enumerate "sufficiently many" proofs. In section 6 we will give some examples to illustrate these situations.

5.2. TREATMENT OF NON-REDUCTIVE EQUATIONS

Condition (1) in the definition of fairness asks for the simplification of proofs which apply equations. An obvious possibility is to orient the equation into a rewrite rule. However this might not be possible due to the fact that there is a term in the condition which violates the reductivity constraint of any possible orientation. This problem is well-known in the literature (Kaplan, 1984*b*; Ganzinger, 1987; Kounalis & Rusinowitch, 1987; Orejas, 1987).

The approach to be described below uses conditional narrowing on conditions of equations, which, in the case of a confluent R, is a complete procedure for computing the solutions of the condition equations (Hussmann, 1985; Jouannaud & Waldmann, 1986). In most cases, however, this technique by itself will not lead to a terminating completion process unless it is accompanied by sufficiently strong techniques for eliminating equations by bounded proofs, the problem which we have addressed in section 5.1.

DEFINITION 5.6. Let a conditional equation $C \wedge u = v \Rightarrow s = t$ and a conditional rule $D \Rightarrow l \to r$ be given and assume that the variables in the rule and the equation have been renamed such that no common variables occur. Let o be a non-variable occurrence in $u = v$ such that $(u = v)/o$ and l can be unified with a *mgu* σ. Moreover, if $u\sigma > v\sigma$, then o is inside u, and if $v\sigma > u\sigma$, then o is inside v. With these assumptions, $(D \wedge C \wedge (u = v)[o \leftarrow r])\sigma \Rightarrow s\sigma = t\sigma$ is called a *superposition instance* from superposing $D \Rightarrow l \to r$ on $u = v$ in $C \wedge u = v \Rightarrow s = t$.

Let us provide some intuition behind our following formal treatment. Remember we have to provide for the simplification of proofs which apply an equation under a substitution σ such that the proofs for any of its substituted conditions $u = v$ are rewrite proofs. A rewrite proof for $u\sigma = v\sigma$ is either empty (i.e. $u\sigma \equiv v\sigma$) or it must contain some step of rewriting from $u\sigma$ or $v\sigma$.

In the first case, the meta-rule $x = x \to \mathit{true}$ can be superposed on $u = v$. This will delete the condition from the equation. If we add to the set of equations this superposition instance, using the new equation with one condition less for proving $s\sigma = t\sigma$ is less complex.

In the second case, the first rewrite step can either be inside σ or the redex overlaps with a non-variable position in, say, u. In the first of these remaining cases, rewriting first σ to σ' and then using the same equation with the reduced substitution σ' to prove $s\sigma' = t\sigma'$, followed by unfolding σ' back into σ is less complex. This is mainly because the application of an equation using a smaller substitution becomes less complex. The last case, finally, can be simplified if the superposition instance that corresponds to the overlap of the redex of the first rewrite step with u is added to the equations. In this instance, u has been narrowed to z, and $u\sigma \equiv u\tau\tau' \to z\tau'$. Applying the given equation under σ can hence be replaced by applying the superposition instance under substitution τ'. This is less complex, mainly because $u\sigma > z\tau'$.

LEMMA 5.7. *Assume E and R to be given sets of equations and rules, respectively. Let $C \wedge u = v \Rightarrow s = t$ be an equation in E. Moreover assume that E contains all instances of the equation generated by superposing each rule in $R \cup \{x = x \to \mathit{true}\}$† on the condition*

† After superposition with $x = x \to \mathit{true}$ we delete *true* from the condition. = is assumed not to be an operator of the given signature Σ.

$u = v$ in the given equation. If

$$P = \frac{Q}{s\sigma \leftrightarrow_{C \wedge u = v \Rightarrow s = t} t\sigma}$$

is a proof in $T_{\mathscr{E}(E,R)}$ with Q a tuple of rewrite proofs for the condition equations in $(C \wedge u = v)\sigma$, then $T_{\mathscr{E}(E,R)}$ also contains a proof of $s\sigma = t\sigma$ which is simpler than P.

PROOF. We assume P to be of the required form. Without loss of generality we assume $C = \{c = d\}$. $Q_{c=d}$ and $Q_{u=v}$ are assumed to be the rewrite proofs for $c\sigma = d\sigma$ and $u\sigma = v\sigma$, respectively. There are three cases to consider:

(1) $u\sigma \equiv v\sigma$:

Then, $\sigma = \tau\tau'$, with $\tau = mgu(u, v)$. From superposition with $x = x \rightarrow true$ we have $c\tau = d\tau \Rightarrow s\tau = t\tau \in E$. The proof

$$\frac{Q_{c=d}}{s\sigma \leftrightarrow_{c\tau = d\tau \Rightarrow s\tau = t\tau} t\sigma}$$

is less complex than P as the used substitution instance of the new equation has one condition less than the corresponding instance of the original one.

(2) *The first step of rewriting in $Q_{u=v}$ occurs at or below an occurrence p of a variable x in u:*

Then, $Q_{u=v} = Q''; Q'$, with Q'' representing this rewrite step $u\sigma/p \rightarrow u'$. Let $\sigma' = \sigma[x \leftarrow u']$. A simpler proof takes the form $P_1; P_2; P_3$. P_1 reduces $s\sigma$ to $s\sigma'$ by applying the rewrite step $u\sigma/p \rightarrow u'$ at each occurrence of x in s. P_2 applies the equation with the reduced substitution, requiring an adaptation of the proofs of the original condition $c\sigma = d\sigma \wedge u\sigma = v\sigma$ to the reduced condition instance $c\sigma' = d\sigma' \wedge u\sigma' = v\sigma'$. More precisely,

$$P_2 = \frac{c\sigma' \leftarrow^* c\sigma;\ Q_{c=d};\ d\sigma \rightarrow^* d\sigma'\ u\sigma' \leftarrow^* u\sigma;\ Q_{u=v};\ v\sigma \rightarrow^* v\sigma'}{s\sigma' \leftrightarrow_{c=d \wedge u=v \Rightarrow s=t} t\sigma'}.$$

P_3, finally, is the inverse of the reduction of $t\sigma$ to $t\sigma'$. The application of the equation under the reduced substitution σ' in P_2 is less complex than the original application under σ. (We have $u\sigma > u\sigma'$.) Moreover, any of the rewrite steps in $Q_{u=v}$, $Q_{c=d}$ and in σ to σ' have redexes smaller or equal to a term in $\mathscr{T}(c = d \wedge u = v \Rightarrow s = t)\sigma$. Hence, Q is bounded by the original application of the equation in P.

(3) *The rewrite proof $Q_{u=v}$ is of form $U_1; U_2; V$, where $U_1 = u\sigma \rightarrow_R u[p \leftarrow r]\tau\tau'$, $U_2 = u[p \leftarrow r]\tau\tau' \rightarrow_R^* w$ and $V = w \leftarrow_R^* v\sigma$, with $u\sigma = u[p \leftarrow l]\tau\tau'$, for p a non-variable occurrence in u*: Hereby we also assume $\tau = mgu(u/p, l)$ and $D \Rightarrow l \rightarrow r$ is the rule for rewriting u at p. Let S be the tuple of rewrite proofs for $D\sigma$ and let $z = u[p \leftarrow r]\tau$. We can also assume that $u\sigma \not< v\sigma$. (Otherwise, we could take as U_1 the first step of rewriting from v and develop the same argument for v which we now develop for u. Note that at least one step of rewriting from $v\sigma$ must exist in any rewrite proof of $u\sigma = v\sigma$ in case $u\sigma < v\sigma$.) Then, $D\tau \wedge (c = d)\tau \wedge z = v\tau \Rightarrow s\tau = t\tau \in E$, according to the assumptions of the lemma. The proof

$$\frac{S, Q_{c=d}, (U_2; V)}{s\sigma \leftrightarrow_{D\tau \wedge c\tau = d\tau \wedge z = v\tau \Rightarrow s\tau = t\tau} t\sigma}$$

is less complex than P, as the following inequalities can easily be proven: It is $\{u\sigma\} > \mathcal{T}(D\sigma)$ because of the reductivity of $D \Rightarrow l \to r$. For the same reason we have $u\sigma > z\tau'$. Hence,

$$\{c\sigma, d\sigma, u\sigma, v\sigma, s\sigma, t\sigma\} > \mathcal{T}(D\sigma) \cup \{c\sigma, d\sigma, z\tau', v\sigma, s\sigma, t\sigma\}.$$

From this and the fact that S is a tuple of rewrite proofs in terms all smaller than $s\sigma$ we conclude that this proof is in fact less complex than P.

The lemma proves that superposing rules on a condition of an equation is an alternative to orienting the equation into a rule. This gives us a particularly interesting refinement of the general fairness constraint about completion inferences.

Let CP_i denote the set of critical pairs between any two rules in R_i. Let SP_i denote the set of superposition instances of an equation in E_i by all rules in $R_i \cup \{x = x \to \mathit{true}\}$ on one selected condition of the equation. Moreover, let U_i denote the subset of unconditional equations of E_i.

LEMMA 5.8. *A $\mathscr{CC}$-derivation $(E_0, R_0), (E_1, R_1), \ldots$ is fair, if*

1. $\bigcap_{j \geq i} U_j = \emptyset$, *for all* i.
2. *If* $C \Rightarrow s = t \in \bigcap_{j \geq i} CP_j$ *for some* i, *then there exists an index* i' *such that* $C \Rightarrow s = t \in E_{i'}$.
3. *If* $C \Rightarrow s = t \in \bigcap_{j \geq i} SP_j$ *for some* i, *then there exists an index* i' *such that* $C \Rightarrow s = t \in E_{i'}$.

Hence, a completion procedure is fair if it can either eliminate or orient any unconditional equation and if it computes each critical pair between final rules and each superposition instance of final rules on one selected condition of each final equation. The proof of this lemma is an immediate consequence of the Lemmas 4.3 and 5.7.

5.3. THE COMPLETION PROCEDURE

In this section we describe a specific instance of a completion procedure for conditional equations based on the inference rules and techniques described in the previous sections. The procedure uses Huet's (1981) labelling scheme to optimize the computation of critical pairs. In the following, simplification or elimination of an equation or rule means to apply the techniques outlined in section 5.1.

In our procedure we will label equations as either "possibly reductive" or "non-reductive". Equations which are possibly reductive will be considered as candidates for orientation. Non-reductive equations will be superposed on one of their conditions by rewrite rules in the way indicated in Lemma 5.7. The procedure, when applied to an initial set E of equations, consists of the following steps:

1. $E_0 = E$, all equations labelled "possibly reductive", $R_0 = \emptyset$, $i = 0$, $p = 0$.
2. If E_i contains an equation labelled as "possibly reductive" then go to 4.
3. If all rules are marked, stop with *success*. Otherwise, select an unmarked rule in R_i, say with label k. Let CP be the set of all contextual critical pairs $C \Rightarrow c = d$ between rule k and any rule of R_i of label not greater than k. Let SP be the set of superposition instances of the rule on a specific condition† of all non-reductive equations in E_i.

† In this step and in step 6 the same condition must be selected!

Then, $E_{i+1} = E_i \cup CP \cup SP$. Any of the new equations is labelled as "possibly reductive". Let R_{i+1} be the same as R_i, except that rule k is now marked. Set $i := i + 1$ and go to 2.

4. Select an equation $D \Rightarrow c = d$ in E_i labelled as "possibly reductive". Simplify the equation by E_i and R_i. This process yields a simplified equation $C \Rightarrow c' = d'$.
5. If $E_i \cup R_i$ has a $(D \Rightarrow c = d)$-bounded proof of $C \Rightarrow c' = d'$, then $E_{i+1} := E_i - \{D \Rightarrow c = d\}$, $R_{i+1} := R_i$, $i := i + 1$, and go to 2.
6. If $c' > d'$, then let $l := c'$, $r := d'$. If $d' > c'$, then let $l := d'$, $r := c'$. In both cases verify that each term in C is smaller than l. If this is the case, go to 7.
   Otherwise, and if C is empty, stop with *failure*. In the remaining case, label the equation $C \Rightarrow c' = d'$ as "non-reductive". Then compute the set SP of all superpositions of any marked rule in R_i on a particular condition in $C \Rightarrow c' = d'$. The equations in SP are labelled as "possibly reductive". Let $E_{i+1} := E_i \cup SP$, $R_{i+1} := R_i$, $i := i + 1$, and go to 2.
7. Let K be the set of labels k of rules $C_k \Rightarrow l_k \rightarrow r_k \in R_i$ such that l_k can be simplified by $\{C \Rightarrow l \rightarrow r\}$ to, say, l'_k. Then,

$$E_{i+1} := (E_i - \{D \Rightarrow c = d\}) \cup \{C_k \Rightarrow l'_k = r_k | k \in K\},$$

   all new equations labelled "possibly reductive". Increment p by 1. Let

$$R_{i+1} := \{j : C'_j \Rightarrow l_j \rightarrow r'_j | j : C_j \Rightarrow l_j \rightarrow r_j \in R_i, j \notin K\} \cup \{p : C \Rightarrow l \rightarrow r\},$$

   where r'_j and C'_j are obtained from bounded simplification using E_i and $R_i \cup \{C \Rightarrow l \rightarrow r\}$. The rules coming from R_i are marked or unmarked as they were in R_i. The new rule $C \Rightarrow l \rightarrow r$ is unmarked. Increment i by 1 and go to 2.

Some further remarks about this procedure seem to be in order. In step 3 one might perhaps want to immediately drop any critical pair $C \Rightarrow c = d$ the condition of which cannot be satisfied. The fairness requirement does not ask for consideration of critical pairs with unsatisfiable conditions. For the case of a monolithic specification this would formally mean that $C\sigma \not\subset \equiv_{R_i}$, for any substitution σ and index i. Jouannaud & Waldmann have suggested and proved correct a bounded narrowing technique in which equations are kept separately if the satisfiability of their conditions has not yet been proved or disproved. The conditions of these equations are further narrowed in each iteration step of the completion procedure. For the case of specification modules or parametric specifications such a "negation as failure" technique cannot prove anything about the yet unknown actual parameter. This technique is therefore not modular as one has to reconsider any old critical pair upon enrichment of the specification.

In our case, if an equation $C \Rightarrow c = d$ gets deleted by a bounded proof, this proof is independent of any additional signature and axioms. Proof orderings can be extended to the proofs in an enriched signature, if the underlying reduction ordering on $T_\Sigma(X)$ can be extended. The deletion of the equation is therefore also allowed in an enriched system. Otherwise, and if the equation cannot be oriented into a reductive rule, our procedure keeps them as "non-reductive" equations. They are superposed on one of their conditions. If the procedure terminates, the final system still contains all these non-reductive equations. They are useless for proving equations in the current system, but they might become relevant when enriching the specification. Then, any new rule must also be superposed on these equations. However, neither superpositions by old rules nor critical pairs between old rules must be recomputed. Hence, our completion procedure

allows separate completion of specification modules. Upon combining two such specifications only the inferences between axioms of different modules need to be taken into consideration.

In step 6, an equation is oriented into a rule if it is reductive. In practice it sometimes turns out to be useful to consider a reductive equation nevertheless as "non-reductive" and superpose on a condition. This choice may even be crucial to make completion terminate. Hence if termination of completion is the major goal, it is not always good to select the maximal literal for "extended superposition" as suggested in Rusinowitch (1987). One of the advantages of our method is that we may select *any* literal for superposition. For example, if the condition contains an equation *true* = *false*, it should be selected for superposition. Either the specification is inconsistent (of course, we intend *true* to be different to *false*), or there is no superposition possible, and the equation does not generate any further equation. If the selected literal is the consequent, we have called the superposition "critical pair computation". The selection of the consequent for superposition is however restricted to reductive rewrite rules. This causes a failure of our procedure if the equation has no condition and cannot be oriented.

Apart from this possibility of failure, our procedure is hence more efficient and will terminate in many more practical examples compared to the procedure in Kounalis & Rusinowitch (1987). We do not have to superpose on more than one literal of any clause, we have much more freedom in choosing the literal for superposition, and we do not have to superpose equations on equations or rules. Moreover, the techniques for eliminating equations based on proof orderings is much more powerful than what Kounalis & Rusinowitch were able to prove.

Our procedure is correct, as established by the following theorem:

THEOREM 5.9. *If the completion procedure terminates successfully, the final set of rules R is canonical and* $\equiv_E = \equiv_R$.

PROOF. Clearly, the procedure produces a *CC*-derivation. Hence, the correctness of *R* follows from the obvious correctness of any of the inference rules.

It remains to be shown that the *CC*-derivation is fair. The first fairness requirement is satisfied as any equation is eventually either deleted, turned into a rule or superposed on by any final rule. This is exactly what is required to apply Lemma 5.8. (In fact, rules that are later simplified on their condition or on their right side are not superposed again on any old non-reductive equation. That this is correct is proved in a way similar to what we are not proving for the case of critical pair computation.)

We now argue that any peak in a *E*-proof caused by a critical pair between *R*-rules can be simplified. This is true even though critical pairs between two rules ρ_1' and ρ_2' that have been obtained by simplifying rules ρ_1 and ρ_2 using (*DC*), (*SC*) and (*SRR*) are not computed, if the critical pairs between the ρ_1 and ρ_2 have been computed previously. To indicate the proof of this fact, let us assume $\rho_1 = C \Rightarrow l \to r$, $C = \{e_1, \ldots, e_n\}$ and $\rho_2 = D \Rightarrow a \to b$ and let us take a closer look at the particular case in which the right side r of ρ_1 has been simplified to r' in step i of the completion procedure. Then there is a $C \Rightarrow r \to r$-bounded proof $\kappa_1 : e_1, \ldots, \kappa_n : e_n \vdash_{E_i, R_i} P : r = r'$. The critical pair branching

$$\frac{PC\ PD}{r'\sigma\tau \leftarrow_{C \Rightarrow l \to r'} u\tau \to_{D \Rightarrow a \to b} M[b\sigma\tau]}$$

with simplified r', $l\sigma = u = M[a\sigma]$ and arbitrary tuples of rewrite proofs $PC = (P_1, \ldots, P_n)$ and PD of $C\sigma\tau$ and $D\sigma\tau$, respectively, is more complex than the proof of form

$$\sigma\tau(i(P))[P_i/\kappa_i; 1 \leq i \leq n]; \quad \frac{PC\ PD}{c\tau \leftrightarrow_{(C \wedge D)\sigma \Rightarrow c=d} d\tau}$$

using the critical pair $(C \wedge D)\sigma \Rightarrow c = d$ between the unsimplified rules as computed in 3 of some step $j < i$. It holds $\{u\} > \mathcal{T}((C \wedge D)\sigma) \cup \{c, d\}$, $c = r\sigma$, $d = M[b\sigma]$. Also, the operators in any of the proofs P, PC and PD have a complexity less than $(\{u\tau\}, \{l\}, \mathcal{T}(C) \cup \{r'\})$. In the case of PC and PD this property follows from the reductivity of the rules and from the fact that these proofs are assumed to be rewrite proofs, cf. fairness requirement 2.

The proofs of 4.3 and 5.7 exhibit the possibility for further improvements of the completion procedure. If the computation of a critical pair or of a superposition instance of a non-reductive equation is immediately followed by its simplification, we may relax the complexity bounds for these simplification proofs compared to what is in general required for the inference rules (SE), (D), (SC) or (DC).

In the case of a critical pair, the proof of 4.3 shows that the superposition term u of a critical pair is more complex than the complexity of a critical pair application

$$c \leftrightarrow_{(C \wedge D)\sigma \Rightarrow c=d} d$$

When simplifying the critical pair equation $cp = (C\sigma \wedge D\sigma \Rightarrow c = d)$ immediately we may therefore relax the complexity bound for the simplification proof to $c(u \rightarrow \min)$. In other words, cp may be simplified by a $u \rightarrow$ min-bounded proof. As this bound exceeds $c(cp)$, the bound which would have to be used in the general case, the simplification process becomes more powerful. The techniques described in Winkler & Buchberger (1983), Küchlin (1985) and Küchlin (1986) are based upon related ideas. In Bachmair (1987), the concept of critical pair criterion is proposed to serve as a framework for describing such techniques.

In the case of conditional equations, if $C \wedge u = v \Rightarrow s = t$ gets superposed on $u = v$ yielding $D\sigma \wedge C\sigma \wedge z = v\sigma \Rightarrow s\sigma = t\sigma$, we may simplify this superposition instance with $(\mathcal{T}(C\sigma) \cup \{u\sigma, v\sigma, s\sigma, t\sigma\}, \mathcal{T}(C) \cup \{u, v, s, t\}, \varnothing)$ as upper bound for the operators in the simplification proof. Any simplification of $D\sigma \wedge C\sigma \wedge z = v\sigma \Rightarrow s\sigma = t\sigma$ by such a proof also allows for a simpler proof of $C\sigma \wedge u\sigma = v\sigma \Rightarrow s\sigma = t\sigma$.

At the moment our inference rules and the notion of proof ordering is not strong enough to support these techniques directly. In Ganzinger (1988) we have described a completion inference system in which the complexities of equation applications in proofs are not fixed, but rather determined dynamically upon generation of the equation. Some of the subsequent examples will require these extended techniques to achieve termination of the completion process.

6. Examples

The first example is a specification of integers with 0, s, p, $<$ taken from Kaplan (1984*a*).

Initial equations

1 $0 < 0 = \text{false}$
2 $0 < s(0) = \text{true}$
3 $s(x) < y = x < p(y)$
4 $p(x) < y = x < s(y)$
5 $(y < x) = \text{true} \Rightarrow y < s(x) = \text{true}$
6 $(y < x) = \text{false} \Rightarrow y < p(x) = \text{false}$
7 $s(p(x)) = x$
8 $p(s(x)) = x$

The critical pair

$$(y < s(x1)) = \mathit{false} \Rightarrow y < x1 = \mathit{false} \tag{9}$$

is obtained from superposing rules 4 and 6. It will be recognized as a non-reductive equation. The same is true for the critical pair

$$(y < p(x1)) = \mathit{true} \Rightarrow y < x1 = \mathit{true} \tag{10}$$

from 3 and 5. For Eq. (9) the superpositions by rules on its condition are as follows:

$$(y < x) = \mathit{false} \Rightarrow y < p(x) = \mathit{false} \tag{11}$$

from superposing $s(p(x)) \rightarrow x$,

$$(y1 < x1) = \mathit{true} \text{ and } \mathit{true} = \mathit{false} \Rightarrow y1 < x1 = \mathit{false} \tag{12}$$

from superposing $(y < x) = \mathit{true} \Rightarrow y < s(x) \rightarrow \mathit{true}$,

$$\mathit{true} = \mathit{false} \Rightarrow 0 < 0 = \mathit{false} \tag{13}$$

from superposing $0 < s(0) \rightarrow \mathit{true}$,

$$(x < p(s(x1))) = \mathit{false} \Rightarrow x < p(x1) = \mathit{false} \tag{14}$$

from superposing $s(x) < y \rightarrow x < p(y)$, and

$$(x < s(s(x1))) = \mathit{false} \Rightarrow x < s(x1) = \mathit{false} \tag{15}$$

from superposing $p(x) < y \rightarrow x < s(y)$. Equation (11) is reduced by rule (6). Equation (12) has an unsatisfiable condition. This will be detected by considering it as a non-reductive equation and by observing that there are no superpositions on $\mathit{true} = \mathit{false}$. Similarly, Eq. (13) is trivial. To delete Eq. (14) one first reduces the condition to $(x < x1) = \mathit{false}$. Now, rewriting $x < p(x1) = \mathit{false}$ using Eq. (6) and the simplified condition as auxiliary rewrite rule (the variables considered as constants), reduces the conclusion to $\mathit{false} = \mathit{false}$. The Eq. (15) is subsumed by $(y < s(x1)) = \mathit{false} \Rightarrow y < x1 = \mathit{false}$, the equation, from which (15) has been obtained. In a similar way one can prove the convergence of the non-reductive Eq. (10).

In this example, the procedure converges with R consisting of the originally given Eqs. 1–8 oriented from left to right and with E consisting of the two Eqs. (9) and (10). These have, however, been proven irrelevant for the equational theory.

The next example is natural numbers with $\leq$ in which a transitivity and totality axion have been included as non-reductive equations. The initial specification is then

1 $0 \leq x = \text{true}$
2 $s(x) \leq 0 = \text{false}$
3 $s(x) \leq s(y) = x \leq y$
4 $x \leq x = \text{true}$
5 $x \leq s(x) = \text{true}$
6 $(x \leq y) = \text{true} \Rightarrow x \leq s(y) = \text{true}$
7 $(x \leq y) = \text{false} \Rightarrow y \leq x = \text{true}$
8 $(x \leq y) = \text{true}$ and $(y \leq z) = \text{true} \Rightarrow x \leq z = \text{true}$.

In this example, our procedure terminates producing

Final Rules
1 $0 \leq x \rightarrow \text{true}$
2 $s(x) \leq 0 \rightarrow \text{false}$
3 $s(x) \leq s(y) \rightarrow x \leq y$
4 $x \leq x \rightarrow \text{true}$
6 $(x \leq y) = \text{true} \Rightarrow x \leq s(y) \rightarrow \text{true}$
Final Equations
7 $(x \leq y) = \text{false} \Rightarrow y \leq x = \text{true}$
8 $(x \leq y) = \text{true}$ and $(y \leq z) = \text{true} \Rightarrow x \leq z = \text{true}$
9 $(s(x) \leq y1) = \text{true} \Rightarrow x \leq y1 = \text{true}$
10 $(s(y) \leq z) = \text{true}$ and $(x1 \leq y) = \text{true} \Rightarrow x1 \leq z = \text{true}$
11 $(s(y) \leq z) = \text{true}$ and $(x \leq y) = \text{true} \Rightarrow s(x) \leq z = \text{true}$

Equations (10) and (11) are added upon superposing Eq. (8) by the rules 6 and 3, respectively. No further non-trivial equations can be derived from these. Despite the simplicity of this example, to make the procedure terminate is not trivial and requires applications of non-reductive equations for the simplification of other generated equations. For example, the equation

$$(s(x1) \leq y) = \mathit{true} \Rightarrow x1 \leq s(y) = \mathit{true} \tag{17}$$

is generated from superposing rule 6 on the condition of Eq. (9). In this case,

$$s(x1) \leq s(y) = \mathit{true} \Rightarrow x1 \leq s(y) = \mathit{true} \tag{9'}$$

is the substitution instance of 9 which constitutes the complexity bound for simplification proofs. Equation (17) may be eliminated as the proof

$$\dfrac{\dfrac{s(x1) \leq y = \mathit{true}}{x1 \leq y \leftrightarrow_9 \mathit{true}}}{s(x1) \leq y = \mathit{true} \vdash x1 \leq s(y) \rightarrow_6 \mathit{true}}$$

is a (17)-bounded proof of (17). Both the application of the Eq. (9) and the rewrite rule 6 are less complex than 17.

In our actual implementation we first iteratively† add to the condition C of any superposition instance (and critical pair) $C \Rightarrow s = t$ an equation $u\sigma = v\sigma$, if there exists a non-reductive equation $D \Rightarrow u = v$ such that there is a bounded proof of $C \Rightarrow D\sigma$ and such that $C \wedge u\sigma = v\sigma \Rightarrow s = t$ is still less complex than the substitution instance of the equation from which $C \Rightarrow s = t$ has been obtained. In the example, we would transform 17 into

$$(s(x1) \leq y) = \mathit{true} \text{ and } (x1 \leq y) = \mathit{true} \Rightarrow x1 \leq s(y) = \mathit{true} \qquad (17')$$

using Eq. (9) with a trivial proof for the fact that the condition of 9 is implied by the condition of 17. The resulting Eq. (17') is less complex than (9'). Now, rewriting the conclusion using 6 and the second condition as additional rewrite rule (cf. 5.1.1) reduces the equation to the identity.

For another example consider the superposition of Eq. (10) on its first condition by rule (6). The instance of (10) is

$$(s(y1) \leq s(y2)) = \mathit{true} \text{ and } (x1 \leq y1) = \mathit{true} \Rightarrow (x1 \leq s(y2)) = \mathit{true}. \qquad (10')$$

The complexity of (10') is the bound for the simplification proof of the new equation

$$(s(y1) \leq y2) = \mathit{true} \text{ and } \mathit{true} = \mathit{true} \text{ and } (x1 \leq y1) = \mathit{true} \Rightarrow (x1 \leq s(y2)) = \mathit{true}. \qquad (18)$$

This equation may be eliminated as the proof

$$x1 \leq y1 = \mathit{true}, s(y1) \leq y2 = \mathit{true} \vdash \dfrac{\dfrac{\dfrac{s(y1) \leq y2 = \mathit{true}}{x1 \leq y1 = \mathit{true},\ y1 \leq y2 \leftrightarrow_9 \mathit{true}}}{x1 \leq y2 \leftrightarrow_8 \mathit{true}}}{x1 \leq s(y2) \rightarrow_6 \mathit{true}}$$

is bounded by (10'). It is however not bounded by (18) as the applications of the Eq. (9) is not less complex than (18). This demonstrates that the superposition instances of rules or old equations from which new equations are constructed must be used as complexity bounds for alternative proofs to achieve termination in more cases.

Altogether, the rules 1–6 are canonical and the equational theory is the same as the one generated by all rules and Eqs. 1–11.

The last example is total orderings with maximum function, an example which is used in Orejas (1987) to demonstrate the failure of the usual completion procedures. The example also appears in Kounalis & Rusinowitch (1987) to demonstrate the power of their method. The version of the example which we use here is more complex due to the fact that we have also included the transitivity axion.

Input specification:

1 $(x \leq y) = \mathsf{true}$ and $(y \leq z) = \mathsf{true} \Rightarrow (x \leq z) = \mathsf{true}$.
2 $(x \leq x) = \mathsf{true}$.
3 $(x \leq y) = \mathsf{true} \Rightarrow \max(x, y) = y$.
4 $(x \leq y) = \mathsf{false} \Rightarrow \max(x, y) = x$.
5 $(x \leq \max(x, y)) = \mathsf{true}$.
6 $(y \leq \max(x, y)) = \mathsf{true}$.

† Note that there is a danger of non-termination.

System after completion:
Rules
2 $x \leq x \rightarrow \text{true}$
3 $(x \leq y) = \text{true} \Rightarrow \max(x, y) \rightarrow y$
4 $(x \leq y) = \text{false} \Rightarrow \max(x, y) \rightarrow x$
12 $(x2 \leq y) = \text{true} \Rightarrow x2 \leq \max(x, y) \rightarrow \text{true}$
13 $(x \leq y2) = \text{true} \Rightarrow x \leq \max(y2, y) \rightarrow \text{true}$
Equations
1 $(x \leq y) = \text{true}$ and $(y \leq z) = \text{true} \Rightarrow x \leq z = \text{true}$
7 $(x1 \leq y1) = \text{false} \Rightarrow y1 \leq x1 = \text{true}$
10 $(\max(y2, y) \leq z) = \text{true}$ and $(x1 \leq y2) = \text{true} \Rightarrow x1 \leq z = \text{true}$
11 $(\max(x, y) \leq z) = \text{true}$ and $(x1 \leq y) = \text{true} \Rightarrow x1 \leq z = \text{true}$

The totality axiom 7 is obtained as a critical pair. This example is interesting as completion transforms the original Eqs. (5) and (6) which are too weak for deciding the equational theory by reductive rewriting (in the presence of transitivity) into the more general rules 12 and 13. At an intermediate stage, the rules

$x1 \leq \max(x, \max(x2, x1)) \rightarrow \text{true}$
$x1 \leq \max(\max(x2, x1), y) \rightarrow \text{true}$
$x1 \leq \max(x, \max(x1, y1)) \rightarrow \text{true}$
$x1 \leq \max(\max(x1, y1), y) \rightarrow \text{true}$

are generated. These rules generate the non-operational equations

8 $(\max(x1, y) \leq z) = \text{true} \Rightarrow x1 \leq z = \text{true}$
9 $(\max(x, x1) \leq z) = \text{true} \Rightarrow x1 \leq z = \text{true}$

Which are later again eliminated upon elimination of the rules from which they have been generated. This example also demonstrates that the behaviour of the completion process depends very much on the choice of the condition on which a non-operational equation is to be superposed on. If one chooses the second condition of the transitivity axiom for superposition, the procedure generates the crucial rules 12 and 13 directly without producing any of the Eqs. (8)-(11).

7. Conclusions

We have shown that the concept of proof orderings by Bachmair, Dershowitz & Hsiang extends to the conditional equational case. Recursive path orderings on proof terms provide for the required reduction ordering on proofs. We have argued that in the conditional case the term structure of proofs must be exploited to obtain sufficiently strong arguments about the complexity of proofs.

A notion of bounded proof terms for *conditional* equations was introduced as the basis to simplify conditional equations and rules during completion.

We have proved the correctness of superposing rules on an arbitrary condition of an equation as an alternative to orienting that equation into a rule. In particular in the case of generating a non-reductive critical pair this technique might still allow the completion procedure to terminate successfully. We have proved the correctness of a specific variant

of a completion procedure in which these techniques have been built-in. We have demonstrated the usefulness of our techniques on some examples. We have argued that our procedure is more efficient and terminates much more often than the one in Kounalis & Rusinowitch (1987).

We have mentioned before that many of the results of this paper can be obtained as special cases of the results about completion of first-order clauses by Bachmair & Ganzinger (1991). Their proofs, however, are based on very different techniques and at present no practical implementation exists.

Some of the ideas for the treatment of non-reductive equations as presented in this paper were stimulated by discussions at the Workshop on Conditional Term Rewriting, Orsay, July 1987. We are grateful to St. Kaplan and J.-P. Jouannaud for their initiative in planning and organizing this event. Discussions with A. Bockmayr and St. Hölldobler on the subject of paramodulation and narrowing helped to find a bug in a previous proof of Lemma 5.7. The author is grateful to H. Bertling for many discussions on the present concepts, and to R. Schäfers, J. Richter, U. Waldmann and U. Wertz for finding many bugs in previous versions of this paper.

References

Ackermann, W. (1954). Solvable cases of the decision problem. North-Holland.

Bachmair, L. (1987). Proof methods for equational theories. PhD Thesis, University of Illinois, Urbana Champaign.

Bachmair, L. (1989). Proof normalization for resolution and paramodulation. In: *Proc. 3rd Int. Conf. Rewriting Techniques and Applications. Springer LNCS*, **355**, 15–28.

Bachmair, L., Dershowitz, N., Hsiang, J. (1986). Proof orderings for equational proofs. *Proc. LICS* **86**, 346–357.

Bachmair, L., Ganzinger, H. (1990). On restrictions of ordered paramodulation with simplification. In: *Proc. 10th Int. Conf. on Automated Deduction. Springer LNCS*, **449**, 427–441.

Bachmair, L., Ganzinger, H. (1991). Completion of first-order clauses with equality by strict superposition. In: *Proc. Second Int. Workshop on Conditional and Typed Rewriting Systems. Springer LNCS*, to appear.

Bergstra, J., Klop, J. W. (1982). Conditional rewrite rules: confluence and termination. Report IW198/82, Mathematisch Centrum, Amsterdam.

Dershowitz, N. (1987). Termination of rewriting. *J. Symbolic Computation* **3**, 69–116.

Dershowitz, N., Manna, Z. (1979). Proving termination with multiset orderings. *CACM* **22**, 465–476.

Ganzinger, H. (1987). Ground term confluence in parametric conditional equational specifications. In: Proc. STACS 1987, *Springer LNCS* **247**.

Ganzinger, H. (1988). Completion with history-dependent complexities for generated equations. In: (Sannella, Tarlecki, eds.) Proc. Workshop on Abstract Data Types, Gullane, 1987, *Springer LNCS*.

Ganzinger, H., Schäfers, R. (1990). System support for modular order-sorted Horn clause specifications. In: *Proc. IEEE 12th Int. Conf. on Software Engineering, Nice*, pp. 150–163.

Huet, G., Oppen, D. C. (1980). Equations and rewrite rules. A survey. In: (Book, R., ed.), *Formal Languages: Perspectives and Open Problems.* New York: Academic Press, 349–405.

Hsiang, J., Rusinowitch, M. (1987). On word problems in equational theories. In: Int. Coll. on Automata Languages and Programming, *Springer LNCS*.

Huet, G. (1981). A complete proof of the correctness of the Knuth-Bendix completion algorithm. *JCSS* **23**, 11–21.

Hussmann, H. (1985). Unification in conditional-equational theories. In: Proc. Eurocal 1985, *Springer LNCS* **204**, 543–553.

Jouannaud, J. P., Waldmann, B. (1986). Reductive conditional term rewriting systems. In: *Proc. 3rd TC2 Working Conference on the Formal Description of Prog. Concepts*, Ebberup, Denmark, Aug. 1986, North-Holland, to appear.

Kaplan, St. (1984*a*). Conditional rewrite rules. *TCS* **33**, 175–193.

Kaplan, St. (1984*b*). Fair conditional term rewrite systems: unification, termination and confluence. Report 194, University de Paris-Sud, Centre d'Orsay.

Kaplan, St. (1987). A compiler for conditional term rewriting. In: Proc. RTA 1987, *Springer LNCS* **256**, 25–41.

Kaplan, St., Remy, J.-L. (1987). Completion algorithms for conditional rewriting systems. MCC Workshop on Resolution of Equations in Algebraic Structures, Austin, May 1987.

Kounalis, E., Rusinowitch, M. (1988). On word problems in Horn logic. In: Proc. First Int. Workshop on Conditional Term Rewriting, Orsay, June 1987, *Springer LNCS*, to appear.

Küchlin, W. (1985). A confluence criterion based on the generalised Newman lemma. In: Proc. Eurocal 1985, *Springer LNCS* **204**, 390–399.

Küchlin, W. (1986). Equational completion by proof transformation. PhD Thesis, Department of Mathematics, ETH Zürich.

Nieuwenhuis, R., Orejas, F. (1991). Clausal Rewriting. In: *Proc. Second Int. Workshop on Conditional and Typed Rewriting Systems. Springer LNCS*, to appear.

Réty, P. (1988). Méthodes d'unification par surréduction. Thesis, University de Nancy 1.

Rusinowitch, M. (1987). Theorem-proving with resolution and superposition: an extension of Knuth and Bendix procedure as a complete set of inference rules. Report 87-R-128, CRIN, Nancy.

Remy, J. L., Zhang, H. (1984). REVEUR 4: A system for validating conditional algebraic specifications of abstract data types. In: *Proc. 6th ECAI*, Pisa 1984, 563–572.

Winkler, F., Buchberger, B. (1983). A criterion for eliminating unnecessary reductions in the Knuth–Bendix algorithm. *Coll. on Algebra, Combinatorics and Logic in Comp. Sci., Györ.*

Zhang, H., Remy, J. L. (1985). Contextual rewriting, Conf. on Rewriting Techniques and Applications, Dijon 1985, *LNCS* **202**.