7-1989

Term Rewriting with Conditionals and Priority Orderings

Chilukuri K. Mohan
Syracuse University, ckmohan@syr.edu

SU-CIS-89-05

Term Rewriting with Conditionals and Priority Orderings

Chilukuri K. Mohan
mohan@top.cis.syr.edu

July, 1989

School of Computer and Information Science
Syracuse University
Suite 4-116
Center for Science and Technology
Syracuse, New York 13244-4100

Term Rewriting with Conditionals and Priority Orderings

Chilukuri K. Mohan
School of Computer and Information Science
Syracuse University
Syracuse, NY 13244-4100
U.S.A.
315-443-2322 mohan@top.cis.syr.edu

{July 6, 1989}

Abstract

Conditional rewriting and priority rewriting are two recent generalizations of term rewriting systems. In the former, each rewrite rule is accompanied by an antecedent which must be shown to hold before rewriting can occur. In the latter, rewrite rules can be used only in a particular order. We compare these formalisms: neither formalism encompasses the other in a practical sense, but we give restrictions under which priority and conditional rewriting can be equivalent. We combine the two operational mechanisms, obtaining a natural and expressive formalism called Priority Conditional Rewriting Systems (PCRS). PCRS can be used to "fully-define" data type specifications and function specifications. Towards this goal, restrictions are given that encourage modularity of specifications and ensure properties of termination, confluence, and total reducibility of ground terms. A logical semantics for priority conditional rewriting is described, using equational formulas $\mathcal{E}(\Re)$ obtained from the rules in the PCRS $\Re$; we give conditions under which rewriting with PCRS is sound and complete.

1 Introduction

Term rewriting systems (TRS) are sets of oriented equations, which implement equational theories using simple operational mechanisms [HO], [DJ]. The rewriting paradigm has clear, localized declarative semantics, and has applications in the areas of theorem-proving, algebraic specifications and functional/logic programming. Recently, attention has been focused on two extensions to term rewriting:

1. Conditional rewriting, using Conditional TRS (CTRS), where each rewrite rule is accompanied by an antecedent that must be shown to hold before rewriting can occur [KJ], *e.g.*, the following "Equational-Inequational-CTRS" specifies an exponentiating function.

$$\left\{\begin{array}{c} int(0) \rightarrow true \\ int(x) = true \Rightarrow int(succ(x)) \rightarrow true \\ power(x,0) \rightarrow 1 \\ power(1,y) \rightarrow 1 \\ int(x) = true \wedge int(y) = true \Rightarrow power(x, succ(y)) \rightarrow times(x, power(x,y)) \\ y \neq 0 \wedge int(x) \neq true \Rightarrow power(x,y) \rightarrow \perp \\ x \neq 1 \wedge int(y) \neq true \Rightarrow power(x,y) \rightarrow \perp \end{array}\right\}$$

2. Priority rewriting, using Priority rewriting systems (PRS), where certain higher priority rewrite rules can inhibit rewriting by lower priority rules [BBK], *e.g.*, the following PRS specifies a function that distinguishes pure lists from other data types, with $r_1 > r_2$ and $r_3 > r_4 > r_5$ in the priority ordering.

$$\left\{\begin{array}{l} \left.\begin{array}{ll} r_1: & and(true,true) \rightarrow true \\ r_2: & and(x,y) \rightarrow false \end{array}\right| \\ \left.\begin{array}{ll} r_3: & Islist(nil) \rightarrow true \\ r_4: & Islist(cons(x,y)) \rightarrow and(Islist(x), Islist(y)) \\ r_5: & Islist(z) \rightarrow false \end{array}\right| \end{array}\right\}$$

A specification written using one formalism (CTRS or PRS) cannot easily be translated into an equivalent specification in the other formalism in a practical sense; attempts to do so distort the language, the properties of the specification, or the concept of equivalence of rewriting systems. In conditional rewriting, no rewriting occurs if the antecedent of a potentially applicable rule does not hold; in general, this effect cannot be achieved by merely using encoded unconditional definitions of if-then and equality predicates. Conversely, priority rewriting is sometimes not "stable on replacement" *i.e.*, it is possible that p rewrites to q but $M(\cdots p \cdots)$ can only rewrite using a higher priority rule to some term other than $M(\cdots q \cdots)$. Hence no equivalent formulation for such PRS can be obtained using TRS or CTRS, which are stable by definition.

We study special cases and restrictions under which priority and conditional rewriting can be equivalent. More importantly, we amalgamate the two operational mechanisms, obtaining a natural combination called Priority Conditional Rewriting Systems (PCRS). Two alternative definitions are possible for priority rewriting. The first (P-rewriting) makes effective use of priority ordering among rules defining different functions, but is not decidable. The second definition (I-rewriting) uses an innermost-first rewriting strategy and is decidable, but ignores the priority ordering among rules defining different functions.

A logical semantics for priority conditional rewriting is described, using equational formulas $\mathcal{E}(\Re)$ obtained from the rules. In priority rewriting with a rule of lower priority, the assumption that higher priority rules cannot apply implicitly necessitates assuming certain default inequations which are not consequences of the rules themselves. Similarly, in conditional rewrite rules, negated equality literals in the antecedent have to be assumed since they cannot be proved. Thus 'p rewrites to q' implies that $p = q$ holds only in some of the models/E-interpretations of $\mathcal{E}(\Re)$. Hence soundness and completeness are defined with respect to the inequations (Ψ) which must be assumed to permit rewriting with conditional and lower priority rules. When functions are partitioned into defined functions and free constructors ($\in C$), soundness and completeness is naturally defined w.r.t. Ψ_C, the set of inequations between non-unifiable constructor terms.

For example, let $\Re$ be the following PCRS which defines a three-valued equality predicate, augmented by other rules like $type(0) \rightarrow number$ and $type(nil) \rightarrow list$.

$$\Re \equiv \left\{ \left| \begin{array}{rl} & eq(x,x) \rightarrow T \\ type(x) = type(y) \ \Rightarrow & eq(x,y) \rightarrow F \\ & eq(x,y) \rightarrow \bot \end{array} \right. \right\}, \text{ where each rule has higher priority}$$

than the next. Then $\mathcal{E}(\Re)$is $\left(\begin{array}{c} \forall x.[eq(x,x) = T] \wedge \\ \forall x,y.[type(x) \neq type(y) \ \vee \ eq(x,y) = F \ \vee \ x = y] \ \wedge \\ \forall x,y.[eq(x,y) = \bot \ \vee x = y \ \vee \ type(x) = type(y)] \end{array} \right)$.

Using this PCRS, the term $eq(nil, 0)$ would reduce to $\bot$, the 'undefined' truth value, assuming that $type(nil)$ and $type(0)$ evaluate to different constants (*list* and *number*, respectively). Such a reduction implicitly assumes the inequations $0 \neq nil$ and $list \neq number$, which occur in Ψ; it can be shown that priority rewriting with $\Re$ is sound and complete w.r.t. this Ψ, which implies inequations between distinct ground constructor terms.

We give conditions under which rewriting with PCRS is sound and complete, generalizing previous results for (unconditional) PRS [Mo2]. Towards the goal of using PCRS to "fully-define" data type and function specifications, restrictions are given that encourage modularity of specifications and ensure properties of termination, confluence, and total reducibility of ground terms. These restrictions suffice to ensure that each ground term has a unique constructor term reduct, thereby assuring soundness and completeness w.r.t. Ψ_C.

This paper extends, combines, and generalizes the results given earlier for unconditional PRS in [Mo2], and unprioritized CTRS in [MS2]. Section 2 introduces conditional and priority rewriting. In section 3, we compare these two rewriting concepts, giving restrictions which enable translation of PRS to CTRS and vice versa. In section 4, we present alternative definitions of priority conditional rewriting, which conform to the requirements of equational reasoning. Section 5 presents the restrictions on PCRS needed for satisfying useful properties like modularity and ground confluence. In section 6, we give an equational logical semantics for PCRS, and criteria for soundness and completeness which are particularly relevant for constructor-based specifications.

2 Preliminaries

We first review a few recently proposed formalisms for conditional rewriting. Then we define the mechanisms involved in priority rewriting with an equational orientation.

We adopt standard notation and definitions for unexplained terms (see [HO, DJ]). "$\forall.F$", "$\exists.F$" respectively denote that all variables in F are universally or existentially quantified. The scope of a quantifier followed by a period is the longest well-formed formula following the period. "$p \equiv q$" denotes that p and q are syntactically identical. "$\mathcal{V}(t)$" denotes the set of variables in t; a term t is ground if $\mathcal{V}(t)$ is empty. We indicate that p is a subterm of M by writing $M[p]$, and $M[p/q]$ is the term resulting from the replacement of an occurrence of subterm p (in $M[p]$) by q. A substitution is a mapping from variables to terms; if t is a term and σ a substitution, then $t\sigma$ denotes the application of σ to t, *i.e.*, the result of simultaneously replacing all occurrences of variables in t by the terms to which σ maps them. A grounding substitution is one which maps each variable (occurring in the terms to which it is applied) to a ground term. A term t matches s if there is a "matching" substitution σ such that $t \equiv s\sigma$; a term p unifies with q if there is a "unifying" substitution θ such that $p\theta \equiv q\theta$. Each sub-formula L_i is referred to as a disjunct when it occurs in the disjunction $L_1 \vee \cdots \vee L_n$, and as a conjunct when it occurs in the conjunction $L_1 \wedge \cdots \wedge L_m$.

We denote by $p \rightarrow_R q$ that a term p rewrites to q in one step using a rewrite system R, with any relevant operational rewriting mechanism; $p \rightarrow_R^* q$ denotes that p rewrites to q using R in zero or more steps, and we then say that q is a reduct of p. A rewriting system is said to be (ground) confluent if for every (ground) term t, there are converging reduction sequences from every pair of distinct reducts of t. Variables in different rules are generally assumed to be distinct.

2.1 Conditional Rewriting

Definition 1 :

- A *Conditional rewrite rule* is a three-tuple, written $\boxed{antecedent \Rightarrow lhs \rightarrow rhs}$, consisting of a conjunction of literals ("*antecedent*"), and two terms ("*lhs*" and "*rhs*").

- A *Conditional term rewriting system (CTRS)* is a finite set of conditional rewrite rules; unconditional TRS are a special case, containing rules with empty antecedents.
- A term t is reduced to s by the CTRS R containing a rule $C \Rightarrow l \rightarrow r$ if
  - t contains a subterm m (the *redex*) which matches l using a substitution σ,
  - s is the result of replacing in t an occurrence of m by $r\sigma$, and
  - the instantiated antecedent $C\sigma$ is shown to "hold" (differently for different CTRS).

Conditional rewriting became a subject of study beginning with [BDJ], and [KJ] contains a representative sample of papers on the topic. CTRS may be classified depending on what literals are contained in the antecedents of rules, and on how the antecedents are shown to "hold" while reducing terms. Earlier work focused mostly on issues like hierarchical CTRS [Re, Na], completion procedures [Ka1, Ga], and implementation [RZ, Ka2] for CTRS with equations in the antecedents of rules. The features which distinguish recent CTRS formalisms are the respective presence of negated equality literals and 'logical' variables ($\notin lhs$) in the antecedents of rules, which are useful for applications like data type specifications and equational programming. We now identify these principal classes of CTRS, which are to be compared with PRS.

- In Equational-CTRS (E-CTRS) the antecedent of each rule is a conjunction of equations [Ka1], [JW]. E-CTRS need not be hierarchical: well-defined evaluation of antecedents is assured by ensuring that terms in the antecedent are strictly smaller than the *lhs* of the rule in some monotonic, well-founded stable term ordering. The equalities in the antecedent are evaluated by checking whether the terms equated have converging reduction sequences.
- In Equational-Inequational-CTRS (EI-CTRS), the antecedent is a conjunction of equations and inequations [MS2], [Ka3]. This formalism extends E-CTRS, by allowing the inclusion of negated equality ($\neq$) literals; in evaluating antecedents, $p \neq q$ is ASSUMED if p, q are shown to be ground terms whose reduction sequences do not converge.

Example 1 : The following EI-CTRS is such that $remove(x, S)$ reduces to an expression representing $S \setminus x$. The relevant constructors are *empty* and *insert*, and the first argument of the latter may be any type of element.

$$\left(\begin{array}{lll} & remove(x, empty) \rightarrow & empty \\ x = y \Rightarrow & remove(x, insert(y, z)) \rightarrow & remove(x, z) \\ x \neq y \Rightarrow & remove(x, insert(y, z)) \rightarrow & insert(y, remove(x, z)) \end{array}\right)$$

Using this specification, together with the integer specification rules

$$s(p(x)) \rightarrow x, \quad p(s(x)) \rightarrow x,$$

we observe that $remove(s(0), insert(p(s(s(0))),\ z))$ reduces using the second rule to $remove(s(0), z)$, because there is a reduction sequence from $p(s(s(0)))$ to $s(0)$. Use of the third rule (with an inequation in the antecedent) is illustrated in reducing $remove(s(0), insert(p(p(s(0))),\ z))$ to $insert(p(p(s(0))),\ remove(s(0), z))$, because there are no converging reduction sequences from $s(0)$ and $p(p(s(0)))$.

- In Satisfiable-Unsatisfiable-CTRS (SU-CTRS), the antecedent of a rule may contain a conjunction of literals to be proved satisfiable, and another conjunction of literals to be proved unsatisfiable in the relevant equational theory [MS1], [MS3]. This formalism thus admits logical variables as well as negation in the antecedents of rules. For the present, we restrict consideration to cases where equality is the only predicate allowed in the antecedent, and assume that an equality is satisfiable iff its arguments can be "narrowed" to unifiable terms [Hu].

Example 2 : The following SU-CTRS specifies a *duplist* function to verify whether the first argument equals a combination of two copies of the second argument. This specification includes an error-check to screen cases where the first argument is not a non-empty list.

$$\left(\begin{array}{lll} \mathbf{sat}[x = cons(y, y)] & \Rightarrow & duplist(x, y) \rightarrow true \\ \mathbf{sat}[x = cons(u, v)]\ \mathbf{unsat}[x = cons(y, y)] & \Rightarrow & duplist(x, y) \rightarrow false \\ \mathbf{unsat}[x = cons(u, v)] & \Rightarrow & duplist(x, y) \rightarrow error \end{array}\right)$$

With this CTRS, $duplist(cons(a,a),a)$ reduces to $true$ using the first rule,
$duplist(cons(a,b),a)$ reduces to $false$ using the second rule, and
$duplist(a,a)$ reduces to $error$ using the last rule.

2.2 Priority Rewriting

Definition 2 : A Priority Rewrite System (PRS) $[R,>]$ consists of an underlying unconditional TRS R, with a partial ordering $>$ among its rules, called the priority ordering.

PRS were initially proposed in [BBK], and further explored in [Mo2]. In PRS, the priority ordering determines which of many competing rules may be used to reduce a given term. We now contrast alternative definitions of priority rewriting, extracting from the results presented earlier in [Mo2].

Notation: $[R_1,>_1]$ is a sub-PRS of $[R_2,>_2]$ if $R_1 \subset R_2$ and $>_1$ is a restriction of $>_2$ to the rules in R_1. We often use the (possibly subscripted) symbol $\Re$ to denote a prioritized system $[R,>]$. In the examples, the priority ordering is indicated by a downward arrow: each rule is of 'higher' priority than a lower rule to which it is connected by '$\downarrow$'. A maximal priority rule is one such that there is no rule of higher priority in the relevant system; since the priority is a partial order, there may be several maximal priority rules in a PRS. Rules may be labelled to allow convenient reference, as in $r_1 : lhs \rightarrow rhs$, where r_1 is the label.

Definition 3 : A term p is P-reducible, and P-rewrites (or P-reduces) to $p[m/t\sigma]$ using a PRS $\Re$ (abbreviated $p \rightarrow^P_{\Re} p[m/t\sigma]$, with $\rightarrow^{P*}_{\Re}$ denoting reflexive transitive closure) if

- $r : s \rightarrow t$ is a highest priority rule in $\Re$ such that
- p contains a subterm m (the redex), such that $m \equiv s\sigma$ for some substitution σ;
- m is ground if r is not a maximal priority rule, and
- m does not have any P-reducible proper subterms $\{m_i\}$ such that $m_i \rightarrow^{P*}_{\Re} n_i$ and $p[m[\cdots m_i/n_i \cdots]]$ is P-reducible using a rule of higher priority than r.

Example 3 : The following PRS (with rule $r_1 >$ rule r_2) specifies a function eq to checks

whether its two arguments are equal.

$$\left\downarrow \begin{array}{l} r_1 : \ eq(x,x) \rightarrow T \\ r_2 : \ eq(x,y) \rightarrow F \end{array} \right.$$

(a) Here, $eq(eq(b,a),eq(a,b))$ cannot be directly reduced at the outermost symbol using either rule, although it matches with the *lhs* of rule r_2. This is because, as per the last part of the above definition, we must first ensure that the higher priority rule r_1 cannot be used to reduce a reduct of the given term obtained by replacing proper subterms. The reducible proper subterms $eq(b,a)$ and $eq(a,b)$ are first reduced to F using rule r_2, resulting in $eq(F,F)$, which can then be reduced to T using rule r_1.

(b) Similarly, $eq(F,eq(x,x))$ cannot be P-reduced at the outer occurrence of eq by either rule; and the priority ordering $(r_1 > r_2)$ prohibits reduction at the inner occurrence of eq using rule r_2. The only possible reduction replaces $eq(x,x)$ by T using rule r_1; the resulting term $eq(F,T)$ can then be P-reduced to F using rule r_2.

The major problems with the above definition are that:

(a) P-rewriting is not stable on replacement, *e.g.*, if the rule $[f(g(x)) \rightarrow a]$ is of higher priority than $[g(h(x)) \rightarrow b]$ in a PRS, we find that $g(h(a))$ P-reduces to b but $f(g(h(a)))$ has no common reducts with $f(b)$.

(b) P-rewriting is not always decidable, *e.g.*, if $[f(a) \rightarrow a] > [f(x) \rightarrow a] > [c \rightarrow g(c)]$ are three rules in a PRS, then an attempt to P-reduce $f(c)$ to a using the second rule does not terminate, because we must first verify that $f(a)$ cannot be obtained on replacing c (in $f(c)$) by any of its reducts.

Since performing a P-rewrite involves checking whether P-reduction of subterms yields a term P-reducible by a higher rule, the original step of P-rewriting may never finish due to non-termination of the P-rewriting sequences issuing from the subterms. But P-rewriting is decidable if every reduction sequence is finite, and stability upon replacement ceases to be of concern when P-rewriting is confluent. Hence the following result, which is applicable for PRS satisfying certain stringent conditions.

Proposition 1 *:* Let $\mathfrak{R}$ be a PRS whose underlying TRS R is confluent and noetherian.

Then any ground term reducible using R is also P-reducible using $\Re$; and for any p, q, if $p \rightarrow^P_{\Re} q$ then there are converging P-reduction sequences from (every) $M[p]$ and $M[p/q]$.
Proof: Given in the appendix.

We thus obtain a sufficient condition for a confluence property which compensates for the lack of stability on replacement. However, the premises of proposition 1 are rather strong, and are not satisfied by simple and interesting PRS's like that in example 3. This motivates the following alternative definition for rewriting using PRS.

Definition 4 : A term p is I-reducible, and I-rewrites (I-reduces) to q using a PRS $\Re$ if

- p contains a ground subterm (redex) m, no proper subterm of which is I-reducible;
- $\Re$ contains a rule $s \rightarrow t$, such that
- m matches s with a matching substitution σ (*i.e.*, $m \equiv s\sigma$),
- q is the result of replacing an occurrence of m in p by $t\sigma$ (*i.e.*, $q \equiv p[m/t\sigma]$), and
- m is not I-reducible by any rule in $\Re$ of higher priority than $s \rightarrow t$.

Notation: We abbreviate 'p I-rewrites to q using $\Re$' by '$p \rightarrow^I_{\Re} q$'. Where justifiable, we denote both '$p \rightarrow^P_{\Re} q$' and '$p \rightarrow^I_{\Re} q$' commonly by '$p \rightarrow_{\Re} q$' or "priority rewriting". The reflexive transitive closure of priority rewriting is indicated by '$\rightarrow^*_{\Re}$'; if $p \rightarrow^*_{\Re} q$, then we say q is a reduct of p. Since $\mathcal{V}(rhs) \subseteq \mathcal{V}(lhs)$ in each rule, ground terms have only ground reducts.

Example 4 (for the same PRS as in example 3, defining eq):

$$\left| \begin{array}{ll} r_1: & eq(x,x) \rightarrow T \\ r_2: & eq(x,y) \rightarrow F \end{array} \right.$$

This time, in attempting to I-reduce $eq(eq(b,a), eq(a,b))$, the innermost subterms $eq(b,a)$ and $eq(a,b)$ are each reduced to F, before any attempt is made to reduce the bigger term at the outermost occurrence of eq. Then, the resulting term $eq(F, F)$ matches with $eq(x,x)$ and can be I-reduced using rule r_1 to T.

Although I-rewriting is decidable and stable on replacement, non-ground I-reductions are practically banned, to preserve the property of "stability on instantiation" (*i.e.*, if p

rewrites to q, then every instance $p\sigma$ must also rewrite to the corresponding instance $q\sigma$). For instance, it should not be the case that $f(g(h(a)))$ reduces only to $f(c)$ and then to a, [and not to b], whereas $f(g(x))$ reduces to b, as might happen if non-ground I-rewriting were to be allowed using the following PRS:

$$\left(\begin{array}{l} f(g(x)) \to b \\ f(x) \to a \\ g(h(x)) \to c \end{array}\right)$$

But this restriction also prohibits some plausible rewrites; *e.g.*, we cannot I-rewrite $eq(x,x)$ to T. Another problem with I-rewriting is that non-termination of I-reduction sequences can have undesirable consequences: intended reductions may not be reachable, because the innermost reduction strategy is not fair. For example, $f(a)$ cannot be I-reduced to c using

$$\left(\begin{array}{l} f(x) \to c \\ a \to a \end{array}\right)$$

For these reasons, we study both competing definitions of rewriting with PRS. Another reason for not discarding the alternative $\to_{\mathcal{R}}^{P}$ (P-rewriting) concept is that it bears resemblance to a specification mechanism (used in practice) of giving different priorities to rules defining different functions, which is irrelevant in I-rewriting. One of the questions investigated in the latter sections of this paper is: when is the use of such a mechanism meaningful?

3 Comparison of Conditional and Priority Rewriting

We now compare conditional rewriting and priority rewriting, addressing when one formalism can equivalently express specifications written in the other. We are interested in straightforward transformations from one formalism to another (*e.g.*, PRS to EI-CTRS, or vice versa) which preserve the property that a term reduces to another in one formalism iff it does so in the other as well. For instance, the set of all priority rewrites possible using a PRS $[R, >]$ is generally a proper subset of the set of rewrites possible using the underlying TRS R, hence we would not say that R is equivalent to $[R, >]$. For a study of analogous equivalence

concepts in logic programming, see [Ma]. At the outset, we observe that the definition of equivalence needs to be restricted to ground term reductions, since the definitions of priority rewriting have largely been restricted to ground reductions.

3.1 PRS vs. EI-CTRS

At first sight, PRS appear to be more general than E-CTRS and EI-CTRS, since the equality predicate can be expressed using a PRS, and priority rewriting implicitly uses negated preconditions in rules of lower priority. For instance, the PRS in example 3 defining the eq predicate is 'equivalent' to the unordered EI-CTRS

$$\{eq(x,x) \to T,\ \ x \neq y \ \Rightarrow\ eq(x,y) \to F\}.$$

By using the corresponding PRS, together with a PRS defining an 'if' predicate, it appears possible to avoid conditionals in rules altogether. For example, a conditional rule like

$$r_c: \qquad p_1 = q_1 \wedge \cdots \wedge p_m = q_m \wedge t_1 \neq s_1 \wedge \cdots \wedge t_n \neq s_n \Rightarrow \lambda \to \rho$$

can be simulated by the following PRS, with each rule r_i having a higher priority than r_{i+1}:

$$\begin{array}{ll} r_1 : not(T) \to F & r_5 : neq(x,y) \to not(eq(x,y)) \\ r_2 : not(F) \to T & r_6 : and(T,T) \to T \\ r_3 : eq(x,x) \to T & r_7 : and(x,y) \to F \\ r_4 : eq(x,y) \to F & r_8 : if(T,x) \to x \end{array}$$

$$r_9 : \lambda \to if(and[eq(p_1,q_1), and(eq(p_2,q_2) \cdots and(neq(t_1,s_1),\cdots,neq(t_n,s_n)\cdots)], \rho)$$

However, this is not an accurate translation of the original conditional rewrite rule: the last (least priority) rule r_9 would apply (λ can always be rewritten), even when the antecedent (if-condition) does not hold. Whereas we would <u>not</u> reduce a term using the corresponding conditional rewrite rule r_c, the PRS yields an irreducible term "$if(\cdots,\rho)$". Superficially, it appears that this problem can be overcome by using a 3-ary $if-then-else$ predicate "$if3$" [instead of just an $if-then$ as above], and replacing rules r_8, r_9 in the above specification

by the following sub-PCRS (again with each $r_i > r_{i+1}$):

$$r_{10}: \ if3(T,x,y) \rightarrow x$$

$$r_{11}: \ if3(z,x,y) \rightarrow y$$

$$r_{12}: \lambda \rightarrow if3(and[eq(p_1,q_1), and(eq(p_2,q_2)\cdots and(neq(t_1,s_1),\cdots,neq(t_n,s_n)\cdots)], \rho, \lambda)$$

But this 'solution' introduces a new problem: r_{12} is a non-noetherian rule which leads to non-terminating rewrite sequences, since the *lhs* of the rule (λ) is a proper subterm of the *rhs*. The above proposal (translation of CTRS to PRS using "$if3$") should be rejected, because it does not preserve the computational behavior (properties like confluence and termination) of the rewrite system.

Another important problem with such a translation is that a "partial" CTRS specification with a negative literal in the antecedent of a rewrite rule cannot be expressed using a PRS in a straightforward way (*e.g.*, $x \neq y \Rightarrow f(x,y) \rightarrow t$). We may use

$$\left| \begin{array}{l} f(x,x) \rightarrow f(x,x) \\ f(x,y) \rightarrow t \end{array} \right.$$

to achieve this effect (when the arguments of f are equal), but using this PRS leads to non-terminating reduction sequences, whereas $f(x,x)$ is just not reducible in the supposedly "equivalent" CTRS. Hence EI-CTRS are not accurately translatable to PRS in general. However, the following result holds for well-behaved EI-CTRS, giving PRS which are equivalent in a certain sense (of having the same normal form), over an extended language with several new "hidden functions". The new PRS specification is much less intuitive and understandable than the translated EI-CTRS, and it is also not the case that single EI-reduction steps correspond to single priority rewriting steps after the transformation.

Proposition 2 : If an EI-CTRS R satisfies the property that every reduction sequence from each ground term t is finite and yields the same irreducible constructor term $\mathcal{N}_R(t)$, then a PRS $\mathcal{P}(R)$ can be constructed such that priority reduction sequences from any ground term t using $\mathcal{P}(R)$ terminate yielding the same normal form $\mathcal{N}_R(t)$ as that produced using R.

Proof: See Appendix.

Conversely, priority rewrite systems cannot (in general) be translated to equivalent unprioritized EI-CTRS. For example, the following PRS

$$\left\downarrow \begin{array}{l} f(c(x,y)) \rightarrow s \\ \qquad f(z) \rightarrow t \end{array} \right.$$

cannot be easily specified using an EI-CTRS, since we cannot have a rule to help us check and decide whether or not $\exists x, y.[c(x,y) = p]$ and accordingly reduce $f(p)$. This is because, traditionally, 'logical' variables not occurring in the *lhs* of a conditional rule are not allowed to occur in the antecedent.

Another problem occurs when non-terminating rewriting I-sequences exist, *e.g.*, using $\{f(x) \rightarrow c, \quad a \rightarrow a\}$, where $f(a)$ cannot be I-reduced to c since the subterm a repeatedly I-reduces to itself. In principle, it is not possible to determine when such non-terminating reduction sequences occur so as to prevent reduction using the "equivalent" CTRS obtained by translating such a PRS. However, PRS which satisfy restrictions given in the following result can be translated into equivalent EI-CTRS.

Proposition 3 : A PRS can be translated into an equivalent EI-CTRS if

- all reduction sequences finitely terminate, and
- whenever we have $[r_1 : f(\overline{s}) \rightarrow p] \quad > \quad [r_2 : g(\overline{t}) \rightarrow q]$ in the priority ordering for any two rules, we have $f \equiv g$, and the tuple $\overline{s}$ contains only variables and ground terms.

Proof: See Appendix.

3.2 PRS vs. SU-CTRS

As in the case of EI-CTRS, there is no obvious way to use PRS to express partial SU-CTRS specifications or to do all the necessary antecedent-checking. In addition, SU-CTRS allow checking the satisfiability of arbitrary equations, whereas priority rewriting can only perform pattern-matching on terms, with no satisfiability-checking mechanisms like narrowing. However, neither does the SU-CTRS formalism encompass priority systems, although SU-CTRS allow logical variables as well as negated literals in the antecedents of rules. Since EI-CTRS can be considered to be a special case of SU-CTRS, the restrictions of the previous proposi-

tion are automatically sufficient to ensure the translatability of PRS to SU-CTRS. We now consider the possibility of relaxing those restrictions.

In the case of P-rewriting, it is obvious that lack of stability on replacement denies the existence of any equivalent unprioritized SU-CTRS, because SU-rewriting is stable on replacement. For example, note that c P-reduces to d whereas $f(c)$ does not P-reduce to $f(d)$, using the following PRS:

$$\left|\begin{array}{l} f(x) \rightarrow b \\ c \rightarrow d \end{array}\right.$$

In any SU-CTRS, if c reduces to d, it is true by definition that $f(c)$ can reduce to $f(d)$.

In the case of I-rewriting, the above problem does not exist, due to stability on replacement; hence it may be possible to construct equivalent SU-CTRS as outlined below. However, the existence of non-terminating I-rewriting sequences causes the same problem as that for the translation of PRS to EI-CTRS. If a PRS has rules like

$$\left|\begin{array}{l} f(t_1, \cdots, t_n) \rightarrow t \\ f(s_1, \cdots, s_n) \rightarrow s \end{array}\right.$$

then the corresponding SU-CTRS (obtained by transforming the above) would have rules like the following, after first renaming variables in different rules in the PRS to be distinct, and where "*tuple*" is a new *n-ary* function symbol which may be considered a constructor for practical purposes (two tuples are equal iff their arguments are pairwise equal).

$$\left(\begin{array}{rl} & f(t_1, \cdots, t_n) \rightarrow t \\ \text{unsat}[tuple(t_1, \cdots, t_n) = tuple(s_1, \cdots, s_n)] \Rightarrow & f(s_1, \cdots, s_n) \rightarrow s \end{array}\right)$$

Although such a transformation is permitted by the SU-CTRS formalism, non-termination often results due to the narrowing mechanism used to check whether an equation is [un]satisfiable in the theory being described by the SU-CTRS in question. For example, $f(h(a,b))$ clearly I-reduces to s using the PRS

$$\left(\begin{array}{c} g(cons(x,y)) \rightarrow g(y) \\ \left|\begin{array}{c} f(g(z)) \rightarrow t \\ f(h(x,y)) \rightarrow s \end{array}\right. \end{array}\right)$$

but the transformed SU-CTRS is (with no "*tuple*" operator since f is unary):

$$\left(\begin{array}{rl} & g(cons(x,y)) \rightarrow g(y) \\ & f(g(z)) \rightarrow t \\ \text{unsat}[g(z) = h(x,y)] \Rightarrow & f(h(x,y)) \rightarrow s \end{array}\right)$$

Hence SU-reducing $f(h(a,b))$ leads to an attempt to prove that $g(z) = h(a,b)$ is unsatisfiable. As per currently known techniques for checking satisfiability in an equational theory [Hu], this has to be done by showing that $g(z)$ and $h(a,b)$ cannot be narrowed to unifiable terms. Unfortunately, an infinite narrowing sequence ensues from the first rule: $g(z) \rightsquigarrow g(z_1) \rightsquigarrow g(z_2) \rightsquigarrow \cdots$, since each $g(z_i)$ narrows to $g(z_{i+1})$ using a narrowing substitution which maps z_i to $cons(x_{i+1}, z_{i+1})$. So the reduction from $f(h(a,b))$ to s using the last conditional rule is never completed.

This illustrates that the ability to transform a PRS to an SU-CTRS does not assure that every possible priority rewrite ($\rightarrow^P$ or $\rightarrow^I$) is reachable by SU-rewriting using the 'equivalent' SU-CTRS. However, it is conceivable that a different method for rewriting using SU-CTRS will overcome the non-termination problem. Restrictions that circumvent the existence of infinite narrowing sequences suffice to help construct SU-CTRS equivalent to a given PRS, following the schema outlined above.

Proposition 4 : Let $\Re$ be a PRS such that all I-reduction sequences finitely terminate, and only non-narrowable terms occur as the arguments of the *lhs* in each non-maximal priority rule. Then there is an equivalent SU-CTRS $\mathcal{S}(\Re)$ such that each ground term has the same irreducible reducts via SU-reduction using $\mathcal{S}(\Re)$ and via I-reduction using $\Re$.

The bottomline of the above comparisons is as follows: priority and conditional rewriting mechanisms do not subsume each other, and direct translations do not appear to be possible, unless we introduce non-terminating rewrite rules, hidden functions, new normal forms, and other unnatural mechanisms which tend to destroy the clarity and utility of the specification. Permitting ourselves both the conditional and priority rewriting mechanisms allows writing specifications in an easier manner than if we try to avoid the features of either mechanism. This motivates combining the two mechanisms, which we explore in the next section.

4 Conditional Rewriting with Priorities

We propose to enhance conciseness of specification by incorporating both priorities and conditionals into a new formalism, *viz., Priority Conditional Rewriting Systems (PCRS).* The associated rewriting mechanism is a combination of priority rewriting and conditional rewriting: informally, terms are reduced by the highest priority rule with matching *lhs*, when the antecedent holds. While conditional rules in any CTRS formalism can be priority-ordered, by way of illustration we consider (in what follows) only PCRS with E-CTRS as the underlying rewrite systems. In other words, rewrite rules have only equations in their antecedents, and all variables in each rule must occur in the *lhs* of the rule. This seems adequate in the sense that the "fully-defined" SU-CTRS specifications we have so far encountered in practice can be naturally encoded in this formalism. The handling of '$\neq$' (negation of equality) in EI-CTRS can be mimicked using the sub-PRS of example 3 which defines the '*eq*' predicate; each inequation "$p \neq q$" occurring in the antecedent is replaced by "$eq(p, q) = F$".

Hence antecedents of rules in PCRS need not contain inequations to be handled in any special manner, although this may yield more readable specifications. [A rewrite rule with 'f' as the outermost symbol of its *lhs* is said to define 'f'.] For purposes of P-rewriting using PCRS, we assume that the rules defining '*eq*', '*and*', and other logical predicates have higher priority than all other rules in the PCRS. For clarity, we will assume that each equation "$eq(t_1, t_2) = T$" in the antecedent of a rule is replaced by "$t_1 = t_2$".

Definition 5 A PCRS consists of an underlying CTRS, containing rules of the form

$$r_i : \; p_1^i = q_1^i \wedge \cdots \wedge p_n^i = q_n^i \Rightarrow lhs_i \rightarrow rhs_i$$

which are partially ('priority') ordered. A term p rewrites to q using a PCRS $\Re$ if

- p has a subterm M, and $\Re$ contains a rule $r : \; p_1 = q_1 \wedge \cdots \wedge p_n = q_n \Rightarrow L \rightarrow R$, which is a rule of the highest priority such that
- M matches L with a matching substitution σ (*i.e.*, $M \equiv L\sigma$);
- $q \equiv p[L\sigma \;/\; R\sigma]$;

- there are finitely converging reduction sequences (using $\Re$) from the instantiated terms $(p_i\sigma, q_i\sigma)$ in each literal of the antecedent (*i.e.*, $\forall i \exists t_i . p_i\sigma \rightarrow_{\Re}^{*} t_i \wedge q_i\sigma \rightarrow_{\Re}^{*} t_i$);
- and {we distinguish between the two possible definitions of priority conditional rewriting}
  - I-rewriting ($\rightarrow_{\Re}^{I}$): M is ground, and no proper subterm of M is I-reducible by $\Re$.
  - P-rewriting ($\rightarrow_{\Re}^{P}$): if r is not a maximal priority rule, then M is a ground term, and M contains no proper subterms N_j with reducts N_j' ($N_j \rightarrow_{\Re}^{P*} N_j'$) such that $p[M[\cdots N_j/N_j' \cdots]]$ is P-reducible using a rule $> r$.

The two cases above distinguish between P-rewriting and I-rewriting with PCRS. Note that with the operational definition of I-rewriting ($\rightarrow^{I}$), the priority ordering among rules defining different functions is irrelevant, and we may consider w.l.o.g. the restriction of the priority ordering to rules defining the same function. For example, even if $f(s_1) \rightarrow t_1$ is given to be a rule of higher priority than $g(s_2) \rightarrow t_2$, the choice of rule used for I-rewriting a term containing f and g is determined exclusively on whether the f-term occurs as a subterm of the g-term or vice versa, and not on the relative priority ordering among the two rules defining the different functions. We will consider only PCRS such that there are no redundant rules or orderings among rules. For example, we will assume that there is no rule which will never be fired because its *lhs* and *antecedent* are 'covered' and implied by those of a higher priority rule.

Example 5 : PCRS $\Re$ below defines a 3-valued equality predicate, with $r_5 > r_6 > r_7$.

$$\left\{ \begin{array}{lll} r_1: & & type(0) \rightarrow number \\ r_2: & type(x) = number \Rightarrow & type(s(x)) \rightarrow number \\ r_3: & & type(nil) \rightarrow list \\ r_4: & type(x) = list \wedge type(y) = list \Rightarrow & type(cons(x,y)) \rightarrow list \\ \left| \begin{array}{lll} r_5: & & eq3(x,x) \rightarrow T \\ r_6: & type(x) = type(y) \Rightarrow & eq3(x,y) \rightarrow F \\ r_7: & & eq3(x,y) \rightarrow \perp \end{array} \right. & & \end{array} \right\}.$$

To I-reduce a term like $eq3(0, nil)$ using $\Re$, we first check whether any of its proper subterms are I-reducible. Since 0 and *nil* do not match with the *lhs* of any rule $\in \Re$, we

proceed to find rules whose *lhs*'s can be matched with $eq3(0, nil)$. The first four rules do not apply because they define a function symbol other than $eq3$, and r_5 also does not apply because 0 and *nil* cannot both be bound to the same variable x in the *lhs* of r_5. The *lhs* of r_6 matches, with the substitution mapping x to 0 and y to *nil*. For reduction to occur using this rule, we must now check whether the correspondingly instantiated antecedent $type(0) = type(nil)$ can be shown to hold. So the terms $type(0)$ and $type(nil)$ are respectively reduced, checking for convergence of their I-reduction sequences. $type(0)$ reduces using the first rule to *number*, and $type(nil)$ similarly reduces to *list* using r_3. Since the resulting terms (*number* and *list*) are not reducible, we conclude that there were no converging reduction sequences from $type(0)$ and $type(nil)$, hence the antecedent of rule r_6 has not been shown to hold, hence r_6 cannot be used to reduce $eq3(0, nil)$. Finally, since the *lhs* of the last rule r_7 can be matched with $eq3(0, nil)$, and the antecedent is empty, reduction <u>can</u> be performed, yielding $\perp$.

Precisely the same result ensues from P-rewriting $eq3(0, nil)$. In trying to P-reduce a bigger term like $eq3(eq3(0, nil), \perp)$, we again find that rules $r_1 - r_4$ cannot be applied because they define a function symbol other than $eq3$. Since the application of rule r_5 requires that the arguments of $eq3$ be identical (they must match the same variable x), even r_5 cannot be applied yet to the term either at the outermost level or at the subterm $eq3(0, nil)$. For application of rule r_6, we need to consider the possibility of reduction at both occurrences of $eq3$. As mentioned above, $eq3(0, nil)$ cannot be P-reduced using r_6 because its instantiated antecedent does not hold. To examine whether r_6 can P-reduce $eq3(eq3(0, nil), \perp)$ at the outer occurrence of $eq3$, with the substitution mapping x to $eq3(0, nil)$ and y to $\perp$, we need to check whether the corresponding antecedent $type(eq3(0, nil)) = type(\perp)$ holds, and also ensure that P-reduction of any proper subterm of $eq3(eq3(0, nil), \perp)$ will not result in a reduct which allows reduction by a rule of higher priority than r_6. In this case, the antecedent is shown to hold, because $eq3(0, nil)$ P-reduces to $\perp$ and hence $type(eq3(0, nil))$ has a reduction sequence that converges with $type(\perp)$. But the requirement that inner subterm reduction should not result in reducibility by a higher priority rule is not satisfied; r_6 cannot be used

for P-reduction, due to P-reducibility of $eq3(0, nil)$ to $\perp$, and P-reducibility of the term $eq3(\perp, \perp)$ [obtained by substituting the reduct of the reducible proper subterm] using higher priority rule r_5. So we are left to examine P-reducibility at both occurrences of $eq3$ using the last rule r_7. But P-reduction by r_7 at the outer occurrence of $eq3$ is prohibited for precisely the same reason as above, *i.e.*, because potential reduction of a subterm can make the result reducible using the higher priority rule r_5. Hence the only possible P-reduction is that obtained by P-reducing the inner term $eq3(0, nil)$ to $\perp$ using r_7, giving $eq3(\perp, \perp)$ as a reduct of $eq3(eq3(0, nil), \perp)$. This reduct can finally be reduced to T using rule r_5, since the arguments of $eq3$ are now identical.

5 Properties of PCRS

We now explore certain desirable properties of ground rewriting using PCRS, in a framework where functions are partitioned into defined functions ($\in D$) and constructors ($\in C$). We say that a rule whose *lhs* has f as outermost symbol defines f; a constructor term is one in which defined functions do not occur; defined functions occur in non-constructor terms. As in the case of unprioritized conditional specifications [MS1], termination of ground reduction, ground confluence and reducibility of every ground non-constructor term are sufficient to ensure that each ground term has a unique constructor term reduct, thereby satisfying the requirements for soundness and completeness of priority rewriting using such PCRS [discussed in a later section].

Finite termination of all reduction sequences using the underlying CTRS is a sufficient condition guaranteeing that priority rewriting also terminates. Methods to show termination of rewriting using E-CTRS extend well-known techniques for TRS [De], and rely on proving that there is a well-founded, stable (on instantiation and replacement) ordering among terms, in which every instance of the *lhs* of each rule is greater than the corresponding instance of the *rhs* and each term in the antecedent of that rule [Ka1]. We note that this is not a

necessary condition for priority rewriting to terminate; for example, priority rewriting using

$$\left\{ \left| \begin{array}{l} p \rightarrow q \\ p \rightarrow p \end{array} \right. \right\}$$

terminates trivially, since the first rule is of higher priority than the second, although the underlying TRS is not noetherian. The following gives an appropriate modification to the termination criterion; the proof is straightforward.

Proposition 5 : Ground priority rewriting with a PCRS terminates if there is a well-founded stable term ordering "$\succ$" such that for every rule $c \Rightarrow l \rightarrow r$ and every grounding substitution σ, we have

either $l\sigma \succ r\sigma$ and $l\sigma \succ$ each term in $c\sigma$;

or it can be shown that $c\sigma$ cannot hold (*e.g.*, because $c\sigma$ contains $p = q$ where p and q are distinct irreducible terms);

or there is a higher priority rule $C \Rightarrow L \rightarrow R$ such that $l\sigma$ is an instance of L, and $C\sigma$ can be shown to hold.

"Confluence" is an issue of central importance in rewriting systems and their applications, if rewriting is to have an equational interpretation, since the equality of terms is to be verified by checking that they have common reducts. A rewriting system is said to be (ground) confluent if for every (ground) term t, there are converging reduction sequences from every pair of distinct reducts of t. Otherwise, there is no easy way to prove (by repeated reduction alone) that the equality of two given terms is entailed by the equational theory which the rewriting system purports to describe.

Ground confluence of PCRS's can be assured by avoiding 'top-level' overlaps between *lhs*'s of rules, giving higher priority to one member of each subset of rules with unifiable *lhs*'s, and ensuring that non-unifiable *lhs*'s of rules cannot be equal in the theory described by the PCRS. These conditions also enforce modularity of function definitions and specifications in a natural way. For modularity, a PCRS specification should consist of disjoint sub-PCRS's separately defining each non-constructor for all possible constructor term arguments.

Definition 6 : A Definitional PCRS (or DPCRS) is a PCRS in which the *lhs* of each rule

is of the form "$f(\tau)$", where f is not a constructor and τ is a tuple of constructor terms. This does not prohibit the antecedents of rules from containing non-constructor terms.

The sets of ground irreducible terms ("normal forms") obtained using a DPCRS $\Re$ coincide for both definitions $(\rightarrow^I, \rightarrow^P)$ of priority rewriting, if $\Re$ satisfies the condition for ground termination. This is true even though non-constructor terms occur in the antecedent of a rule, as long as $\mathcal{V}(antecedent) \subseteq \mathcal{V}(lhs)$. We observe that such a rule $[x = g(y) \Rightarrow f(x,y) \rightarrow t]$ behaves differently from $[f(g(y),y) \rightarrow t]$; in reducing $f(t_1,t_2)$, the former checks for equality of t_1 and $g(t_2)$ via convergence of reduction sequences, whereas the latter checks for the syntactic identity of t_1 and $g(t_2)$. Hence the former (conditional) rule will in general be applicable to many more terms than the latter (unconditional) rule.

Definition 7 : A pair of terms $f(s_1,\cdots,s_n)$, $g(t_1,\cdots,t_m)$ are constructor-wise distinct (and can hence be considered inequal) if

- f and g are distinct constructors; or
- f and g are identical constructors, but $\exists i$. [s_i and t_i are constructor-wise distinct].

For example, if *nil* and *cons* are constructors of arity 0 and 2 respectively, then the following pairs of terms are constructor-wise distinct:

$$cons(x,\ nil) \quad \text{and} \quad cons(nil,\ cons(x, f(y))),$$
$$cons(f(x),\ cons(x,\ nil)) \quad \text{and} \quad cons(f(x),\ cons(nil,\ cons(y,z))).$$

But none of the following pairs of terms are constructor-wise distinct:

$$nil \quad \text{and} \quad f(x),$$
$$f(nil) \quad \text{and} \quad g(cons(x,y)),$$
$$cons(x,y) \quad \text{and} \quad cons(nil,\ cons(u,v)),$$
$$cons(x,\ nil) \quad \text{and} \quad cons(y, f(cons(u,v))).$$

Intuitively, from each of the latter (non-constructor-wise distinct) pairs, it is possible that suitable replacements of non-constructor terms by constructor terms may yield a unifiable pair. If two terms are constructor-wise distinct, their inequality is entailed by axioms which

assert that non-unifiable constructor terms are inequal. A more complex procedure generalizing this condition can be derived from a result in [MSK], which shows how to prove the inequational consequences of a given set of inequations.

Proposition 6 A DPCRS $\Re$ is ground confluent if the following premise is satisfied.

(Premise:) For every pair of rules $r_1 : C_1 \Rightarrow L_1 \rightarrow R_1$, $r_2 : C_2 \Rightarrow L_2 \rightarrow R_2$ whose *lhs*'s unify with m.g.u. σ (*i.e.*, $L_1\sigma \equiv L_2\sigma$), one of the following conditions holds:

- $r_1 > r_2$ or $r_2 > r_1$ in the priority ordering;
- there is another rule $[r : C \Rightarrow L \rightarrow R]$ of higher priority than r_1 (or r_2), such that $\exists\theta.[L_1\sigma \equiv L\theta$ and terms equated by literals in $C\theta$ are identical];
- $R_1\sigma$ and $R_2\sigma$ are identical;
- $\exists eq(p,q) = F \in C_1 \cup C_2$ such that $p\sigma, q\sigma$ are identical;
- $\exists p = q \ \in C_1 \cup C_2$ such that $p\sigma$ and $q\sigma$ are constructor-wise distinct;
- $\exists p = q$ as well as $eq(P,Q) = F \ \in C_1 \cup C_2$ such that $P\sigma, Q\sigma$ are identical upto mutual replacement of some occurrences of their subterms $p\sigma, q\sigma$;
- $C_1 \cup C_2$ contains $p = q$ as well as $P = Q$ (or $Q = P$), such that $P\sigma, p\sigma$ are identical, but $Q\sigma, q\sigma$ are constructor-wise distinct.

Proof: Let $\Re$ be a DPCRS which satisfies the above premise. First consider I-rewriting, where we can focus on top-level reductions, since proper subterms must be I-reduced first, and I-reductions at disjoint subterms must commute. Let $P \rightarrow^I_{\Re} Q_1$ using rule $r_1 : C_1 \Rightarrow L_1 \rightarrow R_1$ and $P \rightarrow^I_{\Re} Q_2$ using rule $r_2 : C_2 \Rightarrow L_2 \rightarrow R_2$, where the two rules have different variables. L_1 and L_2 must be unifiable since there are substitutions θ_1 and θ_2 such that $L_1\theta_1 \equiv P \equiv L_2\theta_2$. Let σ be the most general unifier of L_1 and L_2. Let σ_1, σ_2 be the restrictions of σ to $\mathcal{V}(L_1)$ and $\mathcal{V}(L_2)$ respectively. Then there must be a substitution σ' such that θ_1 is the composition of σ_1 and σ'. We now show that reduction sequences from Q_1, Q_2 must converge if any of the conditions in the premise of the proposition is satisfied.

- If $r_1 > r_2$ or vice versa, such reductions $P \rightarrow^I_{\Re} Q_1$ and $P \rightarrow^I_{\Re} Q_2$ cannot both be possible.

• If there is a rule $r : C \Rightarrow L \rightarrow R$ such that $\exists\theta$. [$L_1\sigma \equiv L\theta$ and $r > r_1$ and literals in $C\theta$ trivially hold], then P would be reducible using r and not r_1, since $P \equiv L\theta\sigma'$, and $C\theta\sigma'$ holds.

• If $R_1\sigma \equiv R_2\sigma$, then $Q_1 \equiv Q_2$, hence the reduction sequences trivially converge.

• If $\exists eq(p,q) = F \in C_1 \cup C_2$ such that $p\sigma \equiv q\sigma$, then neither rule (r_1, r_2) can reduce instances of $L_1\sigma$, $L_2\sigma$, hence neither rule can reduce P.

• If $\exists p = q \in C_1 \cup C_2$ such that $p\sigma$ and $q\sigma$ are constructor-wise distinct, then reduction sequences from $p\theta_1 \equiv p\sigma\sigma'$ and $q\theta_1 \equiv q\sigma\sigma'$ (or $p\theta_2$ and $q\theta_2$) cannot possibly converge hence both rules could not possibly have been used to reduce the redex P.

{The remaining cases are similar to the last two cases above}.

In the case of P-rewriting, we also need to consider the possibility of reductions using r_1, r_2 at different levels in the tree representation of P, *e.g.*, $P \equiv L_1\theta_1$ but $L_2\theta_2 \equiv$ a proper subterm of P. But since $\Re$ is a DPCRS, that reducible proper subterm must occur in P at a position corresponding to that of a variable in L_1. Hence reduction by r_2 leaves the resulting term reducible by r_1, implying convergence of reduction sequences.

• Q.E.D.

Similarly, conditions may be given ensuring that the application of each function is specified by a PCRS for every possible ground constructor term tuple of arguments. For sufficient completeness w.r.t. Ψ_C, in addition to ground termination and confluence, it is also necessary that every ground non-constructor term is reducible using the PCRS (cf. 'inductive reducibility', 'totality' [JK, Ko, MS2]). The simplest way to ensure that a function is defined by a PCRS for every possible constructor term argument is to have an 'otherwise'-case, an unconditional rule whose *lhs* has only variable arguments (or a set of conditional rules 'totally' defining the function, as in [Mo2]). Unlike ordinary TRS with such rules, we can avoid disturbing or complicating the ground confluence requirement: for each defined function f, a rule of the form $f(x_1, \ldots, x_n) \rightarrow \cdots$ is ordered to have a priority lower than every other rule defining the same function f. In the absence of such a rule, it is necessary to use techniques similar to those of [Th, Ko] to ensure that each function definition is total.

Proposition 7 If A DPCRS $\Re$

- satisfies the premise of the previous (confluence) proposition 5,

- satisfies the condition needed for termination (in proposition 4), and

- contains for each defined function f, a rule whose antecedent is empty and lhs is $f(x_1, \ldots, x_n)$,

then every ground term consisting of constructors and (one or more) functions defined by rules in $\Re$ has a unique ground constructor term reduct.

Proof: Let t be any non-constructor term whose non-constructor symbols are defined by rules in $\Re$. Since all reduction sequences terminate, t has an irreducible reduct t'. Since ground terms reduce to ground terms, t' is a ground irreducible term. Since each ground non-constructor term in the language must be reducible, t' must be a constructor term. If t had another irreducible reduct t'', then t'' would also have to be a ground constructor term. Since constructor terms are not reducible, t' and t'' would have to be identical, otherwise the ground confluence condition would be violated.

• Q.E.D.

In the next section, the utility of the above result is illustrated: a combination of the premises in the above result assures soundness and completeness of priority rewriting w.r.t. "Ψ_C", the set of inequations between non-unifiable constructor terms.

6 Semantics

The major motivation for using rewrite systems in algebraic specifications and theorem proving is their equational semantics; otherwise we would not know what is being computed when rewrite rules are invoked. In [BBK], a semantics is given for PRS with respect to a 'sound and complete set of rewrites'. This definition is operational in flavor: a logical semantics is preferable, which characterizes the theory (set of equational first order formulas) described by a PCRS, and whose logical consequences we should expect to compute via the priority rewriting mechanism. Such an approach was followed in [Mo1, Mo2, MS2] for CTRS and PRS, and we now extend it to PCRS.

Term rewriting systems (without priorities) are intended to represent a set of equations;

for example, the TRS $R = \{ p_1 \rightarrow q_1, \cdots, p_n \rightarrow q_n \}$ is intended to implement $\mathcal{E}(R)$ defined as $\{(\forall.p_1 = q_1) \wedge \cdots \wedge (\forall.p_n = q_n)\}$. Soundness of rewriting with R is judged by asking whether $\mathcal{E}(R) \models p = q$ whenever a term p rewrites to another term q using R; also, completeness of rewriting with R is considered to be the property that converging reduction sequences exist from any pair of terms whose equality is entailed by $\mathcal{E}(R)$. Priority orderings are not just a superficial way of improving efficiency; their introduction seriously modifies such semantics of rewriting systems. Since we can no longer perform certain rewrites which were possible using the underlying unprioritized system, the equational theories represented by prioritized systems are themselves different. The semantics of a PRS is no longer just the set of equations obtained by removing the orientation from underlying rewrite rules. The main reason for this is that priority rewriting with non-maximal priority rules is performed after making an assumption that higher priority rules are not applicable. So the soundness of such an assumption is a prerequisite for the soundness of priority rewriting. Note that such assumptions are utilized while constructing the PCRS, and hence cannot be ignored. For instance, the formula $\forall.eq(z,z) = T \wedge eq(x,y) = F$ does not represent the following PRS.

$$\left| \begin{array}{l} eq(z,z) \rightarrow T \\ eq(x,y) \rightarrow F \end{array} \right.$$

Hence the need for a better explanation for the semantics of priority rewriting.

With the above considerations, we suggest that the intended equational formulas $\boxed{\mathcal{E}(\Re)}$ corresponding to a PCRS $\Re$ may instead be defined as:

- For each maximal priority rule $[C \Rightarrow p \rightarrow q]$ in $\Re$, $\mathcal{E}(\Re)$ contains the conditional equation $\forall.C \Rightarrow p = q$.
- For each non-maximal priority rule $[r_k : C_k \Rightarrow f(\tau_k) \rightarrow p_k]$, let $\{r_1 : C_1 \Rightarrow f(\tau_1) \rightarrow p_1, \cdots, r_n : C_n \Rightarrow f(\tau_n) \rightarrow p_n\}$ be the rules in $\Re$ whose *lhs* has outermost symbol f, and which are of higher priority than r_k; then, $\mathcal{E}(\Re)$ contains

  $$\forall \mathcal{V}(\tau_k).[\neg C_k \vee f(\tau_k) = p_k \vee \exists \mathcal{V}(\tau_1)[C_1 \wedge \tau_1 = \tau_k] \vee \cdots \vee \exists \mathcal{V}(\tau_n)[C_n \wedge \tau_n = \tau_k]]$$

  (where each τ_i is a term-tuple, and variables in different rules are renamed to be distinct).

The first part of the above definition of $\mathcal{E}(\Re)$ reflects the fact that when there are no rules of higher priority, a rule $[C \Rightarrow l \rightarrow r]$ can be used for reducing any term matching l for which the correspondingly instantiated antecedent C is satisfied (subject to conditions in the definition of I-rewriting and P-rewriting), and hence is assumed to denote just $[C \Rightarrow l = r]$. The second part of the definition intuitively means that if a term $f(\tau)$, with irreducible arguments, is priority reducible by a rule $[C_k \Rightarrow f(\tau_k) \rightarrow p_k]$, then there is no rule of higher priority defining f whose arguments can be considered equal to τ_k and whose appropriately instantiated antecedent holds.

Example 6

$$\text{Let } \Re \text{ be } \left\{ \left\| \begin{array}{rl} & eq(x,x) \rightarrow T \\ type(u) = type(v) & \Rightarrow eq(u,v) \rightarrow F \\ & eq(z,y) \rightarrow \perp \end{array} \right\| \right\}.$$

$$\text{Then } \mathcal{E}(\Re) \text{ is } \left(\begin{array}{c} \forall x[eq(x,x) = T] \wedge \\ \forall u,v[type(u) \neq type(v) \ \vee \ eq(u,v) = F \ \vee \ u = v] \ \wedge \\ \forall z,y[eq(z,y) = \perp \ \vee z = y \ \vee \ type(z) = type(y)] \end{array} \right).$$

Note that $u = v$ and $z = y$ above are really abbreviations for $\exists x(x = u \wedge x = v)$ and $\exists x(x = z \wedge x = y)$ respectively, and $type(z) = type(y)$ above abbreviates the formula $\exists u, v.[u = z \wedge v = y \wedge type(u) = type(v)]$.

Using this PCRS, the term $eq(nil, 0)$ would reduce to $\perp$, assuming that $type(nil)$ and $type(0)$ evaluate to different constants (*list* and *number*, respectively). Such a reduction implicitly assumes the inequations $0 \neq nil$ and $list \neq number$, which are not directly provable as logical consequences of $\mathcal{E}(\Re)$, but are nevertheless needed for reduction. Thus, in addition to the above $\mathcal{E}(\Re)$ formulas, we need to make implicit default assumptions of inequations when performing reduction with a lower priority rule. We observe that non-Horn-clauses of the kind that may be contained in $\mathcal{E}(\Re)$ do not have initial or minimal models, and using such clauses to infer a single positive literal is possible only if we also assume some negative literals, *a la* negation by failure.

As in [MS2], we seek a set of inequations Ψ with respect to which priority rewriting is "sound" and "complete"; these semantic notions are defined below for ground term reductions, since we prohibit reduction of non-ground terms in most cases.

Definition 8 : Priority rewriting with $\Re$ is said to be

- sound w.r.t. Ψ iff $\forall$ ground $p, q.\ p \rightarrow_{\Re} q \ \Rightarrow\ \mathcal{E}(\Re) \cup \Psi \models p = q$; and
- complete w.r.t. Ψ iff $\forall$ ground $p, q.\ \mathcal{E}(\Re) \cup \Psi \models p = q \ \Rightarrow\ \exists t. p \rightarrow_{\Re}^{*} t \wedge q \rightarrow_{\Re}^{*} t$.

In a language with functions partitioned into free constructors and defined functions of fiexd arity, the semantics of PCRS specifications may be determined with respect to the stipulation that non-unifiable constructor terms cannot be equal. From a specification point of view, a sublanguage of 'free' constructor terms presents a useful candidate for an independently chosen set of inequations Ψ_C. For example, for rewrite specifications using a *List* data type, Ψ_C will consist of the inequations between non-unifiable terms constructed using *nil* and *cons*.

Definition 9 : Ψ_C is the set of inequations between non-unifiable constructor terms.

Note that arbitrary instantiation of variables in non-unifiable constructor terms, perhaps by non-constructor terms, would nevertheless leave them inequal as per this definition of Ψ_C. In other words, Ψ_C entails the inequality of any pair of constructor-wise distinct terms. For soundness and completeness of ground priority rewriting with respect to such independently defined Ψ, we need a property similar to 'sufficient completeness' [GH].

We now explore conditions under which priority rewriting is sound and complete, following a strategy described in [Mol]. Note that ordinary (unprioritized unconditional) rewriting is always sound, and confluence of a TRS implies completeness w.r.t. any Ψ. In priority rewriting, unlike conditional rewriting, we do not explicitly attempt to prove (by tracing reduction sequences to check for possible convergence) that the arguments of a lower priority rule cannot equal those of a higher priority rule whose *lhs* has the same outermost function

symbol. For example, given the PRS

$$\left|\begin{array}{l} f(f(x)) \rightarrow b \\ f(b) \rightarrow a \end{array}\right.$$

we find that $f(b)$ priority reduces to a, although $f(b) = a$ is not a logical consequence of

$$\overbrace{f(f(x)) = b \wedge [f(b) = a \vee \exists x.f(x) = b]}^{\mathcal{E}(\Re)} \wedge \overbrace{a \neq b \wedge \cdots}^{\Psi}$$

The need to avoid disproving existentially quantified disjuncts with non-constructor terms motivates the requirement that we should have a "Definitional" PCRS for soundness and completeness. This ensures that *lhs*'s of rules do not contain arguments whose reduction may conflict with the inequational assumption (*e.g.*, $\forall x.f(x) \neq b$ above) made during reduction with a lower priority rule. We also assume that rules in the PCRS are "normalized", *i.e.*, proper subterms of the *lhs* and the antecedent of each rule should not themselves be reducible. This has the beneficial side-effect of disallowing certain non-terminating rewrite sequence which would otherwise have been possible.

We now present the main results of this section, *viz.*, soundness and completeness of both definitions of priority rewriting when certain premises are satisfied.

Theorem 1 Let $\Re$ be a DPCRS such that every ground non-constructor term (containing constructors and function symbols occurring in $\Re$) I-reduces in finitely many steps to some ground constructor term. Then I-rewriting is sound and complete with respect to Ψ_C.

Theorem 2 Let $\Re$ be a DPCRS such that every ground non-constructor term (containing constructors and function symbols occurring in $\Re$) P-reduces in finitely many steps to some ground constructor term. Then ground P-rewriting is sound and complete with respect to Ψ_C.

Proofs of the above theorems (and a lemma) are given in the appendix. Earlier conjectures relaxing the premises of the above theorems (*viz.*, choice of Ψ_C, reducibility of non-ground constructor terms to ground constructor terms, and the requirement that $\Re$ must be a DPCRS) proved to be false, as illustrated by the following examples.

Example 7 :

$$\text{Let } \Re \text{ be } \left[\!\!\left[\begin{array}{c} f(f(y)) \rightarrow a \\ f(x) \rightarrow a \end{array} \right]\!\!\right]. \text{ Then } \mathcal{E}(\Re) \text{ is } \left(\begin{array}{c} [\forall y.f(f(y)) = a] \wedge \\ \forall x[f(x) = a \ \vee \ \exists y.x = f(y)] \end{array} \right).$$

Although each ground term (with f as the only non-constructor) reduces to the constructor term a, and all reduction sequences terminate, $\Re$ is not a DPCRS. Hence a reduction like $f(a) \rightarrow_{\Re} a$ is not sound, because $[\exists y.a = f(y)]$ in the relevant instance of the conjunct in $\mathcal{E}(\Re)$ is not contradicted by any Ψ consistent with $\mathcal{E}(\Re)$.

Example 8 : The current definition for Ψ_C is rather strong when compared to the analogous case for "EI"-conditional rewriting, whose soundness was proved in [Mo1] with respect to the set of inequations between distinct ground constructor terms. The weaker definition proved to be inadequate. For instance, let (unary) c and (nullary) a be constructors, and let the suggested Ψ be the set of inequations between each pair of distinct ground constructor terms, *i.e.*, $\{c^i(a) \neq c^j(a) | \ i \geq 0, \ j \geq 0, \ i \neq j\}$.

$$\text{Let } \Re \text{ be } \left[\!\!\left[\begin{array}{c} f(c(y)) \rightarrow y \\ f(z) \rightarrow c(z) \end{array} \right]\!\!\right]. \text{ Then } \mathcal{E}(\Re) \text{ is } \left(\begin{array}{c} [\forall y.f(c(y)) = y] \wedge \\ \forall z.f(z) = c(z) \ \vee \ \exists y.z = c(y) \end{array} \right).$$

The reduction of $f(a)$ to $c(a)$ using the second rule would be sound iff we can show that $\mathcal{E}(\Re) \cup \Psi \models \neg\exists y.[a = c(y)]$. But we cannot conclude from $\mathcal{E}(\Re) \cup \Psi$ that $a \neq c(f(a))$, since we cannot assume that $f(a)$ must equal a ground constructor term, although $f(a)$ can be reduced to a ground constructor term. Soundness obtains with the stronger definition, *viz.*, when Ψ_C implies that any two constructor-wise distinct terms like a, $c(f(a))$ are inequal.

Example 9 :

$$\text{Let } \Re \text{ be } \left[\!\!\left[\begin{array}{c} f(c(y)) \rightarrow a \\ f(z) \rightarrow a \\ g(c(a)) \rightarrow a \end{array} \right]\!\!\right]. \text{ Then } \mathcal{E}(\Re) \text{ is } \left[\begin{array}{c} [\forall y.f(c(y)) = a] \wedge \\ [\forall z.f(z) = a \vee \exists y.z = c(y)] \wedge \\ g(c(a)) = a \end{array} \right].$$

Not all terms have ground constructor term reducts using this DPCRS. Here, although $f(g(a))$ reduces to a using the second rule, we cannot conclude or assume the negation of

$\exists y.g(a) = c(y)$, the latter disjunct in the relevant instance of the second conjunct in $\mathcal{E}(\Re)$. Hence we cannot conclude $\mathcal{E}(\Re) \cup \Psi_C \models f(g(a)) = a$.

If the rules defining g had instead reduced $g(a)$ to a constructor term d whose outermost symbol is distinct from c, then $\Psi_C \models \forall.d \neq c(y)$, and soundness induction over reductions in the proof tree implies $\mathcal{E}(\Re) \cup \Psi_C \models \forall.g(a) = d \neq c(y)$, hence allowing the conclusion that $\mathcal{E}(\Re) \cup \Psi_C \models f(g(a)) = a$. In other words, soundness of the reduction of $f(g(a))$ to a depends on the existence of a reduction sequence from $g(a)$ to a ground constructor term. Note that if $g(a)$ had instead been reducible to a constructor term $t \equiv c(\cdots)$, whose outermost symbol is c, then:

- I-rewriting of $f(g(a))$ cannot occur at the top level before first replacing $g(a)$ by t, so that the resulting term $f(c(\cdots))$ is reducible by the first rule and not the second.
- P-rewriting of $f(g(a))$ can also not occur directly using the second rule because the definition of P-rewriting requires that we must first ensure that rewriting sequences from proper subterms should not result in a term reducible by a higher priority rule; hence the subterm $g(a)$ must first be reduced, after which only the first rule can be applied.

7 Summary

In this paper, we have first presented and compared priority rewriting and conditional rewriting, discussing when priority rewriting systems do and do not have equivalent conditional rewriting systems and vice versa. We then studied the powerful combination of these two rewriting mechanisms, *viz.*, priority conditional rewriting systems, which consist of conditional rules on which a partial priority ordering is imposed. We have given the logical semantics of such systems, extending a similar approach to the semantics of conditional rewriting with negation. We gave restrictions needed for soundness and completeness of priority rewriting, focusing on a sufficient condition for ground confluence.

In conclusion, PCRS appear to be provide a promising specification language, with a reasonable equational-logical declarative semantics, and simple operational mechanisms.

References

[BBK] J.C.M.Baeten, J.A.Bergstra, J.W.Klop, *Term Rewriting Systems with Priorities,* Proc. II Conf. Rewriting Techniques and Applications, France, Springer-Verlag LNCS 256, 1987, pp83-94.

[BDJ] D.Brand, J.A.Darringer, W.Joyner, *Completeness of Conditional Reductions,* Res. Rep. RC7404, IBM T.J.Watson Res. Center, Yorktown Heights (NY), 1978.

[De] N. Dershowitz, *Termination of Rewriting,* Proc. First Int'l. Conf. on Rewriting Techniques and Applications, Dijon (France), 1985, Springer-Verlag LNCS 202, pp180-224.

[DJ] N.Dershowitz, J.-P.Jouannaud, *Rewriting Systems,* (to appear) in Handbook of Theoretical Computer Science, 1989.

[DP] N.Dershowitz, D.Plaisted, *Logic Programming cum Applicative Programming,* Proc. Symp. on Logic Programming, Boston, 1985.

[Ga] H.Ganzinger, *Ground Term Confluence in Parametric Conditional Equational Specifications,* Proc. STACS 87, 1987, pp286-298.

[GH] J.V.Guttag, J.J.Horning, *The Algebraic Specification of Abstract Data Types,* Acta Informatica 10, 1978, pp27-52.

[HO] G.Huet, D.S.Oppen, *Equations and Rewrite Rules: A Survey,* in Formal Languages: Perspectives and Open Problems, R.Book (ed.), Academic Press, 1980, pp349-405.

[Hu] J.-M. Hullot, *Canonical Forms and Unification,* 222zProc. Fifth Conf. on Automated Deduction, Les Arcs (France), July 1980, pp318-334.

[JK] J.-P.Jouannaud, E.Kounalis, *Proofs by Induction in Equational Theories without Constructors,* Symp. on Logic in C.S., Cambridge (Mass.), USA, 1986, pp358-366.

[JW] J.-P.Jouannaud, B.Waldmann, *Reductive Conditional Term Rewriting Systems,* Proc. 3rd IFIP Working Conf. on Formal Description of Programming Concepts, Ebberup (Denmark), 1986.

[Ka1] S.Kaplan, *Conditional Rewrite Rules,* Theoretical Computer Science 33, 1984, pp175-193.

[Ka2] S.Kaplan, *A Compiler for Conditional Term Rewriting Systems,* Proc. II Conf. Rewriting Techniques and Applications, France, Springer-Verlag LNCS 256, 1987, pp25-41.

[Ka3] S.Kaplan, *Positive/Negative Conditional Rewriting,* Proc. First Int'l. Workshop on Conditional Term Rewriting Systems, Orsay (France), Springer-Verlag LNCS 308, 1988, pp129-143.

[KJ] S.Kaplan, J.-P.Jouannaud (eds.), *Proc. First Int'l. Workshop on Conditional Term Rewriting Systems,* Orsay (France), Springer-Verlag LNCS 308, 1988.

[Ko] E.Kounalis, *Completeness in Data Type Specifications,* Res. Rep. 84-R-92, C.R.I.N., Nancy (France), 1984.

[Ma] M.J.Maher, *Equivalences of Logic Programs,* in Deductive Databases and Logic Programming, (ed., J.Minker), Morgan Kaufmann Pub., 1988, pp627-658.

[Mo1] C.K.Mohan, *Negation in Equational Reasoning and Conditional Specifications,* Ph.D. Thesis, State University of New York at Stony Brook, 1988.

[Mo2] C.K.Mohan, *Priority Rewriting: Semantics, Confluence and Conditionals,* in Proc. 3^{rd} Conf. on Rewriting Techniques and Applications, Chapel Hill, Springer-Verlag LNCS 355, April 1989.

[MS1] C.K.Mohan, M.K.Srivas, *Function Definitions in Term Rewriting and Applicative Programming,* Information and Control, Dec. 1986.

[MS2] C.K.Mohan, M.K.Srivas, *Conditional Specifications with Inequational Assumptions,* Proc. First Int'l. Workshop on Conditional Term Rewriting Systems, Orsay (France), Springer-Verlag LNCS 308, 1988, pp161-178.

[MS3] C.K.Mohan, M.K.Srivas, *Negation with Logical Variables in Conditional Rewriting,* Proc. 3^{rd} Conf. on Rewriting Techniques and Applications, Springer-Verlag LNCS 355, Chapel Hill, April 1989.

[MSK] C.K.Mohan, M.K.Srivas, D.Kapur, *Inference Rules and Proof Procedures for Inequations,* (to appear in) The Journal of Logic Programming.

[Na] M.L.Navarro, *Tecnicas de Reescritura para Especificaciones Condicionales,* Ph.D. Thesis, Polytechnic Univ. of Catalunya, Barcelona (Spain), 1987.

[Re] J.-L.Remy, *Etudes des systemes de reecriture conditionnels et application aux types abstraits algebriques,* These d'Etat, Nancy (France), 1982.

[Th] J.J.Thiel, *Stop Losing Sleep over Incomplete Specifications,* Proc. 11^{th} ACM Symp. on Princ. of Prog. Lang., 1983.

[ZR] H.Zhang, J.-L.Remy, *REVEUR4: A system to proceed experiments on Conditional Term Rewriting Systems,* Tech. Rep., CRIN, Nancy (France), 1985.

APPENDIX (Proofs of Results)

Proposition 1: Let $\Re$ be a PRS whose underlying TRS R is confluent and noetherian. Then
(1) any ground term reducible using R is also P-reducible using $\Re$; and
(2) if $p \rightarrow^P_{\Re} q$ then there are converging reduction sequences from every $M[p]$ and $M[p/q]$.

Proof: The imposition of priorities merely restricts the set of reductions possible using the underlying TRS, *i.e.*, $p \rightarrow^P_{\Re} q \Rightarrow p \rightarrow_R q$. We first present a weak converse.

(1): Suppose the ground term $p \rightarrow_R q$ but p is not P-reducible by $\Re$. Then (a subterm of) p matches the *lhs* of some rule $\in \Re$. Since R is noetherian, P-rewriting using $\Re$ is decidable, *i.e.*, any attempt to P-reduce a term finitely fails or succeeds. If more than one such rule exists, then choose the rule $r : \lambda \rightarrow \rho \in \Re$ of highest priority such that (a subterm of) $p \equiv \lambda\sigma$ for some substitution σ. If p cannot be P-reduced by this rule, either it must have been P-reducible by a higher priority rule (in which case p is P-reducible), or some subterm t of p must reduce to a term t' such that $p[t/t']$ is reducible by a higher priority rule (in which case p is P-reducible at a subterm). In either case, p is P-reducible by $\Re$.

(2): Suppose $p \rightarrow^P_{\Re} q$, implying also that $p \rightarrow_R q$. Since R is noetherian, every P-reduction sequence finitely terminates, and each term has a P-irreducible reduct. Let M_1 and M_2 be

the respective P-irreducible reducts of $M[p]$ and $M[p/q]$. Since M_1 and M_2 cannot be P-reduced by $\Re$, they cannot be reduced by R either, by part (1) above. Since $M[p] \rightarrow_{\Re}^{P*} M_1$, $p \rightarrow_{\Re}^{P} q$, and $M[p/q] \rightarrow_{\Re}^{P*} M_2$, we conclude that $M[p] \rightarrow_R^* M_1$ as well as $M[p] \rightarrow_R^* M_2$. Since R is confluent and M_1, M_2 are not reducible by R, we conclude that $M_1 \equiv M_2$. Thus the P-reduction sequences from $M[p]$ and $M[p/q]$ have converged.

Q.E.D.

•

Proposition 2: If an EI-CTRS R satisfies the property that every reduction sequence from any given ground term t finitely terminates yielding the same irreducible constructor term $\mathcal{N}_R(t)$, then a PRS $\mathcal{P}(R)$ can be constructed such that priority reduction sequences from any ground term t using $\mathcal{P}(R)$ terminate yielding precisely the same normal form $\mathcal{N}_R(t)$ produced by R.

Proof: This result follows by observing that a modification of the transformation suggested in section 3.1 suffices to efffectively produce the desired PRS $\mathcal{P}(R)$. We include rules $r_1 - r_7$ as well as r_{10}, r_{11} mentioned in section 3.1 which define four logical functions: *not*, *eq*, *neq*, *and*, as well as $if3$ (representing "if-then-else"). These rules, assumed to be of higher priority than others, are:

$$
\left|
\begin{array}{r}
r_1 : not(T) \rightarrow F \\
r_2 : not(F) \rightarrow T \\
r_3 : eq(x,x) \rightarrow T \\
r_4 : eq(x,y) \rightarrow F \\
r_5 : neq(x,y) \rightarrow not(eq(x,y)) \\
r_6 : and(T,T) \rightarrow T \\
r_7 : and(x,y) \rightarrow F \\
r_{10} : if3(T,x,y) \rightarrow x \\
r_{11} : if3(z,x,y) \rightarrow y
\end{array}
\right.
$$

For each pair of rules r, r' in R whose *lhs*'s unify with m.g.u. σ, add the corresponding instances $r\sigma$ and $r'\sigma$ to R; let $\hat{R}$ be the resulting EI-CTRS. For each rule $r : C \Rightarrow \lambda \rightarrow \rho \in \hat{R}$, let R_r be the set of appropriate instances of rules in $\hat{R}$ such that the *lhs*'s of all rules in R_r are identical (to λ). Redundant rule-sets obtained as variants of R_r (differing only in names

of variables) may be discarded. Now, if a ground term which matches λ is EI-reducible at the outermost level using R, it must be reducible using some rule in R_τ.

Let R_τ consist of rules $\{C_1 \Rightarrow \lambda \rightarrow \rho_1, \cdots, C_m \Rightarrow \lambda \rightarrow \rho_m\}$. We now organize these rules into an equivalent sequence R_λ, to be executed one after another as and when a term to be reduced has a subterm that matches with λ. C_i, the antecedent $[p_1 = q_1 \wedge \cdots \wedge r_n \neq s_n]$ of each rule above, can be reformulated as C_i', a term $and[eq(p_1, q_1), and(\cdots\cdots neq(r_n, s_n)]$. The first rule in the sequence R_λ is $[\lambda \rightarrow if3(C_1', \rho_1, \lambda_2(\overline{x}))]$, where $\overline{x} \equiv \mathcal{V}(\lambda)$. Each remaining rule in R_λ is $[\lambda_j(\overline{x}) \rightarrow if3(C_j', \rho_j, \lambda_{j+1}(\overline{x}))]$, (where $1 < j \leq m$). Since every ground term has a unique irreducible constructor term reduct using R, if a ground term matches with λ and EI-reduces using a rule in R_τ then one of the antecedents C_j' will certainly hold (reduce to T), and no ground term will ever reduce to the last term $\lambda_{m+1}(\overline{x})$ in the rhs of the last such rule in the sequence R_λ.

The PRS $\mathcal{P}(R)$ is defined to have the underlying TRS consists of the basic logical rules given above $(r_1 - r_7, r_{10}, r_{11})$ as well as the union of the sets of rules R_λ obtained from each rule (with distinct lhs λ) in $\hat{R}$. A priority ordering is imposed on these rules, such that the logical rules have higher priority than all others, and whenever λ is (strictly) an instance of λ' (and not vice versa), every rule in R_λ is of higher priority than every rule in $R_{\lambda'}$. This ensures that the rules with most specific lhs's are used for reduction whenever possible.

The above construction of $\mathcal{P}(R)$ is such that each ground reduction step using the rule in R with most general lhs is mimicked by a terminating reduction sequence using $\mathcal{P}(R)$ which leads to the same result as that obtained using R.

• Q.E.D.

Proposition 3: Let $\Re$ be a PRS such that every priority reduction sequence terminates, and whenever $[r_1 : f(\overline{s}) \rightarrow p] > [r_2 : g(\overline{t}) \rightarrow q]$ as per the priority ordering of $\Re$, we have $f \equiv g$ and $\overline{s}$ consists only of variables and ground terms. Then an equivalent EI-CTRS $\mathcal{C}(\Re)$ can be constructed such that every ground term has the same irreducible reducts using $\Re$ and $\mathcal{C}(\Re)$.

Proof: Maximal priority rules appear undisturbed in the new CTRS obtained by modifying the given PRS $\Re$. We now show how to construct conditional rewrite rules $\in \mathcal{C}(\Re)$ which

achieve the same effect via EI-rewriting whenever a corresponding rule in $\Re$ (of non-maximal priority) can be used to rewrite a term.

Rename all variables in different rules, to make them distinct. For any rule $[r_k : f(s_1^k, \cdots, s_m^k) \rightarrow \rho^k] \in \Re$, let R_k be $\{r_i : f(s_1^i, \cdots, s_m^i) \rightarrow \rho^i | r_i \in \Re \wedge r_i > r_k\}$. The premise in the statement of the proposition ensures that each s_k^i must be a variable or a ground term.

For each variable multiply occurring as the subterms $s_{j_1}^i, \cdots, s_{j_n}^i$ in the *lhs* of r_i, construct a disjunct asserting that some corresponding pair of arguments in the *lhs* of r_k must be inequal; let C_j^i (for each such variable) be $\{s_{j_1}^k \neq s_{j_2}^k, \cdots, s_{j_a}^k \neq s_{j_b}^k, \cdots, s_{j_{n-1}}^k \neq s_{j_n}^k\}$. Now construct a disjunct asserting that for some other (ground) argument s_j^i of the *lhs* of r_i, the corresponding argument s_j^k in the *lhs* of r_k is not equal to s_j^i; let C_j^i in this case be $\{s_j^k \neq s_j^i\}$. Let C_i be the union (disjunction) of each of the above disjuncts C_j^i obtained from variable and ground term arguments of r_i.

Let C be the conjunction of the disjunctions C_i obtained from each rule r_i in R_k, *i.e.*,

$$C \equiv \bigwedge_{i:r_i \in R_k} [\bigvee_{ground\ s_j^i} s_j^k \neq s_j^i \bigvee_{variable\ s_{j_a}^i \equiv s_{j_b}^i} s_{j_a}^k \neq s_{j_b}^k]$$

By applying DeMorgan's laws, C can be restructured into $[D_1 \vee \cdots \vee D_N]$, a disjunction of conjunctions D_l. The following set of rules are finally obtained corresponding to $r_k \in \Re$:

$$\begin{aligned} D_1 &\Rightarrow f(s_1^k, \cdots, s_m^k) \rightarrow \rho^k \\ \cdots &\Rightarrow \quad \cdots \quad \rightarrow \cdots \\ D_N &\Rightarrow f(s_1^k, \cdots, s_m^k) \rightarrow \rho^k \end{aligned}$$

Q.E.D.

Theorem 1: Let $\Re$ be a DPCRS such that every ground non-constructor term (containing function symbols occurring in $\Re$ in addition to constructors) I-reduces in finitely many steps to some ground constructor term. Then I-rewriting with $\Re$ is sound and complete w.r.t. Ψ_C.

Proof:

Soundness: For soundness, we need to show that if $p \rightarrow_{\Re} q$, then $\mathcal{E}(\Re) \cup \Psi_C \models p = q$. Since

$p = q$ implies $M[p] = M[p/q]$ in equational logic, we need not separately consider reductions which occur by replacing proper subterms of terms. It is sufficient to consider the restricted case of top-level reductions, wherein a term in its entirety matches the *lhs* of a rule, and is hence replaced by the appropriate instance of the *rhs* of that rule.

Let $p \rightarrow^{I}_{\mathcal{R}} q$ using the rule $[r : C \Rightarrow \lambda \rightarrow \rho]$, using a matching substitution σ such that $p \equiv \lambda\sigma$, $q \equiv \rho\sigma$, and for each equation $m = n \in C$ it is the case that top-level reductions from $m\sigma$ and $n\sigma$ converge. It is also necessary that p could not be reduced using any rule $[r_i : C_i \Rightarrow \lambda_i \rightarrow \rho_i]$ of higher priority than r. Thus the single step $p \rightarrow_{\mathcal{R}} q$ involves the following successive terminating computations:

1. Does p match with the *lhs* any of the higher priority rules (r_i)?
2. If yes, with matching substitution σ_i, is there an equation $m_i = n_i \in C_i$ such that reduction sequences from $m_i\sigma_i$, $n_i\sigma_i$ lead to distinct ground constructor terms?
3. Does p match with λ?
4. If yes, with matching substitution σ, are there converging reduction sequences (to the same ground constructor term), from $m\sigma$, $n\sigma$, for each equation $m = n \in C$?

We consider the "Proof Tree" of $p \rightarrow_{\mathcal{R}} q$ to consist of all these steps, and assume for every reduction step $s \rightarrow_{\mathcal{R}} t$ in this proof tree that $\mathcal{E}(\mathcal{R}) \cup \Psi_C \models s = t$. This constitutes our Induction Hypothesis, and is justified because the above proof tree has to be finite if reduction of p to q occurs. The soundness of reductions using unconditional rules, and rules whose antecedents trivially hold, is assured by the soundness of modus ponens and replacement of equals by equals.

Since reduction did not occur using any higher priority rule, we conclude that

1. <u>either</u> p did not match with the *lhs* of any higher priority rule,
2. <u>or</u> if $p \equiv \lambda_i\sigma_i$ for some σ_i and a higher priority rule $[r_i : C_i \Rightarrow \lambda_i \rightarrow \rho_i]$, then there is some literal $m_i = n_i \in C_i$ such that $m_i\sigma_i$ and $n_i\sigma_i$ have reduction sequences which did not converge (or led to distinct ground constructor terms).

Note that for any maximal priority rule $C_0 \Rightarrow \lambda_0 \rightarrow \rho_0$, the following discussion of these two cases is not needed since $\mathcal{E}(\Re)$ contains a conjunct $C_0 \Rightarrow \lambda_0 = \rho_0$, with no disjuncts arising from other rules. For each non-maximal priority rule $[r : C \Rightarrow \lambda \rightarrow \rho]$ of lower priority than rules $\{r_i : C_i \Rightarrow \lambda_i \rightarrow \rho_i\}$ whose *lhs*'s $(\lambda_i \equiv f(t_1^i, \cdots, t_n^i))$ have the same outermost symbol f as that of $\lambda \equiv f(t_1, \cdots, t_n)$, the instance $\mathcal{E}_p$ of the relevant conjunct in $\mathcal{E}(\Re)$ which justifies reduction of $p \equiv \lambda\sigma$ to $q \equiv \rho\sigma$ using r is

$$\mathcal{E}_p : \qquad (C\sigma \Rightarrow f(t_1, \cdots, t_n)\sigma = \rho\sigma) \vee [\cdots \vee [\exists . C_i \wedge t_1\sigma = t_1^i \wedge \cdots \wedge t_n\sigma = t_n^i] \vee \cdots].$$

The following discussion first shows how we may eliminate each of the latter disjuncts in the above formula $\mathcal{E}_p$, and then the antecedent $(C\sigma)$ of the first disjunct. Since rules which do not define the same function symbol do not affect the relevant conjunct $\mathcal{E}_p$ in the definition of $\mathcal{E}(\Re)$, it is sufficient to restrict attention to rules $\{r_i\}$ which define the outermost symbol f of p.

Case $[\forall . p \not\equiv \lambda_i]$: By definition of the I-rewriting mechanism, proper subterms of p cannot be I-reducible since p I-reduced to q by replacement of the entire term. Since all ground non-constructor terms are reducible, all proper subterms of p must be ground constructor terms [this need not hold for P-rewriting]. For each rule r_i whose *lhs* does not match with p, we need to show that $\mathcal{E}(\Re) \cup \Psi_C \models \neg \wedge_j [t_j\sigma = t_j^i\sigma_i]$, for each grounding substitution σ_i. By the lemma that follows, it is sufficient to consider only those substitutions σ_i which map variables (occurring in terms to which they are applied) only to ground constructor terms. Since $p \equiv \lambda\sigma \equiv f(t_1\sigma, \cdots, t_n\sigma) \not\equiv \lambda_i\sigma_i$, the latter must have a ground constructor subterm $t_j^i\sigma_i$ distinct from $t_j\sigma$, hence $\Psi_C \models t_j\sigma \neq t_j^i\sigma_i$. So for no grounding substitution σ_i can it be the case that each $t_j\sigma = t_j^i\sigma_i$ is a logical consequence of $\mathcal{E}(\Re) \cup \Psi_C$ (assuming consistency). Thus we can delete each of the disjuncts in the formula $\mathcal{E}_p$ which arose from rules whose *lhs* could not be matched with p.

Case $\exists . p \equiv \lambda_i$: For disjuncts in $\mathcal{E}_p$ arising from the remaining rules $\{r_k\}$ whose *lhs*'s can be matched with p, since the matching substitution (σ_k) of a ground term p with each λ_k must be unique, each disjunct $[\exists . C_k \wedge t_1\sigma = t_1^k \wedge \cdots \wedge t_n\sigma = t_n^k]$ can be replaced by just $C_k\sigma_k$. Let $\mathcal{E}'_p$ be the new formula $[C\sigma \Rightarrow p = q \vee \cdots \vee C_k\sigma_k \vee \cdots]$, where each remaining disjunct $C_k\sigma_k$

arises from a rule $r_k : C_k \Rightarrow \lambda_k \rightarrow \rho_k$ whose *lhs* (λ_k) was matched with p by substitution σ_k. Since $p \equiv \lambda\sigma$ and $q \equiv \rho\sigma$, we conclude from $\mathcal{E}_p$ and the previous case that $\mathcal{E}(\Re) \cup \Psi_C \models \mathcal{E}'_p$.

Since $p \equiv \lambda_k\sigma_k$ but rule r_k could not reduce p, C_k must contain some literal $m = n$ such that $m\sigma_k$ and $n\sigma_k$ do not have converging reduction sequences. [Attempts to reduce $m\sigma_k$ and $n\sigma_k$ to the same reduct comprise part of the proof tree of $p \rightarrow_{\Re} q$ mentioned earlier.] Let these two terms $m\sigma_k$, $n\sigma_k$ have M and N as ground constructor reducts (which exist as per the premise in the statement of the theorem). M and N must be distinct due to non-convergence of the reduction sequences, so that $\Psi_C \models M \neq N$. Since $m\sigma_k \rightarrow^*_{\Re} M$ and $n\sigma_k \rightarrow^*_{\Re} N$, by induction hypothesis (soundness of each reduction step in the proof tree of $p \rightarrow_{\Re} q$) and transitivity of equality we conclude that $\mathcal{E}(\Re) \cup \Psi_C \models m\sigma_k = M$ and $\mathcal{E}(\Re) \cup \Psi_C \models n\sigma_k = N$. Now $\Psi_C \models M \neq N$ implies that $\mathcal{E}(\Re) \cup \Psi_C \models m\sigma_k \neq n\sigma_k$, hence $C_k\sigma_k$ is inconsistent with $\mathcal{E}(\Re) \cup \Psi_C$. Hence $\mathcal{E}(\Re) \cup \Psi_C \models C\sigma \Rightarrow p = q$, obtained by deleting each such disjunct $C_k\sigma_k$ in the formula $\mathcal{E}'_p$ obtained from the preceding analysis.

Finally, since reduction using rule r succeeded, for each literal $p_j = q_j \in C$, there must have been reduction sequences from $p_j\sigma$, $q_j\sigma$ which converged to some term e_j. By induction hypothesis, and transitivity of equality, we conclude that $\mathcal{E}(\Re) \cup \Psi_C \models p_j\sigma = e_j = q_j\sigma$. Hence $\mathcal{E}(\Re) \cup \Psi_C \models C\sigma$, and it follows by modus ponens that $\mathcal{E}(\Re) \cup \Psi_C \models p = q$. This concludes the proof: if $p \rightarrow^I_{\Re} q$, then $p = q$ is a logical consequence of $\mathcal{E}(\Re) \cup \Psi_C$.

Completeness: For completeness, we need to show that if $\mathcal{E}(\Re) \cup \Psi_C \models p = q$, then there are converging reduction sequences from the ground terms p and q, as per the premise in the statement of the theorem. This result follows because p and q must have some ground constructor reducts p' and q' respectively. Soundness of priority reduction with $\Re$ (proved above) implies that

$\mathcal{E}(\Re) \cup \Psi_C \models (p = p') \wedge (q = q')$. If p' and q' are not identical, $\Psi_C \models p' \neq q'$, and transitivity of equality implies a contradiction with $\mathcal{E}(\Re) \cup \Psi_C \models p = q$. Hence $p' \equiv q'$, and equal terms have converging reduction sequences.

• Q.E.D.

Lemma 1 : Let $\tau = < \cdots t_j \cdots >$ and $\tau' = < \cdots t'_j \cdots >$ be two tuples of constructor terms such that τ is ground, and there is some substitution σ' such that $\mathcal{E}(\Re) \cup \Psi_C \models \tau = \tau'\sigma'$. Then, if $\mathcal{E}(\Re) \cup \Psi_C$ is consistent, there is a substitution θ (mapping variables of τ' to ground constructor terms) such that $\tau \equiv \tau'\theta$.

Proof: If σ' itself is a substitution mapping variables in τ' to ground constructor terms, then τ and $\tau'\sigma'$ must be identical; otherwise, the equality of two distinct ground constructor terms would be implied by $\mathcal{E}(\Re) \cup \Psi_C$, which contradicts the definition of Ψ_C, assuming consistency of $\mathcal{E}(\Re) \cup \Psi_C$. The three cases (below) deal with situations in which σ' maps variables in τ' to non-constructor terms, so that $\tau \not\equiv \tau'\sigma'$, which means that they must differ at some j^{th} position, *i.e.*, $\exists j.t_j \not\equiv t'_j\sigma'$.

Case 1: If τ and $\tau'\sigma'$ differ at some j^{th} position such that $t'_j\sigma'$ and t_j are constructor-wise distinct, then $\Psi_C \models t_j \neq t'_j\sigma'$, contradicting the consistency of $\mathcal{E}(\Re) \cup \Psi_C$ which entails $t_j = t'_j\sigma'$. This leaves the remaining cases where $t'_j\sigma'$ differs from t_j at some non-constructor subterm.

Case 2: If τ and $\tau'\sigma'$ differ at some j^{th} position, and the outermost symbol of $t'_j\sigma'$ is a non-constructor, then t'_j must have been a variable, since τ' was a tuple of constructor terms. Hence $\mathcal{E}(\Re) \cup \Psi_C \models \tau = \tau'\sigma'$ implies that $\mathcal{E}(\Re) \cup \Psi_C \models \tau = \tau'\theta$, where θ is a new substitution which is identical to σ' except that each such variable t'_j is mapped to t_j instead of $t'_j\sigma'$. This process of modifying the substitution σ' is repeated until all such variables t'_j are mapped to the corresponding ground constructor terms t_j. If this cannot be done consistently because of non-linearity (multiple occurrences of the same variable in τ' being assigned to distinct ground constructor subterms from τ), then $\mathcal{E}(\Re) \cup \Psi_C$ must be inconsistent, because $\mathcal{E}(\Re) \cup \Psi_C \models (t_{j1} = t'_j\sigma') \wedge (t_{j2} = t'_j\sigma')$ and $t_{j1} \not\equiv t_{j2}$ is not possible; distinct ground constructor terms cannot be equal.

Case 3: The last remaining case is when τ and $\tau'\sigma'$ differ at some j^{th} position only because

there are some subterm positions where $t'_j\sigma'$ has a non-constructor in a position corresponding to that of a constructor in t_j. For example, $t_j \equiv c[\cdots c_k \cdots]$ and $t'_j\sigma' \equiv c[\cdots s_k \cdots]$ with s_k occurring in t'_j at a position (at arbitrary depth) corresponding to that c_k in t_j, and in the tree representation of the terms t_j and $t'_j\sigma'$ the same sequence of constructors separates the root c from c_k and s_k respectively. Clearly, replacing s_k by c_k in the offending position of $t'_j\sigma_j$ would not affect the equality with t_j entailed by $\mathcal{E}(\Re) \cup \Psi_C$. Since t'_j is a constructor term, σ' must have instantiated some variable x to some superterm of s_k. As in case 2, we can obtain a new equivalent substitution θ which is identical to σ' except in mapping that variable x to c_k. This modification can only preserve equality since $\mathcal{E}(\Re) \cup \Psi_C \models t_j = t'_j\sigma'$ implies $\mathcal{E}(\Re) \cup \Psi_C \models t_j = t'_j\sigma'[s_k/c_k]$. Hence $\mathcal{E}(\Re) \cup \Psi_C \models \tau = \tau'\sigma'$ implies that $\mathcal{E}(\Re) \cup \Psi_C \models \tau = \tau'\theta$, and this process of modifying σ' can be repeated, deleting various occurrences of non-constructor terms.

At the end of deleting all non-constructor terms from $\tau'\sigma'$ as in cases 2 and 3 above, modifying σ' in the process, we are left with a ground constructor term τ'' such that $\mathcal{E}(\Re) \cup \Psi_C \models \tau'\sigma' = \tau''$. The process terminates because there are only finitely many non-constructor terms in $\tau'\sigma'$, at least one of which is deleted at each step, and no new non-constructor symbol is introduced at any step. If τ'' is identical to τ, we conclude that a matching substitution between τ and τ' has been found; otherwise, we conclude an inconsistency, since $\Psi_C \models \tau \neq \tau''$, hence it cannot be the case that $\mathcal{E}(\Re) \cup \Psi_C \models \tau = \tau'\sigma'$.

• Q.E.D.

Theorem 2: Let $\Re$ be a DPCRS such that every ground term (containing constructors and function symbols occurring in $\Re$) P-reduces in finitely many steps to some ground constructor term. Then ground P-rewriting with $\Re$ is sound and complete w.r.t. Ψ_C.

Proof: For P-rewriting, the proof of completeness is identical to that of I-rewriting. But the proof of soundness must additionally consider the fact that a ground term p may be P-reducible at the topmost level although it may have a proper subterm p_i which is also P-reducible using some rule in the PCRS. Since p_i must also have a ground constructor

reduct q_i, we may first focus on the soundness of $p_i \rightarrow_{\Re}^{P*} q_i$ and then on the soundness of the P-reduction of $p[p_i/q_i]$. Even if the P-reduction of p_i does not necessarily precede the reduction by replacement of the entire term p, the replacement of p_i by q_i preserves equality, and is logically permissible, if $\mathcal{E}(\Re) \cup \Psi_C \models p_i = q_i$.

So let p_i be an innermost non-constructor (hence reducible) subterm of p. P-reduction of p_i is sound, using the same proof as that used for the soundness of I-reduction. Since the reduction of p_i could decidably occur, its soundness does not depend circularly on the soundness of the P-reduction of a superterm $f(\cdots p_i \cdots)$ using a rule which is not of maximal priority; this is ensured by the definition of P-rewriting which requires that P-reduction using a rule of non-maximal priority can occur only if it is first established that enforcing prior replacement of proper subterms by their reducts does not lead to reducibility using a higher priority rule ($> r'$). [Note that soundness of reduction by maximal priority rules does not depend on other rules as much, because the corresponding formula in $\mathcal{E}(\Re)$ is just of the form $C' \Rightarrow \lambda' \rightarrow \rho'$, without other disjuncts].

By induction on the soundness of the reduction sequence from p_i to the ground constructor term q_i, and transitivity of equality, we have $\mathcal{E}(\Re) \cup \Psi_C \models p_i = q_i$. Since replacement of equals by equals is sound, we hence have $\mathcal{E}(\Re) \cup \Psi_C \models p = p[p_i/q_i]$. In this fashion, each innermost non-constructor proper subterm of p is successively replaced by the corresponding constructor term reducts until we have $\mathcal{E}(\Re) \cup \Psi_C \models p = p' \equiv p[p_1/q_1, \cdots, p_m/q_m]$, where all proper subterms of p' are ground constructor terms.

Since $\Re$ is a DPCRS, proper subterms of the *lhs* of each rule are constructor terms. So if p is P-reducible to q using the rule $r : C \Rightarrow \lambda \rightarrow \rho$, then $p' \equiv p[p_1/q_1, \cdots, p_m/q_m]$ is also P-reducible to a term $q' \equiv q[p_1/q_1, \cdots, p_m/q_m]$, since each p_i (or a superterm of p_i) must occur in a position corresponding to that of a variable in λ, and since p' cannot be P-reduced by a higher priority rule. The proof of soundness of I-reduction can now be grafted in, giving $\mathcal{E}(\Re) \cup \Psi_C \models p' = q'$. Since for each i it is the case that $\mathcal{E}(\Re) \cup \Psi_C \models p_i = q_i$, we conclude by replacement of equals by equals that $\mathcal{E}(\Re) \cup \Psi_C \models p = q$.

• Q.E.D.