Canonical Conditional Rewrite Systems Containing Extra Variables*

Jürgen Avenhaus · Carlos Loría-Sáenz
Fachbereich Informatik, Universität Kaiserslautern
6750 Kaiserslautern (Germany)
E-mail: {avenhaus , loria}@informatik.uni-kl.de

March 1, 1993

Abstract

We study deterministic conditional rewrite systems, i.e. conditional rewrite systems where the extra variables are not totally free but 'input bounded'. If such a system R is quasi-reductive then $\rightarrow_R$ is decidable and terminating. We develop a critical pair criterion to prove confluence if R is quasi-reductive and strongly deterministic. In this case we prove that R is logical, i.e. $\overset{*}{\longleftrightarrow}_R = =_R$ holds. We apply our results to prove Horn clause programs to be uniquely terminating.

*This research was supported by the Deutsche Forschungsgemeinschaft, SFB 314, Project D4

1 Introduction

Conditional rewrite systems are widely used as a high-level language to write functional programs. This may cause non-deterministic computations. So one wants to prove that such a system R is canonical, i.e. terminating and confluent. This guarantees that for any input all possible computations stop and give the same result. There are well-known methods to prove termination and confluence if no extra variables are allowed. See [DO90] for a survey.

Functional programming naturally demands for the *where*-construct and this construct can be incorporated into the rewrite system approach only by allowing extra variables. But extra variables should be allowed only in a very restricted form since it is not clear how to instantiate them when only the variables in the left-hand side of a rule are instantiated for rewriting. So in this paper we restrict to deterministic rewrite rules (see [Ga91] and [BG89] for this notion): We require that the extra variables are 'input bounded'. In [Ga91] it is proved that $\rightarrow_R$ is decidable and terminating if R is quasi-reductive. We prove that $\rightarrow_R$ is confluent if R is in addition strongly deterministic and all proper critical pairs are joinable. Note that no paramodulation pairs (overlapping into the conditions) and no resolution pairs (factoring of a condition) need to be computed. These pairs may be harmful for arbitrary conditional rewrite systems with extra variables [Ga91], [De91]. As far as critical pairs are concerned, we neither need to consider variable overlappings nor overlappings of a rule with itself on top level. (Both are needed if R is not strongly deterministic.).

For many strongly deterministic rewrite systems R encountered in practice it can be proved that all proper critical pairs are either unfeasible or context-joinable. Then R will be confluent, provided it is quasi-reductive.

If R is a standard conditional rewrite system in the sense of [DO90] and confluent then R is logical, i.e. $\stackrel{*}{\longleftrightarrow}_R$ equals the R-equality $=_R$. This is not true for a strongly deterministic R, one needs in addition the termination of R or a restriction on the right-hand sides of the condition in the rules that is more restrictive than strong determinism.

The class of strongly deterministic rewrite systems seems to be interesting for two reasons. First, interesting problems can be specified rather naturally. And second, well-moded Horn clause programs can be translated into this class of rewrite systems (see [GW92]). We show how to prove that a well-moded program is uniquely terminating: Any derivation starting with a well-moded query stops and all refutations give the same answer substitution.

The paper is organized as follows: In section 2 we give the basic notations. We discuss the condition 'quasi-reductive' in section 3 and confluence in section 4. We show that $\rightarrow_R$ is logical under the conditions mentioned above in section 5 and we prove logic programs to be uniquely terminating in section 6.

Our interest in studying deterministic rewrite systems stems from problems arising in the field of program synthesis [LS92]. Here conditional rewrite systems with extra variables naturally appear.

The results of section 3 are basically contained in [BG89]. We present them here to make the paper self-contained. Beyond that we present an interesting method to prove quasi-reductivity. This is done by incorporating given inequalities in the definition of the recursive path ordering in order to compare terms with extra variables.

Some of the results presented in section 4 are also implicitely contained in [BG89]. The main differences are: (1) We do not want to start a completion algorithm but we want a simple

test on confluence for a given deterministic quasi-reductive rewrite system. We believe that many 'natural specifications' occurring in practice are really confluent (or at least ground confluent). (2) We do not want to use the concept of a non-operational equation. (3) To test confluence we prove that critical pairs resulting from overlapping a rule by itself on top-level need not be considered.

In [BG89] a result is reported that every absolutely deterministic, quasi-reductive and confluent R is logical. Here we show that one either needs additional restrictions on the right-hand sides of the conditions in a rule and no termination or termination and only strong determinism for a confluent R to be logical.

To study unique termination of well-moded logic programs we follow [HA85] and [GW92]. In [GW92] only termination of derivations in well-moded logic programs is studied. We use the translation of well-moded logic programs into deterministic rewrite systems from that paper to extend results from [HA85].

2 Basic notations

We assume the reader to be familiar with basic rewriting techniques and notations. For survey papers we refer to [AM90] and [DJ90] and especially for conditional rewriting to [DO90].

A *signature* is a triple $sig = (S, \mathcal{F}, \tau)$. Here S is a set of sorts, $\mathcal{F}$ a set of operators and τ a function τ: $\mathcal{F} \rightarrow S^+$ denoting the arity of the operators. $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is the set of terms over $\mathcal{F}$ and a set $\mathcal{V}$ of variables. A term is *ground* if it contains no variable. For a term t we denote by $O(t)$ the set of positions p in t such that t/p is not a variable. We denote by $t[s]_p$ the term that results from t by replacing t/p by s. We write $\equiv$ for the syntactic identity of terms. $Var(o)$ is the set of variable occurring in an object (term, equation, ...) o.

A partial ordering $\succ$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is *well-founded* if there is no infinite sequence $t_0 \succ t_1 \succ t_2 \succ \cdots$. It is *compatible with substitutions* (the term structure), if $s \succ s'$ implies $\sigma(s) \succ \sigma(s')$ for any substitution σ (respectively, $t[s]_p \succ t[s']_p$ for any term t and position p in t). A *reduction ordering* is a partial ordering that is well-founded, compatible with substitutions and compatible with the term structure. We denote by $\rhd$ the proper subterm relation and by $\succ_{st} = (\succ \cup \rhd)^+$ the smallest ordering that contains $\succ$ and $\rhd$. It is well-founded if $\succ$ is well-founded and compatible with the term structure. There are well-known methods to construct reduction orderings, we mention the recursive path ordering RPO and the lexicographic path ordering LPO. For these orderings we have $\succ = \succ_{st}$. For a survey paper on orderings see [De87].

We study conditional rewrite systems with extra variables. In this case it is convenient to consider oriented conditions. An *unconditional oriented equation* is a pair of terms, written $u \rightarrow v$. An *oriented condition* C is a finite sequence of unconditional oriented equations $C \equiv u_1 \rightarrow v_1, \ldots, u_n \rightarrow v_n$. It is called *operational* for a set V_0 of variables if

$$Var(u_i) \subseteq V_0 \cup Var(u_1 \rightarrow v_1) \cup \ldots \cup Var(u_{i-1} \rightarrow v_{i-1})$$

for $i = 1, \ldots, n$. (This implies $Var(u_1) \subseteq V_0$). Let R be a rewrite system such that $\rightarrow_R$ is already defined. A substitution σ is a *solution* of $C \equiv u_1 \rightarrow v_1, \ldots, u_n \rightarrow v_n$ wrt. R and V_0 if

$$\begin{array}{lll} \sigma(u_i) \stackrel{*}{\rightarrow}_R \sigma(v_i) & \text{for all} & 1 \leq i \leq n \\ Var(\sigma(x)) \subseteq Var(\sigma(V_0)) & \text{for all} & x \in Var(C) \end{array}$$

σ is *irreducible* if $\sigma(x)$ is irreducible for all $x \in V$. σ *extends* τ with $Dom(\tau) = V_0$ if $\sigma(x) \equiv \tau(x)$ for all $x \in V_0$. If C is empty then every σ is a solution of C wrt. R and V_0.

Note that the problem to compute all solutions of C that extend τ reduces to rewriting and matching. So, if $\rightarrow_R$ is computable and terminating then the set of solutions of C wrt. to R and V_0 that extend τ is finite and computable.

Example 2.1

$$\begin{array}{lllllll} R: & 0+y & \rightarrow & y & 0*y & \rightarrow & 0 \\ & s(x)+z & \rightarrow & x+s(y) & s(x)*y & \rightarrow & y+(x*y) \end{array}$$

$C \equiv x + y \rightarrow x' + y',\ x' * y' \rightarrow z$

Let $V_0 = \{x, y\}$ and $\tau = \{x \leftarrow s^n(0), y \leftarrow s^m(0)\}$. Then for all solutions σ extending τ we have $\sigma(z) \in \{s^k(0) \mid k = i \cdot j, i + j = n + m\}$.

Definition 2.1 *A* deterministic rule *is a formula* $C \Longrightarrow l \rightarrow r$ *such that* $C \equiv u_1 \rightarrow v_1, \ldots, u_n \rightarrow v_n$ *is operational for* $Var(l)$ *and* $Var(r) \subseteq Var(l) \cup Var(C)$ *and* l *is not a variable. The set of* extra variables *of this rule is* $\mathcal{E}Var(C \Longrightarrow l \rightarrow r) = Var(C) - Var(l)$. *A* deterministic term rewrite system *(DTRS) is a finite set* R *of deterministic rules.*

We simply write $l \rightarrow r$ instead of $C \Longrightarrow l \rightarrow r$ if C is empty. We next define the rewrite relation $\rightarrow_R$ for a DTRS R.

Definition 2.2 *Let* R *be a DTRS. The rewrite relation* $\rightarrow_R$ *is the smallest relation satisfying* $t[\sigma(l)]_p \rightarrow t[\sigma(r)]_p$ *whenever* $C \Longrightarrow l \rightarrow r$ *is a rule in* R *and* σ *is a solution of* C *wrt.* R *and* $Var(l)$.

Note that $s \rightarrow_R t$ implies $Var(t) \subseteq Var(s)$. Note also that $\rightarrow_R$ is defined recursively. To make this recursion explicit we define the *approximation* of R as an infinite sequence $(R_i)_{i \geq 0}$ of unconditional rewrite systems

$$\begin{array}{lcl} R_0 & = & \{l \rightarrow r \mid l \rightarrow r \text{ in } R\} \\ R_{i+1} & = & R_i \cup \{\sigma(l) \rightarrow \sigma(r) \mid C \Longrightarrow l \rightarrow r \text{ in } R, \sigma \text{ a solution of } C \text{ wrt. } R_i \text{ and } Var(l)\} \end{array}$$

We have $s \rightarrow_R t$ iff $s \rightarrow_{R_i} t$ for some $i \geq 0$.

We have defined a DTRS with oriented conditions. Such a system is called a normal conditional rewrite system in [D090]. There are some othere ways to evaluate the conditions. If a condition $u = v$ is to be evaluated by a joinability test then we write $u \downarrow v$. A system with a joinability test for the conditions is called a standard conditional rewrite system in [DO90]. For functional programming it is reasonable to allow both sorts of conditions, one for testing and the other one for computing. So a rule would be of the form

$$u_1 \rightarrow v_1, \ldots, u_n \rightarrow v_n, s_1 \downarrow t_1, \ldots, s_m \downarrow t_m \Longrightarrow l \rightarrow r$$

The conditions $s_i \downarrow t_i$ can easily be transformed into oriented conditions $s_i \rightarrow x_i, t_i \rightarrow x_i$ using new extra variables x_i. For this reason we only allow oriented conditions. In the next section we will consider quasi-reductive rules only in order to have $\rightarrow_R$ decidable and terminating. For standard conditional rewriting the conditions 'reductive' and 'decreasing' are needed for the same purpose. It will be easy to see that, given a decreasing standard

rule $s_1 \downarrow t_1, \ldots, s_m \downarrow t_m \Longrightarrow l \to r$ with no extra variables, then the transformed rule $s_1 \to x_1, t_1 \to x_1, \ldots, s_m \to x_m, t_m \to x_m \Longrightarrow l \to r$ is deterministic and quasi-reductive. So our approach is a generalization of standard conditional rewriting.

For standard conditional rewriting we have: If $s \to_R t$ then (i) $\sigma(s) \to_R \sigma(t)$ for any substitution σ and (ii) $t_0[s]_p \to t_0[t]_p$ for any position p in t_0. This holds also if R is a DTRS. The claim (ii) is trivial and (i) is proved by induction on i for the approximation $(R_i)_{i \geq 0}$ of R: If $s \to_{R_i} t$ then $\sigma(s) \to_{R_i} \sigma(t)$. We need this fact in the proof of Theorem 4.1.

3 Quasi-reductive DTRSs

Now we impose a condition on DTRSs R so that $\to_R$ is computable and terminating [GW92].

Definition 3.1
Let $\succ$ be a reduction ordering on $\mathcal{T}(\mathcal{F}, \mathcal{V})$. A DTRS R is quasi-reductive *wrt. $\succ$ if for every substitution σ and every rule $u_1 \to v_1, \ldots, u_n \to v_n \Longrightarrow l \to r$ in R*

(i) $\sigma(u_j) \succeq \sigma(v_j)$ *for* $1 \leq j \leq i$ *implies* $\sigma(l) \succ_{st} \sigma(u_{i+1})$

(ii) $\sigma(u_j) \succeq \sigma(v_j)$ *for* $1 \leq j \leq n$ *implies* $\sigma(l) \succ \sigma(r)$

A DTRS R is called quasi-reductive if there is a reduction ordering $\succ$ such that R is quasi-reductive wrt. $\succ$.

Now we prove that a quasi-reductive DTRS can be used for effective computations. We make this precise in Theorem 3.1 (see [GW92]). Let $\Delta^\star_R(t) = \{s \mid t \xrightarrow{\star}_R s\}$ denote the set of R-successors of t.

Theorem 3.1 *Let R be a DTRS that is quasi-reductive wrt. $\succ$. Then for evey term t the set $\Delta^\star_R(t)$ is finite and effectively computable. We have $\to_R \subseteq \succ$, so R is terminating.*

Proof: Let R be quasi-reductive wrt. $\succ$. We prove the statement of the Theorem by induction on $\succ_{st}$. We have $t \to_R s$ if there is a position p in t, a rule $C \Longrightarrow l \to r$ in R and a substitution σ such that $\sigma(l) \equiv t/p$ and σ is a solution of C wrt. R and $Var(l)$. Since R is finite it is enough to prove that for each rule $C \Longrightarrow l \to r$ there are only finitely many such solutions and one can compute them all.

Let $C \equiv u_1 \to v_1, \ldots, u_n \to v_n$ and τ be given such that $\tau(l) \equiv t/p$ and $Dom(\tau) = Var(l)$. We need to compute all solutions σ of C that extend τ. Let $\sigma_0 = \tau$. Since R is quasi-reductive we have $Var(u_1) \subseteq Var(l)$ and $\sigma_0(l) \succ_{st} \sigma_0(u_1)$. By induction hypothesis $\Delta^\star_R(\sigma_0(u_1))$ is finite and computable. So we can compute all matches $\sigma'(v_1) \equiv w, w \in \Delta^\star_R(\sigma_0(u_1))$, σ' extends σ_0. Let σ_1 be such a match, then σ_1 is a solution of $u_1 \to v_1$ and $\sigma_1(u_1) \succeq \sigma_1(v_1)$ and so $\sigma_1(l) \equiv \tau(l) \succ_{st} \sigma_1(u_2)$. In this way one can compute all solutions σ_i of $u_1 \to v_1, \ldots, u_i \to v_i$ extending τ, and we have $\sigma_i(u_j) \succeq \sigma_i(v_j)$ for $j = 1, \ldots, i$, so $\sigma_i(l) \succ_{st} \sigma_i(u_{i+1})$. For $i = n$ we also have $\tau(l) \equiv \sigma_n(l) \succ \sigma_n(r)$.

This proves that the set $\{s \mid t \to_R s\}$ is finite and computable and that $\to_R \subseteq \succ$ holds. So $\Delta^\star_R(t)$ is finite and computable, too. □

As a by-product of this proof we get

Corollary 3.1 *Let R be a DTRS that is quasi-reductive wrt. $\succ$. If $\sigma(l) \to_R \sigma(r)$ by $\rho \equiv u_1 \to v_1, \ldots, u_n \to v_n \Longrightarrow l \to r$ in R then $\sigma(l) \succ_{st} \sigma(u_i) \underline{\succeq} \sigma(v_i)$ for all $1 \leq i \leq n$.* □

We now discuss how to prove that a DTRS is quasi-reductive. The first approach is to eliminate in a deterministic rule $\rho \equiv u_1 \to v_1, \ldots, u_n \to v_n \Longrightarrow l \to r$ the extra variables by backward substitution and then to apply a test for reductivity on the resulting rule. To do so we define the transformed rule $\overline{\rho}$ of ρ as follows: For $x \in \mathcal{E}Var(\rho)$ let $\alpha(x)$ be the smallest i such that $x \in Var(v_i)$. We simultaneously define terms $\overline{u}_i$ and substitutions φ_i by

$$\begin{array}{lcl} \varphi_1 & = & id \\ \varphi_{i+1} & = & \{x \leftarrow \overline{u}_{\alpha(x)} \mid x \in Var(v_1, \ldots, v_i) \cap \mathcal{E}Var(\rho)\} \\ \overline{u}_i & \equiv & \varphi_i(u_i) \end{array}$$

Definition 3.2 *Let $\rho \equiv u_1 \to v_1, \ldots, u_n \to v_n \Longrightarrow l \to r$ be a deterministic rule. The backward substituted rule $\overline{\rho}$ is $\overline{u}_1 \to c, \ldots, \overline{u}_n \to c \Longrightarrow l \to \overline{r}$ where c is a new constant and $\overline{r} \equiv \varphi_{n+1}(r)$.*

Lemma 3.1 *Let $\succ$ be a reduction ordering. Let R be a DTRS such that for every rule ρ in R and its backward substituted form $\overline{\rho} \equiv \overline{u}_1 \to c, \ldots, \overline{u}_n \to c \Longrightarrow l \to \overline{r}$ we have*

$$l \succ_{st} \overline{u}_i \quad \textit{for} \quad 1 \leq i \leq n \quad \textit{and} \quad l \succ \overline{r}$$

then R is quasi-reductive wrt. $\succ$.

Proof: One verifies the conditions in Definition 3.1 by induction on i. □

Example 3.1

a) $\rho \equiv f(x) \to pair(y_1, y_2),\ y_1 + y_2 \to y_3 \Longrightarrow f(s(x)) \to pair(y_3, y_2)$.
Backward substitution gives $\overline{\rho} \equiv f(x) \to c, f(x) + f(x) \to c \Longrightarrow f(s(x)) \to pair(f(x) + f(x), f(x))$. *Let $\succ$ be the RPO with precedence $f > pair, +$. Then $\succ = \succ_{st}$ and*
$f(s(x)) \succ f(x),\ f(x) + f(x),\ pair(f(x) + f(x), f(x))$.
Hence ρ is quasi-reductive wrt. $\succ$.

b) $\rho \equiv f(x) \to z \Longrightarrow f(s(x)) \to f(z)$.
We have $\overline{\rho} \equiv f(x) \to c \Longrightarrow f(s(x)) \to f(f(x))$. *Let $\succ$ be the RPO with precedence $s \succ f$. Then ρ is quasi-reductive wrt. $\succ$.*

Notice that for a many-sorted specification backward substitution may result in ill-sorted terms. But most of the standard orderings are defined on the well-sorted and the ill-sorted terms as well and so can be used in Lemma 3.1. This holds for examples for the RPOs, LPOs and the polynomial orderings.

There is a direct approach to extend some standard orderings so that they can be used to compare terms with extra variables and so to prove quasi-reductivity. We explain this for the RPO.

If $\mathcal{J} = \{u_i \geq v_i \mid i = 1, \ldots, n\}$ is a set of inequations then we denote by $\geq_{\mathcal{J}}$ the smallest quasi-ordering containing $\mathcal{J}$. If s is a term, then $\mathcal{J}(s) = \{s \geq t \mid t \text{ a subterm of } s\}$ is the set of subterm-inequations defined by s.

Definition 3.3 *Let $\gtrsim$ be a precedence on $\mathcal{F}$ and $\mathcal{J}$ a set of inequations. Then the RPO $\underline{\succeq}_{\mathcal{J}}$ based on $\gtrsim$ and $\mathcal{J}$ is given by*

1. $s \underline{\succeq}_{\mathcal{J}} x \quad$ *if* $s \geq_{\mathcal{J} \cup \mathcal{J}(s)} x$

2. $s \equiv f(s_1, \ldots, s_n) \underline{\succeq}_{\mathcal{J}} t \equiv g(t_1, \ldots, t_m)$
   if $s_i \underline{\succeq}_{\mathcal{J}} t$ *for some* i *or*
   $f > g$ *and* $s \succ_{\mathcal{J}} t_j$ *for all* j *or*
   $f \approx g$ *and* $\{s_1, \ldots, s_n\} \underline{\succeq}_{\mathcal{J}} \underline{\succeq}_{\mathcal{J}} \{t_1, \ldots, t_m\}$

Here $\underline{\succeq}_{\mathcal{J}} \underline{\succeq}_{\mathcal{J}}$ denotes the multiset extension of $\underline{\succeq}_{\mathcal{J}}$ and $\succ_{\mathcal{J}}$ denotes the strict part of $\underline{\succeq}_{\mathcal{J}}$, i.e. $s \succ_{\mathcal{J}} t$ if $s \underline{\succeq}_{\mathcal{J}} t$, but not $t \underline{\succeq}_{\mathcal{J}} s$.

One easily proves

Lemma 3.2 *Let $\succ$ be the RPO based on a precedence $\gtrsim$ and let $\underline{\succeq}_{\mathcal{J}}$ be the RPO based on $\gtrsim$ and $\mathcal{J}$. If $s \underline{\succeq}_{\mathcal{J}} t$ then $\sigma(s) \underline{\succeq} \sigma(t)$ for all substitutions σ with $\sigma(u) \underline{\succeq} \sigma(v)$ for all $u \geq v$ in $\mathcal{J}$.* □

Let $\rho \equiv u_1 \to v_1, \ldots, u_n \to v_n \Longrightarrow l \to r$ be a rule, let $\mathcal{J}_j = \{u_i \geq v_i \mid i = 1, \ldots, j-1\}$ and let $\gtrsim$ be a precedence on $\mathcal{F}$ and let $\succ$ (resp. $\underline{\succeq}_{\mathcal{J}}$) be the RPO based on $\gtrsim$ (and $\mathcal{J}$). We say that ρ is *compatible* with $\succ$ if there are terms t, t', t_i, t_i' such that

$$\begin{array}{ll} l \underline{\succeq}_{\mathcal{J}} t_i \succ t_i' \underline{\succeq}_{\mathcal{J}} u_i & \text{for } i = 1, \ldots, n \\ l \underline{\succeq}_{\mathcal{J}} t \succ t' \underline{\succeq}_{\mathcal{J}} r & \end{array}$$

Now Lemma 3.2 immediately gives the following generalization of Lemma 3.1.

Theorem 3.2 *Let R be a DTRS and $\succ$ an RPO. If every rule in R is compatible with $\succ$ then R is quasi-reductive wrt. $\succ$.* □

Example 3.2
$R: \; f(x) \to f(s(y)) \Longrightarrow f(s(x)) \to f(s(y))$
We have $f(s(s(x)) \to_R f(s(x))$, so $\to_R$ is not empty.
Trying backward substitutions we have to prove $f(s(x)) \succ f(x)$ and $f(s(x)) \succ f(s(f(x)))$ which holds for no reduction ordering. Applying Theorem 3.2 we have to prove with $\mathcal{J} = \{f(x) \geq f(s(y))\}$

$$f(s(x)) \succ f(x) \quad \text{and} \quad f(s(x)) \succ f(x) \underline{\succeq}_{\mathcal{J}} f(s(y))$$

This holds for the RPO based on the empty precedence.

For another method to prove quasi-reductivity we refer to [GW92]. For a method to decide whether R is quasi-reductive wrt. to a given LPO we refer to [Co90].

4 Confluence of a DTRS

We study under which conditions a DTRS R is confluent. We assume that R is a quasi-reductive. In this case $\rightarrow_R$ is terminating and so it is sufficient to prove local confluence. The criterion to be developed for proving local confluence is based on critical pairs. We start with examples to demonstrate the problems arising from extra variables.

Example 4.1

a) R:

$$\begin{array}{rl} & 0 + y \rightarrow y \\ & s(x) + y \rightarrow x + s(y) \\ x + y \rightarrow z + z' \Longrightarrow & f(x, y) \rightarrow z \end{array}$$

We have $f(s(0), 0) \rightarrow_R s(0)$ *and* $f(s(0), 0) \rightarrow 0$, *but not* $s(0) \downarrow_R 0$. *This example shows that a critical pair resulting from overlapping a rule with itself at top-level may be harmful. We call such a critical pair improper.*

b) R:

$$\begin{array}{rlll} & a & \rightarrow & c \\ & g(a) & \rightarrow & h(b) \\ & h(b) & \rightarrow & g(c) \\ g(x) \rightarrow h(z) \Longrightarrow & f(x) & \rightarrow & z \end{array}$$

We have $f(c)_R \leftarrow f(a) \rightarrow_R b$ *but not* $f(c) \downarrow_R b$. *This example shows that variable overlappings may be harmful.*

Notice that in both examples R is a quasi-reductive DTRS. Since we do not want to consider variable overlappings and improper critical pairs we have to look for additional conditions that make them harmless.

Definition 4.1 *Let R be a DTRS.*
a) A term v is strongly irreducible *wrt. R if $\sigma(v)$ is irreducible for every irreducible substitution σ.*
b) R is called strongly deterministic *if for every rule $u_1 \rightarrow v_1, \ldots, u_n \rightarrow v_n \Longrightarrow l \rightarrow r$ in R each v_i is strongly irreducible wrt. R.*

Notice that both DTRSs in Example 4.1 are not strongly deterministic. The following system R for computing the Fibonacci numbers is strongly deterministic.

Example 4.2

R:

$$\begin{array}{rcl} & & fib(0) \rightarrow pair(s(0), 0) \\ fib(x) \rightarrow pair(y_2, y_1),\ y_1 + y_2 \rightarrow y_3 & \Longrightarrow & fib(s(x)) \rightarrow pair(y_3, y_2) \end{array}$$

Definition 4.2 *Let R be a DTRS and $C_1 \Longrightarrow l_1 \rightarrow r_1$ and $C_2 \Longrightarrow l_2 \rightarrow r_2$ be rules in R that have no variables in common. Let $p \in O(l_1)$ such that $\sigma = mgu(l_2, l_1/p)$ exists. Then*

$$\sigma(C_1), \sigma(C_2) \Longrightarrow \sigma(l_1[r_2]_p) = \sigma(r_1)$$

is a critical pair. *It is called* improper *if the rules differ only by a variable renaming and* $l_1 \equiv l_1/p$, *otherwise it is called* proper. *Let* $CP(R)$ *denote the set of proper critical pairs that can be built by pairs of rules in* R.
A critical pair $C \Longrightarrow s = t$ *resulting from* $C_1 \Longrightarrow l_1 \to r_1$ *and* $C_2 \Longrightarrow l_2 \to r_2$ *is* joinable *if* $\tau(s) \downarrow_R \tau(t)$ *for each solution* τ *of* C *wrt.* R *and* $Var(l_1, l_2)$.

Now we prove that a strongly deterministic and quasi-reductive DTRS is confluent if all proper critical pairs are joinable. Note that there are two problems related to this result: (i) Given a term v, is it strongly irreducible ? (ii) Given a conditional equation $C \Longrightarrow s = t$, is it joinable ? Unfortunately, both of these problems are undecidable for a deterministic and quasi-reductive R. But we will develop tools to prove that v is strongly irreducible and $C \Longrightarrow s = t$ is joinable wrt. R.

Theorem 4.1 *Let* R *be a DTRS that is strongly deterministic and quasi-reductive wrt.* $\succ$. R *is confluent iff all critical pairs in* $CP(R)$ *are joinable.*

Proof: Clearly, if R is confluent then all critical pairs in $CP(R)$ are joinable. So we now assume that all critical pairs in $CP(R)$ are joinable and prove by induction on $\succ_{st}$: If $t' \ {}_R\!\overset{*}{\longleftarrow} t \overset{*}{\longrightarrow}_R t''$ then $t' \downarrow_R t''$.
If $t' \equiv t$ or $t'' \equiv t$ then $t' \downarrow_R t''$ holds. So assume $t' \ {}_R\!\overset{*}{\longleftarrow} t_1 \ {}_R\!\longleftarrow t \longrightarrow_R t_2 \overset{*}{\longrightarrow}_R t''$. We will prove $t_1 \downarrow_R t_2$, then an inductive argument easily gives $t' \downarrow_R t''$.

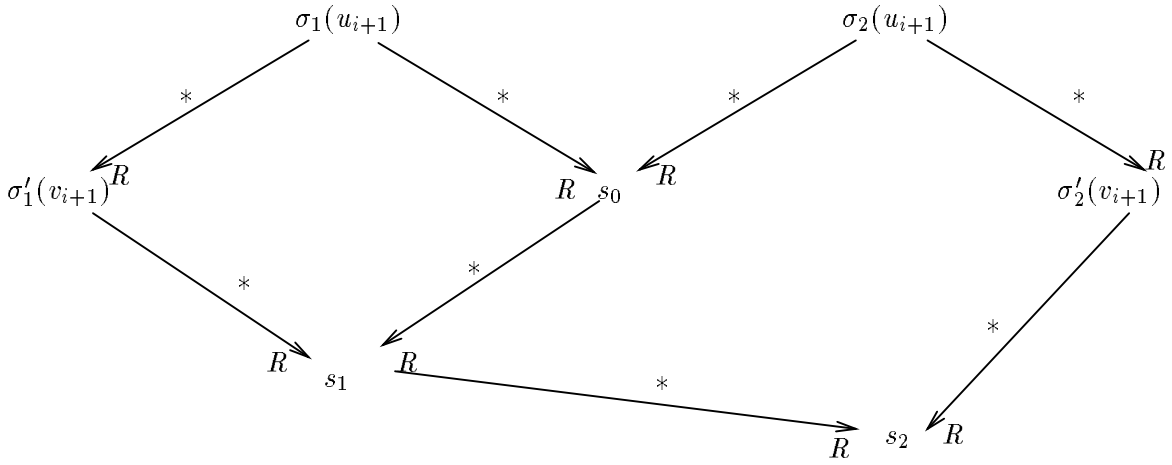
Assume

$t \to_R t_1$ using $C_1 \Longrightarrow l_1 \to r_1$, σ_1, position q in t
$t \to_R t_2$ using $C_2 \Longrightarrow l_2 \to r_2$, σ_2, position p in t

If the t/p and t/q are disjoint subterms of t then $t_1 \downarrow_R t_2$ trivially holds. So we may assume that t/p is a subterm of t/q. If $t \rhd t/q$, then $t \succ_{st} t/q$ and we have $t_1 \downarrow_R t_2$ by induction hypothesis on t/q. So we assume $t \equiv t/q$. Then we have $t \equiv \sigma_1(l_1)$ and $t/p \equiv \sigma_2(l_2)$. There are two cases: (α) $p \in O(l_1)$ and (β) p is a position in l_1 with l_1/p a variable or p is not a position in l_1.

(α) : In this case there is a critical pair $C \Longrightarrow s_1 = s_2$ and a substitution τ such that $t_1 \equiv \tau(s_1)$ and $t_2 \equiv \tau(s_2)$ and τ is a solution of C wrt. R and $Var(l_1, l_2)$. If this critical pair is proper then it is in $CP(R)$ and hence joinable. This gives $t_1 \downarrow t_2$. So assume that this critical pair is improper. Then $t \equiv \sigma_1(l_1) \equiv \sigma_2(l_2)$ and we may assume that $C_1 \Longrightarrow l_1 \to r_1$ and $C_2 \Longrightarrow l_2 \to r_2$ are identical, i.e. $C_i \Longrightarrow l_i \to r_i \equiv C \Longrightarrow l \to r$ for $i = 1, 2$. We have $\sigma_1(x) \equiv \sigma_2(x)$ for all $x \in Var(l)$ and we will prove $\sigma_1(x) \downarrow_R \sigma_2(x)$ for all $x \in Var(C \Longrightarrow l \to r)$. Since $t_1 \equiv \sigma_1(r)$ and $t_2 \equiv \sigma_2(r)$ this will prove $t_1 \downarrow_R t_2$.

Let $C = u_1 \to v_1, \ldots, u_n \to v_n$ and let σ_1', σ_2' be irreducible substitutions such that $\sigma_i(x) \overset{*}{\longrightarrow}_R \sigma_i'(x)$ for $i = 1, 2$ and all $x \in \mathcal{E}Var(C \Longrightarrow l \to r)$. It is enough to prove that $\sigma_1'(x) \equiv \sigma_2'(x)$ for all $x \in \mathcal{E}Var(C \Longrightarrow l \to r)$. If $x \in Var(u_1 \to v_1)$ then $x \in Var(v_1)$ since $Var(u_1) \subseteq Var(l)$. We have $\sigma_1'(v_1) \ {}_R\!\overset{*}{\longleftarrow} \sigma_1(u_1) \equiv \sigma_2(u_1) \overset{*}{\longrightarrow} \sigma_2'(v_1)$ and $\sigma_i'(v_1)$ is irreducible since σ_i' is irreducible. (Here we need that R is strongly deterministic and hence the v_j are strongly irreducible.) Since $t \succ_{st} \sigma_i(u_1)$ we have $\sigma_1'(v_1) \downarrow_R \sigma_2'(v_1)$ by induction hypothesis on $\sigma_1(u_1)$. This gives $\sigma_1'(v_1) \equiv \sigma_2'(v_1)$ and hence $\sigma_1'(x) \equiv \sigma_2'(x)$. Now assume $\sigma_1'(x) \equiv \sigma_2'(x)$ for all $x \in Var(u_1 \to v_1, \ldots, u_i \to v_i) - Var(l)$, we have to prove $\sigma_1'(x) \equiv \sigma_2'(x)$ for $x \in Var(u_{i+1} \to v_{i+1}) - Var(l)$. If $x \in Var(u_{i+1})$ then this is trivial since $C \Longrightarrow l \to r$ is deterministic. So let $x \in Var(v_{i+1})$. We have s_0 in the following picture since $\sigma_1(y) \downarrow_R \sigma_2(y)$ for all $y \in Var(u_{i+1})$.



The term s_1 exists by induction hypothesis on $\sigma_1(u_{i+1})$ since $t \succ_{st} \sigma_1(u_{i+1})$, and s_2 exists by induction hypothesis on $\sigma_2(u_{i+1})$. This proves $\sigma'(v_{i+1}) \downarrow_R \sigma_2'(v_{i+1})$ and now $\sigma_1'(x) \equiv \sigma_2'(x)$ follows as above.

(β) : In this case there is a variable $x \in Var(l_1)$ such that $\sigma_2(x) \rightarrow_R t_0$ with $C_2 \Longrightarrow l_2 \rightarrow r_2$ and σ_2. Define τ to be the substitution $\tau(y) \equiv \sigma_1(y)$ if $y \not\equiv x$ and $\tau(x) \equiv t_0$. Let τ' be an irreducible substitution with $\tau(y) \stackrel{*}{\longrightarrow}_R \tau'(y)$ for all y. Then we have

$$\begin{array}{lcl} t & \equiv & \sigma_1(l_1) \rightarrow_R t_1 \equiv \sigma_1(r_1) \stackrel{*}{\longrightarrow}_R \tau(r_1) \stackrel{*}{\longrightarrow}_R \tau'(r_1) \\ t & \equiv & \sigma_1(l_1) \rightarrow_R t_2 \equiv t[\sigma_2(r_2)]_p \stackrel{*}{\longrightarrow}_R \tau(l_1) \stackrel{*}{\longrightarrow}_R \tau'(l_1) \end{array}$$

(Here we assume that t and $C_1 \Longrightarrow l_1 \rightarrow r_1$ have no variables in common). We prove that τ' is a solution of C_1 wrt. R and $Var(l_1)$. Then we have $\tau'(l_1) \rightarrow_R \tau'(r_1)$ and so $t_1 \downarrow_R t_2$.

Let $C_1 = u_1 \rightarrow v_1, \ldots, u_n \rightarrow v_n$. We know that $\sigma_1(u_i) \stackrel{*}{\longrightarrow}_R \sigma_1(v_i)$ and have to prove $\tau'(u_i) \stackrel{*}{\longrightarrow}_R \tau'(v_i)$. We have $\tau'(u_i) \; {}_R\!\stackrel{*}{\longleftarrow} \tau(u_i) \; {}_R\!\stackrel{*}{\longleftarrow} \sigma_1(u_i) \stackrel{*}{\longrightarrow}_R \sigma_1(v_i) \stackrel{*}{\longrightarrow}_R \tau'(v_i)$. By induction hypothesis on $\sigma_1(u_i)$ – notice that $t \succ_{st} \sigma_1(u_i)$ holds by Corollary 3.3 – we have $\tau'(u_i) \downarrow \tau'(v_i)$ and hence $\tau'(u_i) \stackrel{*}{\longrightarrow}_R \tau'(v_i)$ since $\tau'(v_1)$ is irreducible. So τ' is indeed a solution of C_1 wrt. R and $Var(l_1)$. □

As mentioned above, it is undecidable whether a DTRS is strongly deterministic. But there is a sufficient condition that is easily testable.

Definition 4.3 *Let R be a DTRS*
a) A term v is absolutely irreducible wrt. R if for all $p \in O(v)$ and each rule $C \Longrightarrow l \rightarrow r$ in R v/p and l are not unifyable.
b) R is absolutely deterministic if for every rule $u_1 \rightarrow v_1, \ldots, u_n \rightarrow v_n \Longrightarrow l \rightarrow r$ in R each v_i is absolutely irreducible wrt. R.

In many applications a DTRS is used to define functions over 'free constructors'. If R is such a system and all right-hand sides of conditions in R are constructor terms then R is absolutely deterministic.

Lemma 4.1 *Let R be a DTRS.*
a) If v is absolutely irreducible wrt. R then v is strongly irreducible wrt. R.

b) If R is absolutely deterministic then R is strongly deterministic.
c) It is undecidable whether a quasi-reductive DTRS is strongly deterministic.

Proof:
a) Let σ be an irreducible substitution. If $\sigma(v)$ is reducible then there is a position $p \in O(v)$ and a rule $C \Longrightarrow l \to r$ in R such that l matches $\sigma(v/p)$, so l and v/p are unifyable. This is impossible if v is absolutely irreducible.
b) This follows from a).
c) We reduce Post's Correspondence Problem (PCP) to our problem. Let $\Sigma = \{\alpha_1, \ldots, \alpha_m\}$ be an alphabet and $A = a_1, \ldots, a_n$ and $B = b_1, \ldots, b_n$ two lists of words over Σ. Then

$$PCP(A,B) \quad \text{iff} \quad a_{i_1}a_{i_2}\ldots a_{i_k} \equiv b_{i_1}b_{i_2}\ldots b_{i_k} \text{ for some } 1 \le i_j \le n,\ j = 1, \ldots, n.$$

By abuse of notation we identify the letters α_i with unary function symbols $\alpha_i(x)$, then the words a_j and b_j become terms $a_j(x)$ and $b_j(x)$, respectively. We need a new constant 0 and new function symbols f, g, h, h' and use lists over $\{1, \ldots, n\}$. Let

$$\begin{array}{lllll} R: & f(nil) \to 0 & & g(nil) \to 0 & \\ & f(i.l) \to a_i(f(l)) & & g(i.l) \to b_i(g(l)) & 1 \le i \le n \\ & eq(x,x) \to true & & & \\ & eq(f(i.l), g(i.l)) \to true & \Longrightarrow & h(i.l) \to nil & 1 \le i \le n \\ & h(l) \to h(l) & \Longrightarrow & h'(l) \to l & \end{array}$$

Let R_0 consist of the rules in R except the last one. Then $v \equiv h(l)$ is not strongly irreducible wrt. R_0 iff $PCP(A,B)$ holds, so R is strongly deterministic iff $PCP(A,B)$ does not hold. Notice that R is a quasi-reductive DTRS. Since PCP is undecidable, part c) of the Lemma is proved. □

We now develop a sufficient criterion for confluence of a quasi-reductive and strongly deterministic DTRS. It is based on contextual rewriting. We use here a more restrictive version of contextual rewriting than that used in [BG89]. It is easier to implement and results in easier proofs.

Let $C = \{u_1 \to v_1, \ldots, u_n \to v_n\}$ be a set of oriented equations, called here a *context*. We denote by $\overline{C}$ its skolemized form, i.e. $\overline{C}$ results from C by replacing each variable x by a new constant $\overline{x}$. If t is a term, then $\overline{t}$ results from t by replacing each $x \in Var(C)$ by the constant $\overline{x}$. We write $s \to_{R,C} t$ if $\overline{s} \to_{R \cup \overline{C}} \overline{t}$. Obviously, we have

Lemma 4.2 *If $s \xrightarrow{*}_{R,C} t$ then $\sigma(s) \xrightarrow{*}_R \sigma(t)$ for every solution σ of C.* □

Definition 4.4 *Let R be a DTRS that is quasi-reductive wrt. $\succ$ and let $C \Longrightarrow s = t$ be a critical pair resulting from $C_i \Longrightarrow l_i \to r_i$ for $i = 1, 2$ and $\sigma = mgu(l_1/p, l_2)$. We call $C \Longrightarrow s = t$* unfeasible *if there are terms t_0, t_1, t_2 such that $\sigma(l_1) \succ_{st} t_0$, $t_0 \xrightarrow{*}_{R,C} t_1$, $t_0 \xrightarrow{*}_{R,C} t_2$ and t_1, t_2 are not unifyable and strongly irreducible. We call $C \Longrightarrow s = t$* context-joinable *if there is some t_0 such that $s \xrightarrow{*}_{R,C} t_0$, $t \xrightarrow{*}_{R,C} t_0$.*

Note that in the definition of an unfeasible critical pair $C \Longrightarrow s = t$ we have $\sigma(l_1) \succ u$ if $u \to v$ in C and $Var(u) \subseteq Var(\sigma(l_1))$. So we may choose t_0 to be any such term u. In many cases a quasi-reductive and strongly deterministic DTRS can be proved to be confluent by the following Theorem.

Theorem 4.2 *Let R be a DTRS that is quasi-reductive wrt. $\succ$ and strongly deterministic. If every critical pair in $CP(R)$ is either unfeasible or context-joinable then R is confluent.*

Proof: We repeat the proof of Theorem 4.1. The only point we have to consider is the first case in (α): Let $C \Longrightarrow s_1 = s_2$ be a proper critical pair resulting from $C_i \Longrightarrow l_i \to r_i, i = 1, 2$, and $\sigma = mgu(l_1/p, l_2)$. There is a solution τ of C such that $t \equiv \tau\sigma(l_1)$ and $t_1 \equiv \tau(s_1), t_2 \equiv \tau(s_2)$. By induction hypothesis on t we may assume: If $s' \ {}_R\!\xleftarrow{*} s \xrightarrow{*}_R s''$ and $\tau\sigma(l_1) \succ_{st} s$ then $s' \downarrow_R s''$. Then we have to prove $t_1 \downarrow_R t_2$.

$C \Longrightarrow s_1 = s_2$ is not unfeasible: Otherwise there are t_0, t_0', t_0'' such that $\sigma(l_1) \succ_{st} t_0, t_0 \xrightarrow{*}_{R,C} t_0', t_0 \xrightarrow{*}_R t_0''$ and t_0', t_0'' are not unifyable and strongly irreducible. Since τ is a solution of C we have by Lemma 4.2 that $\tau\sigma(l_1) \succ_{st} \tau(t_0), \tau(t_0) \xrightarrow{*}_R \tau(t_0')$ and $\tau(t_0) \xrightarrow{*}_R \tau(t_0'')$. Let μ be an irreducible substitution such that $\tau(x) \xrightarrow{*}_R \mu(x)$ for all x. Then $\tau(t_0) \xrightarrow{*}_R \mu(t_0')$ and $\tau(t_0) \xrightarrow{*}_R \mu(t_0'')$ and $\mu(t_0') \not\equiv \mu(t_0'')$. By induction hypothesis on $\tau(t_0)$ we have $\mu(t_0') \downarrow_R \mu(t_0'')$. But this is impossible since $\mu(t_0')$ and $\mu(t_0'')$ are irreducible.

So $C \Longrightarrow s_1 = s_2$ is context-joinable. Since τ is a solution of C we have $t_1 \equiv \tau(s_1) \downarrow_R \tau(s_2) \equiv t_2$ by Lemma 4.2 □

Let $C \Longrightarrow s = t$ be a critical pair of some R. If $s \equiv t$ then it is called *trivial*. A trivial critical pair is always context-joinable. If C contains $u \to a$ and $u \to b$, where a and b are distinct irreducible constants, then $C \Longrightarrow s = t$ is unfeasible. But there are less trivial examples where Theorem 4.2 is applicable.

Example 4.3 *(see [BG89])*
We specify the Quicksort-algorithm

$R:$		$0 \le x \to true$	*(1)*
		$s(x) \le 0 \to false$	*(2)*
		$s(x) \le s(y) \to x \le y$	*(3)*
		$app(nil, l_2) \to l_2$	*(4)*
		$app(x.l_1, l_2) \to x.app(l_1, l_2)$	*(5)*
		$split(x, nil) \to pair(nil, nil)$	*(6)*
$x \le y \to false, split(x, l) \to pair(l_1, l_2)$	$\Longrightarrow$	$split(x, y.l) \to pair(y.l_1, l_2)$	*(7)*
$x \le y \to true, split(x, l) \to pair(l_1, l_2)$	$\Longrightarrow$	$split(x, y.l) \to pair(l_1, y.l_2)$	*(8)*
		$sort(nil) \to nil$	*(9)*
$split(x, l) \to pair(l_1, l_2)$	$\Longrightarrow$	$sort(x.l) \to app(sort(l_1), x.sort(l_2))$	*(10)*

There is only one proper critical pair, it is $C \Longrightarrow pair(y.l_1, l_2) = pair(l_1, y.l_2)$ with $C \equiv x \le y \to true, split(x, l) \to pair(l_1, l_2), x \le y \to false, split(x, l) \to pair(l_1', l_2')$ and it results from rules (7) and (8). Since C contains $x \le y \to false$ and $x \le y \to true$, this critical pair is unfeasible. R is strongly deterministic and R is quasi-reductive wrt. to a semantic path ordering [St93], [Ge92]. So we have a simple proof that R is confluent.

5 On the descriptive power of $\to_R$

Any conditional term rewriting system R can naturally be regarded as a set of conditional equations. So the R-equality $=_R$ is well-defined (for details see below). If R is a standard

system without extra variables then the following fact is well-known: If R is confluent then $\stackrel{*}{\longleftrightarrow}_R = =_R$. We prove a similar result for strongly deterministic systems. We start with an example to demonstrate the problems arising from directed conditions.

Example 5.1

R:

$$g(x) \to h(x)$$
$$h(x) \to g(y) \implies f(x) \to y$$

Let $\succ$ be the RPO with precedence $f > g > h$. Then R is quasi-reductive wrt. $\succ$. We have $CP(R) = \emptyset$, so R is terminating and confluent. We have $f(x) =_R x$ but not $f(x) \downarrow_R x$. So $\stackrel{*}{\longleftrightarrow}_R$ is properly included in $=_R$. Notice that R is not strongly deterministic.

We now formally define $=_R$.

A *conditional equation* over a signature $sig = (S, \mathcal{F}, \tau)$ is a formula of the form

$$u_1 = v_1, \ldots, u_n = v_n \implies s = t$$

A *conditional equational system* E is a set of conditional equations. We are going to define $=_E$.

Let E be a conditional equational system and let G be a set of unconditional equations. We first define an inference system depending on E and G. It consists of the following five rules

Reflexivity	$\vdash t = t$
Symmetry	$t_1 = t_2 \vdash t_2 = t_1$
Transitivity	$t_1 = t_2, t_2 = t_3 \vdash t_1 = t_3$
Congruence	$t_1 = s_1, \ldots, t_n = s_n \vdash f(t_1, \ldots, t_n) = f(s_1, \ldots, s_n)$
	if $f(t_1, \ldots, t_n), f(s_1, \ldots, s_n)$ are terms
E-Application	$\sigma(u_1) = \sigma(v_1), \ldots, \sigma(u_n) = \sigma(v_n) \vdash \sigma(s) = \sigma(t)$
	if $\sigma(u_i) = \sigma(v_i)$ in G for $i = 1, \ldots, n$
	$u_1 = v_1, \ldots, u_n = v_n \implies s = t$ in E

We write $G \vdash_E u = v$ if $u = v$ can be deduced by this inference system.

Next we define an infinite sequence $(E_i)_{i \geq 0}$ of unconditional equational systems as an *approximation* of E:

$$E_0 = \{u = v \mid \emptyset \vdash_E u = v\}$$
$$E_{i+1} = E_i \cup \{u = v \mid E_i \vdash_E u = v\}$$

Now

$u =_E v$ iff $u = v$ in some $E_i, i \geq 0$.

The relevance of this definition is based on Birkhoff's Theorem: $u =_E v$ iff $u = v$ holds in every model of E.

Now let R be a DTRS. We associate to R the conditional equational system

$$E(R) = \{u_1 = v_1, \ldots, u_n = v_n \implies l = r \mid u_1 \to v_1, \ldots, u_n \to v_n \implies l \to r \text{ in } R\}$$

and simply write $u =_R v$ instead of $u =_{E(R)} v$.

Theorem 5.1 *Let R be strongly deterministic, terminating and confluent. Then $s \overset{*}{\longleftrightarrow}_R t$ iff $s =_R t$.*

Proof: Let $(E_i)_{i\geq 0}$ be the approximation of $E(R)$ and $=_i \; = =_{E_i}$. Let $(R_i)_{i\geq 0}$ be the approximation of R and $\rightarrow_i = \rightarrow_{R_i}$.
a) $s \overset{*}{\longleftrightarrow}_R t$ implies $s =_R t$: Induction on i proves $\rightarrow_i \subseteq =_i$. This gives $\rightarrow_R \subseteq =_R$ and $\overset{*}{\longleftrightarrow}_R \subseteq =_R$.
b) $s =_R t$ implies $s \overset{*}{\longleftrightarrow}_R t$: We prove by induction on i that $=_i \subseteq \overset{*}{\longleftrightarrow}_R$ for all $i \geq 0$. Then $=_R \subseteq \overset{*}{\longleftrightarrow}_R$ immediately follows.
For $i = 0$ we trivially have $=_0 \; = \overset{*}{\longleftrightarrow}_0 \subseteq \overset{*}{\longleftrightarrow}_R$. Let $i > 0$. Then $=_i$ is the smallest congruence relation with $=_{i-1} \subseteq =_i$ and $\sigma(l) =_i \sigma(r)$ whenever $\rho \equiv u_1 \rightarrow v_1, \ldots, u_n \Longrightarrow l \rightarrow r$ is in R and $\sigma(u_j) =_{i-1} \sigma(v_j)$ for $1 \leq j \leq n$. Also $\overset{*}{\longleftrightarrow}_R$ is a congruence relation. So it is enough to prove $\sigma(l) \overset{*}{\longleftrightarrow}_R \sigma(r)$ if $\sigma(u_j) =_{i-1} \sigma(v_j)$ for $1 \leq j \leq n$. Let σ' be the irreducible substitution with $\sigma(x) \overset{*}{\longrightarrow}_R \sigma'(x)$ for all $x \in Var(\rho)$. By induction hypothesis we have $\sigma(u_j) \overset{*}{\longleftrightarrow}_R \sigma(v_j)$ and hence $\sigma'(u_j) \overset{*}{\longleftrightarrow}_R \sigma'(v_j)$. Since $\rightarrow_R$ is confluent and $\sigma'(v_j)$ is irreducible we get $\sigma'(u_j) \overset{*}{\longrightarrow}_R \sigma'(v_j)$ for $1 \leq j \leq n$. This implies $\sigma'(l) \rightarrow_R \sigma'(r)$ and hence we have $\sigma(l) \overset{*}{\longleftrightarrow}_R \sigma(r)$. □

We call R *logical* if $\overset{*}{\longleftrightarrow}_R$ equals $=_R$. By Theorem 5.1, R is logical if R is strongly deterministic, quasi-reductive and confluent. For some applications quasi-reductivity is a requirement that is too hard. The rewrite relation $\rightarrow_R$ may be computable but not terminating. So the question arises whether Theorem 5.1 holds true if R is allowed to be non-terminating. The proof of Theorem 5.1 shows that the answer is 'yes' if R is *weakly terminating*, i.e. for every t there is an irreducible term t' such that $t \overset{*}{\longrightarrow}_R t'$. The next example shows that the answer to the question is 'no' in general.

Example 5.2

$$\begin{array}{lll} R: & a \rightarrow b & f(a) \rightarrow a \qquad f'(a) \rightarrow a \\ & b \rightarrow a & f(b) \rightarrow a \qquad f'(b) \rightarrow a \\ & f'(x) \rightarrow f(x) \Longrightarrow g(x) \rightarrow x & \end{array}$$

R is a normal conditional rewrite system in the sense of [D090]. It is strongly deterministic since $f(x)$ is strongly irreducible, but R is not terminating and not absolutely deterministic. To prove confluence one easily shows by induction on $\rhd$: If $t_1 \; {}_R\!\longleftarrow \; t \longrightarrow_R t_2$ then there is a term s with $t_1 \overset{\leq 1}{\longrightarrow}_R s$ and $t_2 \overset{\leq 1}{\longrightarrow}_R s$. (Here $t \overset{\leq 1}{\longrightarrow} s$ denotes $t \equiv s$ or $t \rightarrow_R s$.) By Lemma 2.5 in [Hu80] this proves that $\rightarrow_R$ is indeed confluent. We have $f(a) =_R f'(a)$ and hence $g(a) =_R a$. But $g(a) \downarrow_R a$ does not hold. So R is not logical.

Now we prove that the result of Theorem 5.1 still holds if the termination assumption is replaced by stronger restrictions on the rules in R.

Theorem 5.2 *Let R be a confluent DTRS such that for every rule $u_1 \rightarrow v_1, \ldots, u_n \rightarrow v_n \Longrightarrow l \rightarrow r$ in R*

$$\begin{array}{l} v_i \text{ is absolutely irreducible and} \\ Var(v_i) \cap Var(u_1, \ldots, u_i) = \emptyset \end{array}$$

for $1 \leq i \leq n$. Then $s \overset{*}{\longleftrightarrow}_R t$ iff $s =_R t$.

Proof: We repeat the proof of Theorem 5.1. Part a) carries over directly. For b) we have to show:

Claim: If $\rho \equiv u_1 \to v_1, \ldots, u_n \to v_n \Longrightarrow l \to r$ in R and $\sigma(u_j) \downarrow_R \sigma(v_j)$ then there is a σ' such that $\sigma'(u_j) \xrightarrow{*}_R \sigma'(v_j)$ for $1 \leq j \leq n$ and $\sigma(x) \xrightarrow{*}_R \sigma'(x)$ for all $x \in Var(\rho)$.

From this claim we get $\sigma'(l) \to_R \sigma'(r)$ and $\sigma(l) \xleftrightarrow{*}_R \sigma(v)$. Then the proof of Theorem 5.1 can be carried over.

Let the assumption of the claim hold true. We will define σ' by induction on j. To do so we need the following fact: If v is absolutely irreducible and $\tau(v) \xrightarrow{*}_R w$ then $w \xrightarrow{*}_R \tau'(v)$ and $\tau(x) \xrightarrow{*}_R \tau'(x)$ for all $x \in Var(v)$ for some substitution τ'. This fact holds true even if v is non-linear since $\to_R$ is confluent. Together with $\sigma(u_j) \downarrow \sigma(v_j)$ it implies that there are substitutions σ'_j such that $\sigma(u_j) \xrightarrow{*}_R \sigma'_j(v_j)$.

We are going to define $\sigma_1, \ldots, \sigma_n$ such that $\sigma_j(u_i) \xrightarrow{*}_R \sigma_j(v_i)$ for $1 \leq i \leq j$ and $\sigma(x) \xrightarrow{*}_R \sigma_j(x)$ for $x \in Var(u_1, v_1, \ldots, u_j, v_j)$: For $j = 1$ we may choose $\sigma_1 = \sigma'_1$. Using σ_{j-1} we now have to define σ_j. We have $\sigma_{j-1}(u_j)\ {}_R\!\xleftarrow{*}\ \sigma(u_j) \xrightarrow{*}_R \sigma'_j(v_j)$, so there is σ''_j with $\sigma_{j-1}(u_j) \xrightarrow{*}_R \sigma''_j(v_j)$ since $\to_R$ is confluent and v_j is absolutely irreducible. If $x \in Var(v_j) \cap Var(v_1, \ldots, v_{j-1})$ then $\sigma_{j-1}(x)\ {}_R\!\xleftarrow{*}\ \sigma(x) \xrightarrow{*}_R \sigma''_j(x)$, so $\sigma_{j-1}(x) \xrightarrow{*}_R t_x\ {}_R\!\xleftarrow{*}\ \sigma''_j(x)$ for some t_x. Now we define σ_j by

$$\sigma_j(x) \equiv \begin{cases} \sigma_{j-1}(x) & x \notin Var(v_j) \\ \sigma''_j(x) & x \in Var(v_j) - Var(v_1, \ldots, v_{j-1}) \\ t_x & x \in Var(v_j) \cap Var(v_1, \ldots, v_{j-1}) \end{cases}$$

Then $\sigma_{j-1}(x) \xrightarrow{*}_R \sigma_j(x)$ for $x \in Var(u_1, v_1, \ldots, u_{j-1}, v_{j-1})$ and $\sigma(x) \xrightarrow{*} \sigma_j(x)$ for $x \in Var(u_1, v_1, \ldots, u_j, v_j)$. We have $\sigma_j(u_i) \xrightarrow{*}_R \sigma_j(v_i)$ for $1 \leq i < j$, since $Var(v_j) \cap Var(u_1, \ldots, u_j) = \emptyset$ and $\sigma_{j-1}(u_i) \xrightarrow{*}_R \sigma_{j-1}(v_i)$, and we have $\sigma_j(u_j) \xrightarrow{*} \sigma_j(v_j)$ by construction.

Now choosing $\sigma' = \sigma_n$ the claim is proved. This finishes the proof of the theorem. □

Theorem 5.2 holds if in each rule $u_1 \to v_1, \ldots, u_n \to v_n \Longrightarrow l \to r$ the v_i's are irreducible constants. One may ask whether the condition $Var(v_i) \cap Var(u_1, \ldots, u_i) = \emptyset$ can be dropped. The following example shows that then the Theorem may not hold, even if the v_i's are restricted to be variables. This indicates that conditional rewrite systems are very sensitive according to the introduction of extra-variables.

Example 5.3

R :

$$\begin{array}{rcl} & & b \to g(a,b) \\ b \to z, g(a,z) \to z & \Longrightarrow & f(a) \to c \end{array}$$

R is confluent, absolutely deterministic and not terminating. Taking $\sigma = \{z \leftarrow b\}$ one gets $f(a) =_R c$. But both terms $f(a)$ and c are irreducible, so $f(a) \downarrow c$ does not hold. Hence R is not logical. Note that Theorem 5.2 is not applicable to R only because of $v_2 \equiv z \in Var(u_2) = Var(g(a,z))$.

6 Uniquely terminating well-moded logic programs

We now apply the results of the previous sections to well-moded logic programs.

A logical program is a set of Horn-clauses. Normally such a program is used to describe a search in a search space. In this case no distinction is made between input and output

positions for the predicate symbols. But it has turned out that logic programs are also widely used to write programs in a functional manner. In this case each predicate symbol is assigned a fixed 'mode' to distinguish between input and output positions. To compute with such a moded program one starts with a query such that its input positions are filled by ground terms. Then, doing logical operations the information sweeps into the output positions.

So the question arises whether a well-moded logic program is uniquely terminating, i.e. whether for each input all possible computations stop and give the same result. In this section we develop means to assure this property.

This problem has been studied earlier by the first author [HA85]. The results presented here go far about those presented there. For example, we do not restrict to hierarchical specifications of the logic program. Here we follow the approach of [GW92]. In that paper only termination of computations in logic programs is studied. Here we are also interested in guaranteeing uniqueness of the results.

We start with some notations and definitions but we assume that the reader has some knowledge on logic programming and SLD-derivations.

A *signature* is a quadruple $sig_0 = (S, \mathcal{P}_0, \mathcal{F}_0, \tau_0)$ where S is a set of sorts, $\mathcal{F}_0$ is a set of function symbols, $\mathcal{P}_0$ is a set of predicate symbols and $\tau_0 : \mathcal{F}_0 \cup \mathcal{P}_0 \to S^+$ assigns to each function and predicate symbol its arity. If $P \in \mathcal{P}_0$ and $t_1, \ldots, t_n$ are terms then $P(t_1, \ldots, t_n)$ is an *atom*. A *Horn-clause* is a formula of the form $A \leftarrow B_1, \ldots, B_m$ where $m \geq 0$ and A, B_i are atoms. A *logic program* $\mathcal{P}$ is a set of Horn-clauses. A *query* is a formula of the form $\leftarrow B_1, \ldots, B_m$ where $m \geq 1$ and B_i are atoms.

A logic program $\mathcal{P}$ is *moded* if for every occurrence of an atom $A \equiv P(t_1, \ldots, t_n)$ there is a function $m_A : \{1, \ldots, n\} \to \{in, out\}$. If $m_A(i) = in$ ($m_A(i) = out$) then position i is called an *input position* (*output position*) of A. A variable x occurs in an input (output) position in A if $x \in Var(t_i)$ for some i with $m_A(i) = in$ ($m_A(i) = out$).

To evaluate a moded logic program $\mathcal{P}$ we only consider left-to-right SLD-derivations. They always select the left-most literal of a query for the next resolution step. So we restrict to LR-well-moded programs as defined next.

Definition 6.1 *a) Let $C \equiv A \leftarrow B_1, \ldots, B_m$ be a clause and $x \in Var(C)$. The head A of C is called a* producer (consumer) *of x, if x occurs in an input (output) position of A. The body atom B_j is called a producer (consumer) of x, if x occurs in an output (input) position of B_j.*
b) The clause $B_0 \leftarrow B_1, \ldots, B_m$ is called LR-well-moded, *if every variable x in the clause has a producer B_i ($0 \leq i \leq n$) such that for every consumer B_j of x we have $i < j$. A logic program $\mathcal{P}$ is* LR-well-moded *if every clause in $\mathcal{P}$ is LR-well-moded.*
c) A query $\leftarrow B_1, \ldots, B_m$ is LR-well-moded *if every variable x in the query has a producer B_i such that for every consumer B_j of x we have $i < j$.*

By this definition, if $\leftarrow B_1, \ldots, B_m$ is LR-well-moded and $B_1 \equiv P(t_1, \ldots, t_n)$ then t_i are ground terms for all input positions i of B_1. One easily proves [GW92]

Lemma 6.1 *Let $\mathcal{P}$ be an LR-well-moded logic program and $G_0, G_1, \ldots$ a left-to-right SLD-derivation starting with an LR-well-moded query G_0. Then all queries G_i are LR-well-moded, and the first atom of every non-empty G_i is ground on all its input positions.* □.

Now we associate to every logic program $\mathcal{P}$ over the signature $sig = (S, \mathcal{P}_0, \mathcal{F}_0, \tau_0)$ a DTRS $R(\mathcal{P})$ over the signature $sig = (S, \mathcal{F}, \tau)$. Here $\mathcal{F}$ consists of the $f \in \mathcal{F}_0$ and for each atom $A \equiv P(t_1, \ldots, t_n)$ with input positions $i_1, \ldots, i_k$ and output positions $i_{k+1}, \ldots, i_n$ the two function symbols Pin and $Pout$. We associate to the atom $A \equiv P(t_1, \ldots, t_n)$ the rule $\rho(A) \equiv Pin(t_{i_1}, \ldots t_{i_k}) \to Pout(t_{i_{k+1}}, \ldots, t_{i_n})$. We associate to a clause $C \equiv A \leftarrow B_1, \ldots, B_m$ the rule

$$\rho(C) \equiv \rho(B_1), \ldots, \rho(B_n) \Longrightarrow \rho(A)$$

and to $\mathcal{P}$ the system $R(\mathcal{P}) = \{\rho(C) \mid C \text{ in } \mathcal{P}\}$.

Example 6.1

$\mathcal{P}$: $APP(nil, l_2, l_2) \leftarrow$

$APP(x.l_1, l_2, x.l_3) \leftarrow APP(l_1, l_2, l_3)$

a) If all atoms have mode $m(1) = m(2) = in, m(3) = out$ *then*

$R(\mathcal{P})$

$$APPin(l_1, l_2) \to APPout(l_3) \quad \Longrightarrow \quad APPin(nil, l_2) \to APPout(l_2)$$
$$APPin(x.l_1, l_2) \to APPout(x.l_3)$$

b) If all atoms have mode $m(1) = m(2) = out, m(3) = in$ *then*

$R(\mathcal{P})$

$$APPin(l_3) \to APPout(l_1, l_2) \quad \Longrightarrow \quad APPin(l_2) \to APPout(nil, l_2)$$
$$APPin(x.l_3) \to APPout(x.l_1, l_2)$$

In both cases $R(\mathcal{P})$ is strongly deterministic and quasi-reductive. In case a) $R(\mathcal{P})$ is confluent, but in case b) $R(\mathcal{P})$ is not confluent.

The following lemma states that $R(\mathcal{P})$ is always absolutely deterministic if $\mathcal{P}$ is LR-well-moded.

Lemma 6.2
a) If $C = A \leftarrow B_1, \ldots, B_n$ is LR-well-moded then $\rho(C)$ is deterministic.
b) If $\mathcal{P}$ is LR-well-moded then $R(\mathcal{P})$ is absolutely deterministic.
c) If $\leftarrow B_1, \ldots, B_m$ is an LR-well-moded query and $\rho(B_1) \equiv Pin(t_1, \ldots, t_n) \to Pout(s_1, \ldots, s_k)$ then all t_i are ground terms.

Proof: a) This follows directly from the definitions.
b) $R(\mathcal{P})$ is deterministic by a). If $u_1 \to v_1, \ldots, u_n \to v_n \Longrightarrow l \to r$ is a rule $R(\mathcal{P})$ then v_i is of the form $Pout(s_1, \ldots, s_k)$ and l is of the form $Pin(t_1, \ldots, t_n)$. So each v_i is absolutely irreducible wrt. $R(\mathcal{P})$. Hence $R(\mathcal{P})$ is absolutely deterministic.
c) This follows from Lemma 6.1. □

The following lemma relates computations in $\mathcal{P}$ to computations in $R(\mathcal{P})$. It is proved in [GW92].

Lemma 6.3 *Let $\mathcal{P}$ be an LR-well-moded logic program such that $R(\mathcal{P})$ is quasi-reductive and let $\leftarrow B$ be an LR-well-moded query. If there is a left-to-right SLD-refutation of $\leftarrow B$ with computed answer substitution θ and $\rho(B) \equiv Pin(t_1, \ldots, t_m) \to Pout(s_1, \ldots, s_k)$ then $Pin(t_1, \ldots, t_m) \to_R Pout(\theta(s_1), \ldots, \theta(s_k))$* □.

We now come to the main result of this section.

Definition 6.2 *An LR-well-moded logic program $\mathcal{P}$ is* uniquely terminating *if for every LR-well-moded query $\leftarrow B$ every left-to-right SLD-derivation is terminating and every left-to-right SLD-refutation computes the same answer substitution θ.*

Theorem 6.1 *Let $\mathcal{P}$ be an LR-well-moded logic program such that $R(\mathcal{P})$ is quasi-reductive and every critical pair in $CP(R(\mathcal{P}))$ is either unfeasible or context-joinable. Then $\mathcal{P}$ is uniquely terminating.*

Proof: By Theorem 4.5 of [GW92] every left-to-right SLD-derivation starting with an LR-well-moded query $\leftarrow B$ terminates. Since $R(\mathcal{P})$ is strongly deterministic it is confluent by Theorem 4.2. Now the statement follows from Lemma 6.3. □

Example 6.2

a) The following logic program is intended to compute from a given list l the last element z of l and the list l' resulting from l by eliminating z.

$\mathcal{P}$: $P(x.nil, x, nil) \leftarrow$
$P(x.y.l, z, x.l') \leftarrow P(y.l, z, l')$

Translation of $\mathcal{P}$ into a rewrite system gives

$$R(\mathcal{P}): \quad \begin{array}{rl} & Pin(x.nil) \rightarrow Pout(x, nil) \\ Pin(y.l) \rightarrow Pout(z, l') \Longrightarrow & Pin(x.y.l) \rightarrow Pout(z, x.l') \end{array}$$

We prove that $R(\mathcal{P})$ is quasi-reductive by using the backward substitution technique: We have to prove that $Pin(x.y.l) \succ Pout(Pin(y.l), x.Pin(y.l))$. This holds true for the RPO based on the precedence $Pin > Pout, Pin >$. We have $CP(R(\mathcal{P})) = \emptyset$, so $R(\mathcal{P})$ is confluent and $\mathcal{P}$ is uniquely terminating.

b) The following logic program $\mathcal{P}$ is intended to compute the greatest common divisor z of the natural numbers x, y.

$\mathcal{P}$: $GCD(x, 0, x) \leftarrow$
$GCD(0, y, y) \leftarrow$
$GCD(s(x), s(y), z) \leftarrow LESS(x, y, true), SUB(y, x, z_1), GCD(s(x), z_1, z)$
$GCD(s(x), s(y), z) \leftarrow LESS(x, y, false), SUB(x, y, z_1), GCD(z_1, s(y), z)$
$SUB(x, 0, x) \leftarrow$
$SUB(s(x), s(y), z) \leftarrow SUB(x, y, z)$
$LESS(x, 0, false) \leftarrow$
$LESS(0, s(y), true) \leftarrow$
$LESS(s(x), s(y), u) \leftarrow LESS(x, y, u)$

This gives $R(\mathcal{P})$:

$$
\begin{array}{lll}
(1) & & GCDin(x,0) \to GCDout(x) \\
(2) & & GCDin(0,y) \to GCDout(y) \\
(3) & LESSin(x,y) \to LESSout(true), & \\
 & SUBin(y,x) \to SUBout(z_1), & \\
 & GCDin(s(x),z_1) \to GCDout(z) \Longrightarrow & GCDin(s(x),s(y)) \to GCDout(z) \\
(4) & LESSin(x,y) \to LESSout(false), & \\
 & SUBin(x,y) \to SUBout(z_1), & \\
 & GCDin(z_1,s(y)) \to GCDout(z) \Longrightarrow & GCDin(s(x),s(y)) \to GCDout(z) \\
(5) & & SUBin(x,0) \to SUBout(x) \\
(6) & SUBin(x,y) \to SUBout(z) \Longrightarrow & SUBin(s(x),s(y)) \to SUBout(z) \\
(7) & & LESSin(x,0) \to LESSout(false) \\
(8) & & LESSin(0,s(y)) \to LESSout(true) \\
(9) & LESSin(x,y) \to LESSout(u) \Longrightarrow & LESSin(s(x),s(y)) \to LESSout(u)
\end{array}
$$

One can prove that $R(\mathcal{P})$ is quasi-reductive using backward substitution and an appropriate semantic path ordering [St93]. There is only one proper critical pair $C \Longrightarrow GCDout(z) = GCDout(z')$ generated by the third and fourth rule in $R(\mathcal{P})$. Here C contains $LESSin(x,y) \to LESSout(true)$ and $LESSin(x,y) \to LESSout(false)$, so this critical pair is unfeasible. Hence $R(\mathcal{P})$ is confluent and $\mathcal{P}$ is uniquely terminating.

References

[AM90] Avenhaus, J., Madlener, K.: Term rewriting and equational reasoning, in: R.B. Banerji, ed., Formal Techniques in Artifical Intelligence, North Holland, Amsterdam (1990), pp. 1 – 43.

[BG89] Bertling, H., Ganzinger, H.: Completion-time optimization of rewrite-time goal solving, 3rd RTA (1989), LNCS 355, pp. 45 – 58.

[De87] Dershowitz, N.: Termination, 1st RTA (1985), LNCS 202, pp. 180 – 224 and J. Symbolic Comp. 3 (1987), pp. 69 – 116.

[De91] Dershowitz, N.: Ordering-based strategies for Horn-clauses, 12th IJCAI (1991), pp. 118 – 124.

[DJ90] Dershowitz, N., Jouannaud, J.P.: Rewriting systems, in J. van Leeuwen, ed., Handbook of Theoretical Computer Science, Vol. B, Elsevier, Amsterdam (1990), pp. 241 – 320.

[DO90] Dershowitz, N., Okada, M.: A rationale for conditional equational programming, TCS 75 (1990), pp. 111 – 138.

[DOS88] Dershowitz, N., Okada, M., Sivakumar, G.: Confluence of conditional rewrite systems, 1st CTRS (1988), LNCS 308, pp. 31 – 44.

[Ga91] Ganzinger, H.: Order-sorted completion: the many-sorted way, TCS 89 (1991), pp. 3 – 32.

[**Ge92**] Geser, A.: On a monotonic semantic path ordering, Ulmer Informatik-Berichte No. 92 – 13, Universität Ulm (1992).

[**GW92**] Ganzinger, H., Waldmann, U.: Termination proofs of well-moded logic programs via conditional rewrite systems, 3rd CTRS (1992), Extended Abstract, pp. 216 – 222.

[**HA85**] Heck, N., Avenhaus, J.: On logic programs with data-driven computations, EUROCAL-85 (1985), LNCS 204, pp. 433 – 443.

[**Hu80**] Huet, G.: Confluent reductions: Abstract properties and applications to term rewriting systems, J. ACM 27 (1980), pp. 797 – 821.

[**Ka87**] Kaplan, S.: Simplifying conditional term rewriting systems: Unification, termination and confluence, JSC (1987), pp. 295 – 334.

[**LS92**] Loría-Sáenz, C.: Synthesis of narrowing programs, in: T. Clement, K.K. Lau, eds., Logic Program Synthesis and Transformation, Springer (1992).

[**St93**] Steinbach, J.: Private communication.