New Completeness Results for Lazy Conditional Narrowing

Mircea Marin
Johann Radon Institute for Computational and Applied Mathematics
Austrian Academy of Sciences
A-4040 Linz, Austria
mircea.marin@oeaw.ac.at

Aart Middeldorp
Institute of Computer Science
University of Innsbruck
A-6020 Innsbruck, Austria
aart.middeldorp@uibk.ac.at

ABSTRACT

We show the completeness of the lazy conditional narrowing calculus (LCNC) with leftmost selection for the class of deterministic conditional rewrite systems (CTRSs). Deterministic CTRSs permit extra variables in the right-hand sides and conditions of their rewrite rules. From the completeness proof we obtain several insights to make the calculus more deterministic. Furthermore, and similar to the refinements developed for the unconditional case, we succeeded in removing all nondeterminism due to the choice of the inference rule of LCNC by imposing further syntactic conditions on the participating CTRSs and restricting the set of solutions for which completeness needs to be established.

Categories and Subject Descriptors

D.1.6 [**Programming Techniques**]: Logic Programming; F.4.2 [**Mathematical Logic and Formal Languages**]: Grammars and Other Rewriting Systems

General Terms

Algorithms, Theory

Keywords

Narrowing, Conditional Rewriting, Evaluation Strategies

1. INTRODUCTION

Narrowing was originally invented as a general method for solving unification problems in equational theories that are presented by confluent term rewriting systems (TRSs for short). More recently, narrowing was proposed as the computational mechanism of functional-logic programming languages and several new completeness results concerning the completeness of various narrowing strategies and calculi have been obtained in the past few years. Here completeness means that for every solution to a given goal a solution that is at least as general is computed by the narrowing strategy. In the literature numerous calculi consisting of a small number of more elementary inference rules that simulate narrowing have been proposed (e.g. [7, 11, 16]).

Completeness issues for the lazy narrowing calculus LNC have been extensively studied in [11] and [10]. The main result of [11] is the completeness of LNC with leftmost selection for arbitrary confluent TRSs and normalized solutions. In [10] restrictions on the participating TRSs and solutions are presented which guarantee that all nondeterminism due to the choice of inference rule of LNC is removed. The resulting calculus LNC_d satisfies similar optimality properties as the needed narrowing strategy of Antoy *et al.* [4].

In this paper we consider the lazy conditional narrowing calculus LCNC of [12]. LCNC is the extension of LNC to conditional term rewrite systems (CTRSs for short). The extension is motivated by the observation that CTRSs are much more suitable than unconditional TRSs for describing interesting problems in a natural and concise way. However, the additional expressive power raises two problems: (1) completeness results are harder to obtain, and (2) the search space increases dramatically because the conditions of the applied rewrite rule are added to the current goal.

In [12] three completeness results for LCNC are presented: (a) LCNC with leftmost selection is complete with respect to normalized solutions for confluent CTRSs without extra variables, (b) LCNC is strongly complete whenever basic conditional narrowing is complete, and (c) LCNC is complete for terminating level-confluent conditional rewrite systems. Strong completeness means completeness with respect to any selection function and basic conditional narrowing is known to be complete for several classes of terminating CTRSs. So the only completeness result which does not assume some kind of termination assumption does not permit extra variables in the conditions and right-hand sides of the rewrite rules. In this paper we show the completeness of LCNC with leftmost selection (LCNC_ℓ) for the class of *deterministic* CTRSs. Determinism was introduced by Ganzinger [6] and has proved to be very useful for the study of the (unique) termination behavior of well-moded Horn clause programs (cf. [14]).

An important problem is to find refinements of LCNC_ℓ which reduce the search space while preserving completeness. From the completeness proof of LCNC_ℓ we obtain several insights to make the calculus more deterministic. Furthermore, and similar to the refinements developed in [10] for the unconditional case, we succeeded in removing all nondeterminism due to the choice of the inference rule by imposing further syntactic conditions on the participating

PPDP'04, August 24–26, Verona, Italy.

CTRSs and restricting the set of solutions for which completeness needs to be established.

The paper is structured as follows. In Section 2 we fix our notation and terminology and recall some relevant properties of conditional term rewriting and lazy conditional narrowing. In Section 3 we show that LCNC$_\ell$ is complete with respect to normalized solutions for deterministic CTRSs. In the next two sections we propose various refinements which reduce the search space of LCNC$_\ell$ without affecting completeness. Section 4 contains the refinements which hold in general whereas the refinements that rely on further conditions on the participating CTRSs and the solutions for which completeness is guaranteed are discussed in Section 5. Most of the proofs of the results in Sections 3, 4, and 5 can be found in the appendix. We provide some examples in Section 6 to confirm the effectiveness of our refinements. In Section 7 we argue why it makes sense to develop efficient narrowing calculi that operate on CTRSs rather than to rely on transformations into unconditional TRSs.

2. PRELIMINARIES

Familiarity with term rewriting ([5]) will be helpful. A recent survey of conditional term rewriting can be found in [15]. We consider a set $\mathcal{T}(\mathcal{F},\mathcal{V})$ of terms built from a set of function symbols with given arities $\mathcal{F}$ and a countably infinite set of variables $\mathcal{V}$. We write $\mathrm{root}(t)$ for the root symbol of a term t and $\mathcal{V}\mathrm{ar}(t)$ for the set of variables which occur in t. An *equation* is either an *unoriented* equation $s \approx t$, an *oriented* equation $s \triangleright t$, or the constant $\mathtt{true}$. We assume that $\approx, \triangleright, \mathtt{true} \notin \mathcal{F}$. An equation between identical terms is called *trivial*. A *goal* is a sequence of equations. The empty sequence is denoted by $\Box$. A *proper* goal does not contain any occurrences of $\mathtt{true}$. A CTRS $\mathcal{R}$ is a set of rewrite rules of the form $l \to r \Leftarrow c$ such that $l \notin \mathcal{V}$ and the conditional part c is a proper goal. We require that $\mathcal{V}\mathrm{ar}(r) \subseteq \mathcal{V}\mathrm{ar}(l, c)$. (So we deal with so-called 3-CTRSs in this paper.) We write $l \to r$ for $l \to r \Leftarrow \Box$. If the conditional part of every rewrite rule consists of unoriented (oriented) equations then $\mathcal{R}$ is called a *join* (*oriented*) CTRS. We define the rewrite relation $\to_\mathcal{R}$ associated with $\mathcal{R}$ on terms inductively as follows: $s \to_\mathcal{R} t$ if there exists a rewrite rule $l \to r \Leftarrow c$ in $\mathcal{R}$, a position p in s, and a substitution θ such that $s|_p = l\theta$, $t = s[r\theta]_p$, and $\mathcal{R} \vdash e$ for all equations e in $c\theta$. The latter is defined as follows: $\mathcal{R} \vdash s \triangleright t$ if $s \to^*_\mathcal{R} t$ and $\mathcal{R} \vdash s \approx t$ if there exists a term u such that $s \to^*_\mathcal{R} u$ and $t \to^*_\mathcal{R} u$. We extend $\to_\mathcal{R}$ to equations as follows: $s \approx t \to_\mathcal{R} e$ if one of the following three alternatives holds: (1) $e = \mathtt{true}$ and $s = t$, (2) $e = s' \approx t$ and $s \to_\mathcal{R} s'$, or (3) $e = s \approx t'$ and $t \to_\mathcal{R} t'$, and $s \triangleright t \to_\mathcal{R} e$ if either $e = \mathtt{true}$ and $s = t$ or $e = s' \approx t$ and $s \to_\mathcal{R} s'$. So unoriented equations are interpreted as joinability statements. Furthermore, for goals G we define $G \to_\mathcal{R} G'$ if $G = G_1, e, G_2$; $G' = G_1, e', G_2$, and $e \to_\mathcal{R} e'$. It is well-known that $\mathcal{R} \vdash G$ if and only if $G \to^*_\mathcal{R} \top$ where $\top$ denotes any sequence of $\mathtt{true}$s. The set $\mathcal{F}_\mathcal{D}$ of defined function symbols of $\mathcal{R}$ is defined as $\{\mathrm{root}(l) \mid l \to r \Leftarrow c \in \mathcal{R}\}$. Function symbols in $\mathcal{F}_\mathcal{C} = \mathcal{F} \setminus \mathcal{F}_\mathcal{D}$ are called *constructors*. A substitution θ is a *solution* of a goal G if $G\theta \to^*_\mathcal{R} \top$. Since confluence is insufficient [19] to guarantee that joinability coincides with the equality relation induced by the underlying conditional equational theory, we decided to drop confluence and use the above notion of solution. We say that θ is *X-normalized* for a subset X of $\mathcal{V}$ if every variable in X is mapped to a normal form with respect to $\mathcal{R}$.

The lazy conditional narrowing calculus LCNC$_\ell$ consists of the following inference rules:

[o] *outermost narrowing*

$$\frac{f(s_1,\ldots,s_n) \doteq t, G}{s_1 \triangleright l_1, \ldots, s_n \triangleright l_n, c, r \doteq t, G} \quad \doteq \in \{\approx, \triangleright, \bowtie\}$$

if $f(l_1,\ldots,l_n) \to r \Leftarrow c$ is a fresh variant of a rewrite rule in $\mathcal{R}$,

[i] *imitation*

$$\frac{f(s_1,\ldots,s_n) \doteq x, G}{(s_1 \doteq x_1, \ldots, s_n \doteq x_n, G)\theta} \quad \doteq \in \{\approx, \triangleright, \bowtie, \triangleleft\}$$

if $\theta = \{x \mapsto f(x_1,\ldots,x_n)\}$ with $x_1,\ldots,x_n$ fresh variables,

[d] *decomposition*

$$\frac{f(s_1,\ldots,s_n) \doteq f(t_1,\ldots,t_n), G}{s_1 \doteq t_1, \ldots, s_n \doteq t_n, G} \quad \doteq \in \{\approx, \triangleright\}$$

[v] *variable elimination*

$$\frac{s \doteq x, G}{G\theta} \qquad \frac{x \doteq s, G}{G\theta}\ s \notin \mathcal{V}$$

if $x \notin \mathcal{V}\mathrm{ar}(s)$, $\doteq \in \{\approx, \triangleright\}$, and $\theta = \{x \mapsto s\}$,

[t] *removal of trivial equations*

$$\frac{s \doteq s, G}{G} \quad \doteq \in \{\approx, \triangleright\}$$

In the above rules we use $s \bowtie t$ to denote the equation $t \approx s$ and $s \triangleleft t$ to denote the equation $t \triangleright s$. Compared to the definition of LCNC in [12], in LCNC$_\ell$ leftmost selection is built-in. Another difference is that the *parameter-passing* equations $s_1 \triangleright l_1, \ldots, s_n \triangleright l_n$ created by the outermost narrowing rule [o] are regarded as oriented equations. This is in line with the earlier completeness proofs for LNC and LCNC [10, 11, 12]. Parameter-passing equations must eventually be solved in order to obtain a refutation, but it is not required that they are solved right away. That is the reason why calculi in the LNC family are called *lazy*. Furthermore, the conditions c are placed before $r \doteq t$ whereas in [12] they are placed after $r \doteq t$. The relative locations of c and $r \doteq t$ are irrelevant for the completeness results reported in that paper, but our new completeness proofs do not go through if we stick to the order in [12]. If the variable elimination rule [v] is applied to an equation between two different variables, we can only eliminate the variable on the right-hand side. Finally, the removal of trivial equations rule [t] is usually restricted to equations between identical variables. The variation adopted here avoids some infinite derivations as well as refutations which eventually compute substitutions which are subsumed by the empty substitution of the [t]-step and therefore redundant.

If G_1 and G_2 are the upper and lower goal in the inference rule $[\alpha]$, we write $G_1 \Rightarrow_{[\alpha]} G_2$. The applied rewrite rule or substitution may be supplied as subscript, that is, we write things like $G_1 \Rightarrow_{[\mathtt{o}], l \to r \Leftarrow c} G_2$ and $G_1 \Rightarrow_{[\mathtt{i}],\theta} G_2$. A finite LCNC$_\ell$-derivation $G_1 \Rightarrow_{\theta_1} \cdots \Rightarrow_{\theta_{n-1}} G_n$ may be abbreviated to $G_1 \Rightarrow^*_\theta G_2$ where θ is the composition $\theta_1 \cdots \theta_{n-1}$ of the substitutions $\theta_1,\ldots,\theta_{n-1}$ computed along its steps. An LCNC$_\ell$-refutation is an LCNC$_\ell$-derivation ending in the empty goal $\Box$.

3. SOUNDNESS AND COMPLETENESS

Soundness of LCNC$_\ell$ is not difficult to prove.

THEOREM 1. *Let $\mathcal{R}$ be an arbitrary CTRS and G a proper goal. If $G \Rightarrow^*_\theta \square$ then θ is a solution of G.* $\square$

It is known that the unconditional variant of LCNC$_\ell$ is complete for (confluent) TRSs and normalized solutions θ ([11, Corollary 40]). The following well-known example shows that extra variables in the rewrite rules may result in incompleteness.

Example 1. Consider the (confluent) CTRS consisting of the rules $\mathsf{f}(x) \to \mathsf{a} \Leftarrow y \approx \mathsf{g}(y)$ and $\mathsf{b} \to \mathsf{g}(\mathsf{b})$. The substitution $\theta = \{x \mapsto \mathsf{a}\}$ is a normalized solution of the goal $G = \mathsf{f}(x) \approx \mathsf{a}$. Any LCNC$_\ell$-derivation of G must start with the [o]-step

$$G \Rightarrow_{[\mathsf{o}]} x \rhd x_1, y_1 \approx \mathsf{g}(y_1), \mathsf{a} \approx \mathsf{a}.$$

The newly generated goal contains the equation $y_1 \approx \mathsf{g}(y_1)$ whose descendants are always of the form $z \approx \mathsf{g}(z)$ with $z \in \mathcal{V}$. Hence LCNC$_\ell$ cannot solve the goal G.

Definition 1. Let X be a finite set of variables. A proper goal $G = e_1, \ldots, e_n$ is called *X-deterministic* if

$$\mathcal{V}\mathrm{ar}(s_i) \subseteq X \cup \bigcup_{j=1}^{i-1} \mathcal{V}\mathrm{ar}(e_j)$$

when $e_i = s_i \rhd t_i$, and

$$\mathcal{V}\mathrm{ar}(e_i) \subseteq X \cup \bigcup_{j=1}^{i-1} \mathcal{V}\mathrm{ar}(e_j)$$

when $e_i = s_i \approx t_i$, for all $1 \leqslant i \leqslant n$. A CTRS $\mathcal{R}$ is called *deterministic* if the conditional part c of every rewrite rule $l \to r \Leftarrow c$ in $\mathcal{R}$ is $\mathcal{V}\mathrm{ar}(l)$-deterministic.

The CTRS of Example 1 is not deterministic since the variable y occurring in the condition of the rewrite rule $\mathsf{f}(x) \to \mathsf{a} \Leftarrow y \approx \mathsf{g}(y)$ does not occur in its left-hand side $\mathsf{f}(x)$.

It should be remarked that the above definition of determinism (which was introduced by Ganzinger [6] to study the termination behavior of well-moded logic programs) has nothing to do with confluence or deterministic evaluation of terms. CTRSs are called deterministic because there is no guessing of suitable instantiations of extra variables when executing a conditional rewrite rule. In particular, every CTRS without extra variables is deterministic.

The following oriented CTRS $\mathcal{R}$ is a natural encoding of the efficient computation of Fibonacci numbers:

$$\begin{array}{rcl@{\qquad}rcl}
0 + y & \to & y & \mathsf{fst}(\langle x, y\rangle) & \to & x \\
\mathsf{s}(x) + y & \to & \mathsf{s}(x + y) & \mathsf{snd}(\langle x, y\rangle) & \to & y \\
\mathsf{fib}(0) & \to & \langle 0, \mathsf{s}(0)\rangle & & & \\
\mathsf{fib}(\mathsf{s}(x)) & \to & \langle z, y + z\rangle \Leftarrow \mathsf{fib}(x) \rhd \langle y, z\rangle & & &
\end{array}$$

Note that $\mathcal{R}$ is deterministic. It is also terminating. However, when we change the oriented condition $\mathsf{fib}(x) \rhd \langle y, z\rangle$ into an unoriented condition $\mathsf{fib}(x) \approx \langle y, z\rangle$ then we lose termination: Because $\mathsf{fib}(\mathsf{s}(0)) \to^+ \langle \mathsf{s}(0), \mathsf{s}(0)\rangle$ we have $\mathsf{fib}(0) \downarrow \langle 0, \mathsf{fst}(\mathsf{fib}(\mathsf{s}(0)))\rangle$ and thus

$$\mathsf{fib}(\mathsf{s}(0)) \to \langle \mathsf{fst}(\mathsf{fib}(\mathsf{s}(0))), 0 + \mathsf{fst}(\mathsf{fib}(\mathsf{s}(0)))\rangle,$$

which gives rise to an infinite rewrite sequence. As a consequence, the completeness results for LCNC presented in [12], which deal only with join CTRSs, do not apply to this CTRS. The completeness result presented below does not have this problem.

THEOREM 2. *Let $\mathcal{R}$ be a deterministic CTRS and G an X-deterministic goal. If θ is an X-normalized solution of G then there exists an* LCNC*-refutation $G \Rightarrow^*_{\theta'} \square$ such that $\theta' \leqslant \theta$ $[\mathcal{V}\mathrm{ar}(G)]$.* $\square$

The proof is based on a transformation of rewrite proofs of $G\theta$ and is given in Appendix A. The crucial insight is that determinism permits one to identify suitable sets of variables X_1, X_2, ... such that each intermediate goal G_i in the refutation $G \Rightarrow G_1 \Rightarrow G_2 \Rightarrow \cdots \Rightarrow \square$ that is being constructed is X_i-deterministic.

Note that any proper goal G has a minimal set of variables $X \subseteq \mathcal{V}\mathrm{ar}(G)$ for which it is X-deterministic. This set will be denoted by X_G in the following. An X_G-normalized solution of G will simply be called *normalized*.

4. REFINEMENTS I

In this section we present several improvements of LCNC$_\ell$ which do neither affect the completeness result stated in Theorem 2 nor require any further restrictions on the participating CTRSs and the solutions for which completeness needs to established.

The correctness of the refinement expressed in the following lemma is an immediate consequence of the completeness proof. Here "eagerly" means that we do not have to consider other applicable inference rules in order to guarantee completeness.

LEMMA 1. *If $\mathcal{R}$ is a deterministic CTRS then the calculus obtained from* LCNC$_\ell$ *by applying inference rule* [t] *eagerly, and applying inference rule* [v] *eagerly to equations $x \rhd t$ with $x \neq t$ is complete with respect to normalized solutions.* $\square$

Completeness with respect to normalized solutions does not imply that all solutions computed by LCNC$_\ell$ are normalized. This is illustrated in the following example.

Example 2. Consider the TRS consisting of the two rules $\mathsf{f}(\mathsf{a}) \to \mathsf{b}$ and $\mathsf{g}(\mathsf{b}) \to \mathsf{c}$ together with the goal

$$G = x \approx \mathsf{f}(y), \mathsf{g}(x) \approx \mathsf{c}, \mathsf{f}(y) \approx \mathsf{b}.$$

The following LCNC$_\ell$-refutation computes the non-normalized solution $\theta = \{x \mapsto \mathsf{f}(\mathsf{a}), y \mapsto \mathsf{a}\}$:

$$\begin{aligned}
G &\Rightarrow_{[\mathsf{v}],\, \{x \mapsto \mathsf{f}(y)\}} \mathsf{g}(\mathsf{f}(y)) \approx \mathsf{c}, \mathsf{f}(y) \approx \mathsf{b} \\
&\Rightarrow_{[\mathsf{o}],\, \mathsf{g}(\mathsf{b}) \to \mathsf{c}} \mathsf{f}(y) \rhd \mathsf{b}, \mathsf{c} \approx \mathsf{c}, \mathsf{f}(y) \approx \mathsf{b} \\
&\Rightarrow_{[\mathsf{o}],\, \mathsf{f}(\mathsf{a}) \to \mathsf{b}} y \rhd \mathsf{a}, \mathsf{b} \rhd \mathsf{b}, \mathsf{c} \approx \mathsf{c}, \mathsf{f}(y) \approx \mathsf{b} \\
&\Rightarrow_{[\mathsf{v}],\, \{y \mapsto \mathsf{a}\}} \mathsf{b} \rhd \mathsf{b}, \mathsf{c} \approx \mathsf{c}, \mathsf{f}(\mathsf{a}) \approx \mathsf{b} \\
&\Rightarrow^2_{[\mathsf{t}]} \mathsf{f}(\mathsf{a}) \approx \mathsf{b} \\
&\Rightarrow_{[\mathsf{o}],\, \mathsf{f}(\mathsf{a}) \to \mathsf{b}} \mathsf{a} \rhd \mathsf{a}, \mathsf{b} \approx \mathsf{b} \\
&\Rightarrow^2_{[\mathsf{t}]} \square.
\end{aligned}$$

Substitution θ can be normalized to $\theta' = \{x \mapsto \mathsf{b}, y \mapsto \mathsf{a}\}$. Since θ' is also a computed answer substitution of G:

$$G \Rightarrow_{[\mathsf{o}],\, \mathsf{f}(\mathsf{a}) \to \mathsf{b}} y \rhd \mathsf{a}, x \approx \mathsf{b}, \mathsf{g}(x) \approx \mathsf{c}, \mathsf{f}(y) \approx \mathsf{b}$$

$$\Rightarrow_{[\mathtt{v}],\{y \mapsto \mathsf{a}\}} x \approx \mathsf{b}, \mathsf{g}(x) \approx \mathsf{c}, \mathsf{f}(\mathsf{a}) \approx \mathsf{b}$$
$$\Rightarrow_{[\mathtt{v}],\{x \mapsto \mathsf{b}\}} \mathsf{g}(\mathsf{b}) \approx \mathsf{c}, \mathsf{f}(\mathsf{a}) \approx \mathsf{b}$$
$$\Rightarrow_{[\mathtt{o}],\mathsf{g}(\mathsf{b}) \to \mathsf{c}} \mathsf{b} \rhd \mathsf{b}, \mathsf{c} \approx \mathsf{c}, \mathsf{f}(\mathsf{a}) \approx \mathsf{b}$$
$$\Rightarrow^2_{[\mathtt{t}]} \mathsf{f}(\mathsf{a}) \approx \mathsf{b}$$
$$\Rightarrow_{[\mathtt{o}],\mathsf{f}(\mathsf{a}) \to \mathsf{b}} \mathsf{a} \rhd \mathsf{a}, \mathsf{b} \approx \mathsf{b}$$
$$\Rightarrow^2_{[\mathtt{t}]} \Box,$$

there is no need to compute the non-normalized solution θ.

We propose a simple marking strategy to avoid computing (many) non-normalized solutions. Before presenting the formal details, we illustrate how this strategy avoids the first refutation in the previous example. We want to compute normalized bindings $x\theta$ and $y\theta$ for the variables x and y. Therefore, we mark the occurrences of x and y in G with the marker †. Whenever a marked variable is instantiated we mark the root symbol of the instantiation. So the first step becomes

$$x^\dagger \approx \mathsf{f}(y^\dagger), \mathsf{g}(x^\dagger) \approx \mathsf{c}, \mathsf{f}(y^\dagger) \approx \mathsf{b}$$
$$\Rightarrow_{[\mathtt{v}],\{x^\dagger \mapsto \mathsf{f}^\dagger(y^\dagger)\}} \mathsf{g}(\mathsf{f}^\dagger(y^\dagger)) \approx \mathsf{c}, \mathsf{f}(y^\dagger) \approx \mathsf{b}.$$

Note that only the first occurrence of f in the new goal is marked. The next step is the same as before since the marker is not involved:

$$\mathsf{g}(\mathsf{f}^\dagger(y^\dagger)) \approx \mathsf{c}, \mathsf{f}(y^\dagger) \approx \mathsf{b}$$
$$\Rightarrow_{[\mathtt{o}],\mathsf{g}(\mathsf{b}) \to \mathsf{c}} \mathsf{f}^\dagger(y^\dagger) \rhd \mathsf{b}, \mathsf{c} \approx \mathsf{c}, \mathsf{f}(y^\dagger) \approx \mathsf{b}.$$

At this point we do *not* perform the [o] step since the marker on f signals that any instantiation $\mathsf{f}(y\theta)$ of $\mathsf{f}(y)$ should be normalized, but then $\mathsf{f}(y\theta)$ cannot be rewritten to b. Since there are no other applicable inference rules, the derivation ends in failure after just two steps.

By modifying the inference rules [i], [d], [v], and [t] of LCNC$_\ell$ as described below, the introduction and propagation of the markers is achieved. We consider terms from $\mathcal{T}(\mathcal{F} \cup \mathcal{F}^\dagger, \mathcal{V} \cup \mathcal{V}^\dagger)$ where $\mathcal{F}^\dagger$ ($\mathcal{V}^\dagger$) is the set of marked function symbols (variables). We write $t^\dagger$ for the term obtained from t by marking its root symbol, provided the root symbol of t is unmarked. Otherwise $t^\dagger = t$. Furthermore, the result of erasing all markers in a term t (goal G, substitution θ) is denoted by $\mathrm{u}(t)$ ($\mathrm{u}(G)$, $\mathrm{u}(\theta)$).

[i] *imitation*

$$\frac{f(s_1,\ldots,s_n) \rhd x, G}{(s_1 \rhd x_1,\ldots,s_n \rhd x_n, G)\theta}$$

$$\frac{f(s_1,\ldots,s_n) \doteq x^\dagger, G}{s_1\theta' \doteq x_1^\dagger,\ldots,s_n\theta' \doteq x_n^\dagger, G\theta'}$$

with $\doteq \in \{\approx, \rhd, \lessapprox\}$, $\theta = \{x \mapsto f(x_1,\ldots,x_n), x^\dagger \mapsto f^\dagger(x_1,\ldots,x_n)\}$, $\theta' = \{x, x^\dagger \mapsto f^\dagger(x_1,\ldots,x_n)\}$, and $x_1,\ldots,x_n$ fresh variables,

[d] *decomposition*

$$\frac{f(s_1,\ldots,s_n) \doteq f(t_1,\ldots,t_n), G}{s_1 \doteq t_1,\ldots,s_n \doteq t_n, G}$$

$$\frac{f^\dagger(s_1,\ldots,s_n) \doteq f(t_1,\ldots,t_n), G}{s_1^\dagger \doteq t_1,\ldots,s_n^\dagger \doteq t_n, G}$$

$$\frac{f(s_1,\ldots,s_n) \doteq f^\dagger(t_1,\ldots,t_n), G}{s_1 \doteq t_1^\dagger,\ldots,s_n \doteq t_n^\dagger, G}$$

$$\frac{f^\dagger(s_1,\ldots,s_n) \doteq f^\dagger(t_1,\ldots,t_n), G}{s_1^\dagger \doteq t_1^\dagger,\ldots,s_n^\dagger \doteq t_n^\dagger, G}$$

with $\doteq \in \{\approx, \rhd\}$,

[v] *variable elimination*

$$\frac{x^\dagger \doteq s, G}{G\theta'}\ s \notin \mathcal{V} \cup \mathcal{V}^\dagger \qquad \frac{s \doteq x^\dagger, G}{G\theta'} \qquad \frac{s \rhd x, G}{G\theta}$$

with $x \notin \mathcal{V}\mathrm{ar}(\mathrm{u}(s))$, $\doteq \in \{\approx, \rhd\}$, $\theta = \{x \mapsto s, x^\dagger \mapsto s^\dagger\}$, and $\theta' = \{x, x^\dagger \mapsto s^\dagger\} \cup \{y \mapsto y^\dagger \mid y \in \mathcal{V}\mathrm{ar}(\mathrm{u}(s))\}$,

[t] *removal of trivial equations*

$$\frac{s \doteq t, G}{G} \qquad \begin{array}{l}\mathrm{u}(s) = \mathrm{u}(t)\\ \doteq \in \{\approx, \rhd\}\end{array}$$

Note that the imitation rule [i] is not applicable if the root symbol of the non-variable side of the equation is marked. Imitation is also not applicable to oriented equations whose left-hand side is a (marked) variable. Further note that [i] and [v] do not apply to equations of the form $s \approx x$, $x \approx s$, or $x \rhd s$. These optimizations are a consequence of the proof of Theorem 3.

Let LCNC$^\dagger_\ell$ be the calculus obtained from LCNC$_\ell$ by adjusting the inference rules [i], [d], [v], [t] as described above and adopting the selection strategy illustrated in Tables 1 and 2. Here the symbol ";" separates (groups of) rules in decreasing order of priority. The subscript (1 or 2) in [o] indicates to which side (left or right) of the selected equation the rule is applied. Note that the result of Lemma 1 is incorporated in the selection strategy of LCNC$^\dagger_\ell$. For a proper goal G we denote by $G^\dagger$ the result of marking in G all variables belonging to X_G. The completeness proof of LCNC$^\dagger_\ell$ is given in Appendix B.

THEOREM 3. *Let $\mathcal{R}$ be a deterministic CTRS and θ a normalized solution of a proper goal G. There exists an* LCNC$^\dagger_\ell$*-refutation $G^\dagger \Rightarrow^*_{\theta'} \Box$ such that $\mathrm{u}(\theta') \leqslant \theta\ [\mathcal{V}\mathrm{ar}(G)]$.* □

Our marking strategy avoids the computation of many non-normalized solutions, but it does not always succeed. This can be seen from the goal $G = x \approx \mathsf{f}(\mathsf{a})$ and the TRS $\mathcal{R} = \{\mathsf{f}(x) \to x\}$. We have the refutation

$$G^\dagger = x^\dagger \approx \mathsf{f}(\mathsf{a}) \Rightarrow_{[\mathtt{v}],\{x, x^\dagger \mapsto \mathsf{f}^\dagger(\mathsf{a})\}} \Box$$

which computes the non-normalized solution $\{x \mapsto \mathsf{f}(\mathsf{a})\}$.

5. REFINEMENTS II

Tables 1 and 2 reveal that the don't know nondeterminism of LCNC$^\dagger_\ell$ is still high. Further refinements are needed if we want to make LCNC$^\dagger_\ell$ an appealing computational model. For the unconditional version LNC of LCNC$_\ell$, such refinements have been successfully identified [10] by confining the computation to more restricted classes of rewrite systems and to equations with strict semantics. In the sequel we investigate whether these refinements can be generalized to LCNC$_\ell$.

Table 1: LCNC$^{\dagger}_{\ell}$: Selection of inference rule for unoriented equation $s \approx t$.

root(s) \ root(t)	$\mathcal{F}^{\dagger}$	$\mathcal{F}_{\mathcal{C}}$	$\mathcal{F}_{\mathcal{D}}$	$\mathcal{V}^{\dagger}$	$\mathcal{V}$
$\mathcal{F}^{\dagger}$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}],[\mathtt{o}]_2$	$[\mathtt{v}]$	$\times$
$\mathcal{F}_{\mathcal{C}}$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{o}]_2$	$[\mathtt{i}],[\mathtt{v}]$	$\times$
$\mathcal{F}_{\mathcal{D}}$	$[\mathtt{t}];[\mathtt{d}],[\mathtt{o}]_1$	$[\mathtt{o}]_1$	$[\mathtt{t}];[\mathtt{d}],[\mathtt{o}]_1,[\mathtt{o}]_2$	$[\mathtt{o}]_1,[\mathtt{i}],[\mathtt{v}]$	$\times$
$\mathcal{V}^{\dagger}$	$[\mathtt{v}]$	$[\mathtt{i}],[\mathtt{v}]$	$[\mathtt{o}]_2,[\mathtt{i}],[\mathtt{v}]$	$[\mathtt{t}];[\mathtt{v}]$	$\times$
$\mathcal{V}$	$\times$	$\times$	$\times$	$\times$	$\times$

Table 2: LCNC$^{\dagger}_{\ell}$: Selection of inference rule for oriented equation $s \rhd t$.

root(s) \ root(t)	$\mathcal{F}^{\dagger}$	$\mathcal{F}_{\mathcal{C}}$	$\mathcal{F}_{\mathcal{D}}$	$\mathcal{V} \cup \mathcal{V}^{\dagger}$
$\mathcal{F}^{\dagger}$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{v}]$
$\mathcal{F}_{\mathcal{C}}$	$[\mathtt{t}];[\mathtt{d}]$	$[\mathtt{t}];[\mathtt{d}]$	$\times$	$[\mathtt{i}],[\mathtt{v}]$
$\mathcal{F}_{\mathcal{D}}$	$[\mathtt{t}];[\mathtt{d}],[\mathtt{o}]_1$	$[\mathtt{o}]_1$	$[\mathtt{t}];[\mathtt{d}],[\mathtt{o}]_1$	$[\mathtt{o}]_1,[\mathtt{i}],[\mathtt{v}]$
$\mathcal{V}^{\dagger}$	$[\mathtt{v}]$	$[\mathtt{v}]$	$[\mathtt{v}]$	$[\mathtt{t}];[\mathtt{v}]$
$\mathcal{V}$	$\times$	$\times$	$\times$	$\times$

5.1 Eager Variable Elimination

Eager variable elimination is the problem of identifying sufficient criteria which guarantee that the eager application of inference rule [v] preserves completeness. For LNC with leftmost selection it is known that if the underlying TRS satisfies the *standardization* theorem then eager variable elimination for so-called *descendants* of parameter-passing equations preserves completeness [11]. Descendants for LCNC$_{\ell}$ are defined as follows. The selected equation $f(s_1,\ldots,s_n) \doteq t$ in rule [o] has $r \doteq t$ as only one-step descendant. In rule [i] all equations $s_i\theta \approx x_i$ are one-step descendants of the selected equation $f(s_1,\ldots,s_n) \doteq x$. The selected equation $f(s_1,\ldots,s_n) \doteq f(t_1,\ldots,t_n)$ in rule [d] has all equations $s_i \doteq t_i$ as one-step descendants. Finally, the selected equations in rules [v] and [t] have no one-step descendants. One-step descendants of non-selected equations are defined as expected. Descendants are obtained from one-step descendants by reflexivity and transitivity. Observe that every equation in an LCNC$_{\ell}$-derivation descends from either an equation in the initial goal, a parameter-passing equation, or an equation in the conditional part of the rewrite rule used in the [o] rule.

Suzuki [17] showed that standardization holds for left-linear *join* CTRSs. However, left-linearity is insufficient for *oriented* CTRSs as shown in Example 4 in Appendix C. Standardization is recovered if we impose a natural freshness condition on the variables in the right-hand sides of oriented equations. A CTRS is called *fresh* if every oriented equation $s \rhd t$ in the conditional part c of every rewrite rule $l \to r \Leftarrow c$ satisfies $\mathcal{V}\mathrm{ar}(l) \cap \mathcal{V}\mathrm{ar}(t) = \varnothing$.

To distinguish parameter-passing descendants from other oriented equations we denote them by $s \blacktriangleright t$ instead of $s \rhd t$. Let LCNC$^{\mathrm{eve}}_{\ell}$ be the calculus obtained from LCNC$^{\dagger}_{\ell}$ by incorporating the introduction and propagation of parameter-passing equations (in the obvious way) and imposing the selection strategy depicted in Table 3 for parameter-passing descendants. The $\mathcal{F}^{\dagger} \cup \mathcal{V}^{\dagger}$ column is explained by the (easy to prove) observation that, due to left-linearity, no right-hand side of a parameter-passing equations contains any marks.

THEOREM 4. *Let $\mathcal{R}$ be a left-linear fresh deterministic CTRS and θ a normalized solution of a proper goal G. There exists an* LCNC$^{\mathrm{eve}}_{\ell}$*-refutation* $G^{\dagger} \Rightarrow^{*}_{\theta'} \square$ *such that* $\mathrm{u}(\theta') \leqslant \theta\ [\mathcal{V}\mathrm{ar}(G)]$. $\square$

This result generalizes the eager variable elimination strategy for LNC. Our proof, which is sketched in Appendix C, is much simpler than the one in [10, 11]. The following example shows the necessity of the freshness condition.

Example 3. Consider the left-linear deterministic CTRS $\mathcal{R}$ consisting of the rules (1) $\mathsf{f}(x) \to x$ and (2) $\mathsf{g}(x,y) \to x \Leftarrow x \rhd y$ together with the goal $G = \mathsf{g}(x,\mathsf{f}(y)) \approx \mathsf{a}$. Note that $\mathcal{R}$ is not fresh. The substitution $\theta = \{x, y \mapsto \mathsf{a}\}$ is a normalized solution of G. The (only maximal) LCNC$^{\mathrm{eve}}_{\ell}$-derivation

$$\begin{aligned} G^{\dagger} &\Rightarrow_{[\mathtt{o}],(2)} x^{\dagger} \blacktriangleright x_1, \mathsf{f}(y^{\dagger}) \blacktriangleright y_1, x_1 \rhd y_1, x_1 \approx \mathsf{a} \\ &\Rightarrow_{[\mathtt{v}],\{x_1, x_1^{\dagger} \mapsto x^{\dagger}\}} \mathsf{f}(y^{\dagger}) \blacktriangleright y_1, x^{\dagger} \rhd y_1, x^{\dagger} \approx \mathsf{a} \\ &\Rightarrow_{[\mathtt{v}],\{y_1 \mapsto \mathsf{f}(y^{\dagger}),\, y_1^{\dagger} \mapsto \mathsf{f}^{\dagger}(y^{\dagger})\}} x^{\dagger} \rhd \mathsf{f}(y^{\dagger}), x^{\dagger} \approx \mathsf{a} \\ &\Rightarrow_{[\mathtt{v}],\{x, x^{\dagger} \mapsto \mathsf{f}^{\dagger}(y^{\dagger})\}} \mathsf{f}^{\dagger}(y^{\dagger}) \approx \mathsf{a} \end{aligned}$$

terminates in the nonempty goal $\mathsf{f}^{\dagger}(y^{\dagger}) \approx \mathsf{a}$. Hence LCNC$^{\mathrm{eve}}_{\ell}$ is incomplete for arbitrary left-linear deterministic CTRSs.

Just as in the unconditional case [10], the remaining nondeterminism in the selection of inference rules for descendants of parameter-passing equations (viz. $[\mathtt{d}],[\mathtt{o}]_1$ in the $\mathcal{F}_{\mathcal{D}}/\mathcal{F}_{\mathcal{D}}$ entry of Table 3) is eliminated by adopting the *constructor discipline*. A CTRS is called a conditional *constructor system* (CCS for short) if the arguments $l_1,\ldots,l_n$ of the

Table 3: $\text{LCNC}_\ell^{\text{eve}}$: Selection of inference rule for parameter-passing descendant $s \blacktriangleright t$.

$\text{root}(s)$ \ $\text{root}(t)$	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$	$\mathcal{V}$	$\mathcal{F}^\dagger \cup \mathcal{V}^\dagger$
$\mathcal{F}^\dagger$	[t]; [d]	[t]; [d]	[v]	×
$\mathcal{F}_\mathcal{C}$	[t]; [d]	×	[v]	×
$\mathcal{F}_\mathcal{D}$	$[\text{o}]_1$	[t]; [d], $[\text{o}]_1$	[v]	×
$\mathcal{V}^\dagger$	[v]	[v]	[v]	×
$\mathcal{V}$	×	×	×	×

left-hand side $f(l_1, \ldots, l_n)$ of every rewrite rule do not contain defined symbols. The proof of the following lemma is similar to the one for unconditional systems [10, Lemma 3.1].

LEMMA 2. *Let $\mathcal{R}$ be a left-linear CTRS. If $G \Rightarrow^* G_1, s \blacktriangleright t, G_2$ is an $\text{LCNC}_\ell^{\text{eve}}$-derivation then $\mathcal{V}\text{ar}(G_1, s) \cap \mathcal{V}\text{ar}(t) = \varnothing$. Moreover, if $\mathcal{R}$ is a CCS then t does not contain defined symbols.* □

In particular, the occur-check in rule [v] is unnecessary when dealing with descendants of parameter-passing equations.

5.2 Strict Semantics

Next we consider the nondeterminism in the selection of inference rules for descendants of initial equations and equations originating from the conditional parts of the applied rewrite rules in applications of the rule [o]. Just as in the unconditional case [10], it turns out that *strictness* is the key to eliminating this nondeterminism. So we no longer require that all normalized solutions are subsumed by computed answer substitutions, instead we only demand completeness with respect to normalized *strict* solutions.

In the literature [4, 8, 13] a substitution θ is called a strict solution of an equation $s \approx t$ if $s\theta$ and $t\theta$ rewrite to the same ground term without defined symbols. We do not require groundness. We say that θ is a strict solution of an oriented equation $s \rhd t$ if $s\theta$ rewrites to $t\theta$ and $t\theta$ is a term without defined symbols. These notions are easily incorporated into the definition of conditional rewriting. So when a conditional rewrite rule $l \to r \Leftarrow c$ is applied with substitution θ, we require that θ is a strict solution of the equations in c. Hence, as far as semantics is concerned, we do not distinguish equations in the initial goal from equations in the conditional parts of the rewrite rules.

The inference rules of $\text{LCNC}_\ell^{\text{s}}$, the strict version of LCNC_ℓ, are given below. Note that we take LCNC_ℓ rather than $\text{LCNC}_\ell^\dagger$ as the starting point, but we distinguish parameter-passing descendants from descendants of other oriented equations by using the symbol $\blacktriangleright$ to denote the former.

[o] *outermost narrowing*

$$\frac{f(s_1, \ldots, s_n) \doteq t, G}{s_1 \blacktriangleright l_1, \ldots, s_n \blacktriangleright l_n, c, r \doteq t, G} \quad \doteq \in \{\approx, \eqslantless, \rhd, \blacktriangleright\}$$

if $f(l_1, \ldots, l_n) \to r \Leftarrow c$ is a fresh variant of a rewrite rule in $\mathcal{R}$,

[i] *imitation*

$$\frac{g(s_1, \ldots, s_n) \doteq x, G}{(s_1 \doteq x_1, \ldots, s_n \doteq x_n, G)\theta} \quad \begin{array}{l} \doteq \in \{\approx, \eqslantless, \rhd\} \\ g \in \mathcal{F}_\mathcal{C} \end{array}$$

$$\frac{f(t_1, \ldots, t_n) \blacktriangleright x, G}{(t_1 \blacktriangleright x_1, \ldots, t_n \blacktriangleright x_n, G)\theta'}$$

if $g(s_1, \ldots, s_n) \notin \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$, $\theta = \{x \mapsto g(x_1, \ldots, x_n)\}$, and $\theta' = \{x \mapsto f(x_1, \ldots, x_n)\}$ with $x_1, \ldots, x_n$ fresh variables,

[d] *decomposition*

$$\frac{g(s_1, \ldots, s_n) \doteq g(t_1, \ldots, t_n), G}{s_1 \doteq t_1, \ldots, s_n \doteq t_n, G} \quad \begin{array}{l} \doteq \in \{\approx, \rhd\} \\ g \in \mathcal{F}_\mathcal{C} \end{array}$$

$$\frac{f(s_1, \ldots, s_n) \blacktriangleright f(t_1, \ldots, t_n), G}{s_1 \blacktriangleright t_1, \ldots, s_n \blacktriangleright t_n, G}$$

[v] *variable elimination*

$$\frac{x \blacktriangleright s, G}{G\theta} \quad s \notin \mathcal{V} \qquad \frac{s \doteq x, G}{G\theta} \quad s \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$$

$$\frac{s \blacktriangleright x, G}{G\theta} \qquad \frac{x \doteq s, G}{G\theta} \quad s \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V}) \setminus \mathcal{V}$$

if $x \notin \mathcal{V}\text{ar}(s)$, $\doteq \in \{\approx, \rhd\}$, and $\theta = \{x \mapsto s\}$,

[t] *removal of trivial equations*

$$\frac{s \doteq s, G}{G} \quad \begin{array}{l} \doteq \in \{\approx, \rhd\} \\ s \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V}) \end{array} \qquad \frac{s \blacktriangleright s, G}{G}$$

The calculus $\text{LCNC}_\ell^{\text{s}}$ imposes the selection strategy given in Table 4. We state the completeness of $\text{LCNC}_\ell^{\text{s}}$ without proof.

THEOREM 5. *Let $\mathcal{R}$ be a deterministic CTRS and θ a normalized strict solution of G. There exists an $\text{LCNC}_\ell^{\text{s}}$-refutation $G \Rightarrow^*_{\theta'} \square$ such that $\theta' \leqslant \theta$ $[\mathcal{V}\text{ar}(G)]$.* □

From Table 4 we learn that $\text{LCNC}_\ell^{\text{s}}$ is nondeterministic only for parameter-passing descendants $s \blacktriangleright t$ with $\text{root}(s) \in \mathcal{F}_\mathcal{D}$ and either $\text{root}(t) \in \mathcal{F}_\mathcal{D}$ or $\text{root}(t) \in \mathcal{V}$. The latter disappears by adopting the left-linearity and freshness conditions from the first half of this section (since it can be shown that the completeness proofs of Theorems 4 and 5 are compatible) and the former disappears by restricting ourselves to left-linear CCSs. Hence we conclude this section by stating that for the class of left-linear fresh deterministic CCSs we have a fully deterministic set of inference rules, which we denote by LCNC_d, that is complete with respect to normalized strict solutions.

6. BENCHMARKS

We compare the three calculi LCNC_ℓ, $\text{LCNC}_\ell^{\text{eve}}$, and LCNC_d on a small number of examples. We implemented these calculi in ρLog [9], a system for rule based programming which

Table 4: $\mathrm{LCNC}^{\mathrm{s}}_{\ell}$: Selection of inference rule.

$s \approx t$

	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$	$\mathcal{V}$
$\mathcal{F}_\mathcal{C}$	[t]; [d]	$[\mathtt{o}]_2$	[v]; [i]
$\mathcal{F}_\mathcal{D}$	$[\mathtt{o}]_1$	$[\mathtt{o}]_1$	$[\mathtt{o}]_1$
$\mathcal{V}$	[v]; [i]	$[\mathtt{o}]_2$	[t]; [v]

$s \rhd t$

	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$	$\mathcal{V}$
$\mathcal{F}_\mathcal{C}$	[t]; [d]	×	[v]; [i]
$\mathcal{F}_\mathcal{D}$	$[\mathtt{o}]_1$	[t]; $[\mathtt{o}]_1$	$[\mathtt{o}]_1$
$\mathcal{V}$	[v]	×	[t]; [v]

$s \blacktriangleright t$

	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$	$\mathcal{V}$
$\mathcal{F}_\mathcal{C}$	[t]; [d]	×	[v]; [i]
$\mathcal{F}_\mathcal{D}$	$[\mathtt{o}]_1$	[t]; ($[\mathtt{o}]_1$, [d])	$[\mathtt{o}]_1$, [i], [v]
$\mathcal{V}$	[v]	[v]	[t]; [v]

Table 5: The goals $\mathsf{fib}(x) \approx \langle x, y\rangle$ and G_n $(0 \leqslant n \leqslant 4)$.

	LCNC_ℓ			$\mathrm{LCNC}^{\mathrm{eve}}_\ell$			LCNC_d		
length	R	N	T	R	N	T	R	N	T
5	2	33	0.06	1	15	0.03	1	15	0.04
10	10	296	0.40	3	69	0.11	1	37	0.06
15	278	3521	4.67	13	484	0.63	2	71	0.11
20	390	39912	99.81	13	3848	5.22	2	113	0.15
25	?	?	?	13	51777	198.26	2	167	0.20

	LCNC_ℓ			$\mathrm{LCNC}^{\mathrm{eve}}_\ell$			LCNC_d		
n	R	N	T	R	N	T	R	N	T
0	4	16	0.04	2	9	0.02	1	8	0.02
1	14	52	0.08	5	20	0.04	2	16	0.03
2	36	128	0.19	9	33	0.06	3	24	0.03
3	82	284	0.40	14	48	0.08	4	32	0.05
4	176	600	0.83	20	65	0.11	5	40	0.06

is built on top of *Mathematica 5*. The benchmarks were run on a PC equipped with a 1.3 GHz Centrino processor, 1 GB RAM memory, and the Windows XP operating system. In Table 5 we use the CTRS for computing Fibonacci numbers given in Section 3 and the goal $G = \mathsf{fib}(x) \approx \langle x, y\rangle$, which admits the two normalized solutions $\{x \mapsto \mathsf{0}, y \mapsto \mathsf{s(0)}\}$ and $\{x \mapsto \mathsf{s(0)}, y \mapsto \mathsf{s(0)}\}$, as well as the goals $G_n = x + y \approx \mathsf{s}^n(\mathsf{0})$ for $0 \leqslant n \leqslant 4$. The search trees for the goals G_n are finite for all three calculi. Since the search tree for G is infinite, we give the number of nodes (N) and the number of refutations (R) for given depths. The time (T) needed by our implementation to compute the (approximated) search trees is listed as well. The "?" entries denote a timeout of 30 minutes.

Comparing the N-columns of Table 5, we observe a dramatic reduction of the search space for solutions as we move from LCNC_ℓ via $\mathrm{LCNC}^{\mathrm{eve}}_\ell$ to LCNC_d. For example, the partial search trees of depth 20 of these calculi have the following sizes: 39912 nodes for LCNC_ℓ, 3848 nodes for $\mathrm{LCNC}^{\mathrm{eve}}_\ell$, and 113 nodes for LCNC_d. A smaller N implies a smaller computation time T, but the relationship is not linear because the node size is not constant and therefore the memory requirements and thus the memory allocation time may vary among nodes.

Next we consider the following specification of the quicksort algorithm:

$$
\begin{aligned}
\mathsf{0} + y &\to y & \mathsf{0} \leq x &\to \mathsf{t}\\
\mathsf{s}(x) + y &\to \mathsf{s}(x + y) & \mathsf{s}(x) \leq \mathsf{0} &\to \mathsf{f}\\
\mathsf{split}(x, [\,]) &\to \langle [\,], [\,]\rangle & \mathsf{s}(x) \leq \mathsf{s}(y) &\to x \leq y\\
\mathsf{split}(x, [y|z]) &\to \langle u, [y|v]\rangle \Leftarrow x \leq y \rhd \mathsf{t}, \mathsf{split}(x, z) \rhd \langle u, v\rangle\\
\mathsf{split}(x, [y|z]) &\to \langle [y|u], v\rangle \Leftarrow x \leq y \rhd \mathsf{f}, \mathsf{split}(x, z) \rhd \langle u, v\rangle\\
\mathsf{app}([\,], y) &\to y & \mathsf{app}([x|y], z) &\to [x|\mathsf{app}(y, z)]\\
\mathsf{qsort}([\,]) &\to [\,]\\
\mathsf{qsort}([x|y]) &\to \mathsf{app}(\mathsf{qsort}(u), [x|\mathsf{qsort}(v)]) \Leftarrow \mathsf{split}(x, y) \rhd \langle u, v\rangle
\end{aligned}
$$

and the goals $H_1 = \mathsf{qsort}([\mathsf{s(0)}, \mathsf{0}, \mathsf{s(s(0))}]) \approx x$ and $H_2 = \mathsf{qsort}([\mathsf{s(0)}, \mathsf{0}, \mathsf{s(s(s(0)))}, \mathsf{s(s(0))}]) \approx x$. Note that these rules constitute a left-linear fresh deterministic CTRS, and therefore we can employ any of our calculi to solve them. The behavior of LCNC_ℓ, $\mathrm{LCNC}^{\mathrm{eve}}_\ell$, and LCNC_d on these goals is shown in Tables 6 and 7. The tables reveal that both memory consumption (N) and computing time (T) become much smaller as we adopt a better refinement. For H_1, LCNC_d generates a finite search space with 193 nodes and a single solution in 0.23 seconds, whereas LCNC_ℓ and $\mathrm{LCNC}^{\mathrm{eve}}_\ell$ require more than 30 minutes to compute search trees that have more than 32000 nodes.

7. CONCLUDING REMARKS

In general, a rewrite step in a CTRS involves the search for suitable instances for the extra variables in the conditions and right-hand sides of the rewrite rules, and this search boils down to solving E-unification problems. Deterministic CTRSs have the nice property that the there is no need for unification during rewriting. When we apply a rule we first match its left-hand side with the expression that we attempt to evaluate and, if the matching succeeds, we evaluate its conditions from left to right. So it is perhaps no surprise that deterministic CTRSs can be automatically transformed into unconditional TRSs. We refrain from giving a formal definition of this transformation—the interested reader is referred to the comprehensive survey of Ohlebusch [15]—but note that the process is similar to lambda-lifting. For the Fibonacci example in Section 3, the conditional rule would be transformed into the two unconditional rules $\mathsf{fib}(\mathsf{s}(x)) \to U(\mathsf{fib}(x))$ and $U(\langle y, z\rangle) \to \langle z, y + z\rangle$ where U is an auxiliary function symbol. Given an arbitrary deterministic CTRS $\mathcal{R}$ and a goal G that we want to solve, there are now two ways to proceed. We can apply a narrowing calculus like LCNC_d developed in this paper with respect to $\mathcal{R}$ and G or we could first transform $\mathcal{R}$ into an unconditional TRS $\mathcal{U}(\mathcal{R})$ and then apply a simpler calculus like LNC_d with respect to $\mathcal{U}(\mathcal{R})$ and G. Which approach is to be preferred?

- By eliminating conditions (or any transformation for that matter) the syntactic structure is modified which may adversely affect the applicability of certain optimizations that rely on syntactic restrictions. For in-

Table 6: The goal H_1.

	LCNC_ℓ			$\text{LCNC}_\ell^{\text{eve}}$			LCNC_d		
length	R	N	T	R	N	T	R	N	T
5	5	35	0.06	1	10	0.03	0	9	0.02
10	43	549	0.71	1	25	0.05	0	24	0.04
15	59	6837	9.23	1	39	0.07	0	38	0.06
20	59	51373	71.89	1	54	0.09	0	53	0.08
25	59	226525	698.47	1	72	0.10	0	71	0.10
50	?	?	?	2729	32535	57.48	0	122	0.13
89	?	?	?	?	?	?	1	193	0.23

Table 7: The goal H_2.

	LCNC_ℓ			$\text{LCNC}_\ell^{\text{eve}}$			LCNC_d		
length	R	N	T	R	N	T	R	N	T
5	5	35	0.06	1	10	0.03	0	9	0.02
10	29	549	0.73	1	25	0.05	0	24	0.04
15	249	6837	12.63	1	39	0.07	0	38	0.06
20	281	51373	371.42	1	54	0.09	0	53	0.08
151	?	?	?	?	?	?	1	363	0.41

stance, if we transform a left-linear fresh deterministic CCS (for which we showed LCNC_d to be complete) into an unconditional TRS, are the syntactic conditions that are needed to ensure the completeness of LNC_d satisfied?

- By introducing auxiliary function symbols, derivations typically will get longer and, unless we adopt strict semantics, computed solutions may contain these symbols. In Table 8 we compare LCNC_d and LNC_d on the goals H_1 and H_2 with respect to the quicksort CTRSs $\mathcal{R}$ and its unconditional counterpart $\mathcal{U}(\mathcal{R})$. There is not much difference in execution speed, but the refutations computed by LCNC_d are clearly shorter.

- The main problem with the transformational approach is that it is not clear whether it is sound for narrowing. Already for the pure rewriting case there exists a surprisingly complicated example by Marchiori (see [15, Example 7.2.14]) which shows that after applying the transformation sketched above the rewrite relation restricted to terms of the original CTRS is enlarged. This example does not involve any extra variables and hence is trivially deterministic.

  Investigating under which conditions known transformations are sound for narrowing and developing transformations that are better suited to work with narrowing we leave as an interesting but challenging topic for further research.

With respect to the last item, it is interesting to mention the recent paper by Antoy *et al.* [2] in which a modification of a transformation of Viry [18] is defined that maps a certain class of left-linear CCSs without extra variables to weakly orthogonal TRSs such that the parallel narrowing strategy of [3] can be used on the target system in order to solve equations in the original CTRSs. Concerning the restriction of disallowing extra variables, the authors write that extra variables can be eliminated by applying lambda-lifting. As indicated above, it is unclear to us whether such a preprocessing step would preserve the solvability of equations.

Another recent paper about transformations and narrowing is [1]. Here a simple transformation (*deconditionalization*) from left-linear CCSs with strict equality to TRSs is defined which preserves (a special form of) normalized rewriting. The proof sketch in the paper does not consider extra variables and from the discussion in Section 4.5 we are unable to reconstruct the intended semantics of rewriting and narrowing in the presence of extra variables.

8. REFERENCES

[1] S. Antoy. Constructor-based conditional narrowing. In *Proc. 3rd PPDP*, pages 199–206, 2001.

[2] S. Antoy, B. Brassel, and M. Hanus. Conditional narrowing without conditions. In *Proc. 5th PPDP*, pages 20–31, 2003.

[3] S. Antoy, R. Echahed, and M. Hanus. Parallel evaluation strategies for functional logic languages. In *Proc. 14th ICLP*, pages 138–152, 1997.

[4] S. Antoy, R. Echahed, and M. Hanus. A needed narrowing strategy. *J. ACM*, 47(4):776–822, 2000.

[5] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[6] H. Ganzinger. Order-sorted completion: The many-sorted way. *TCS*, 89:3–32, November 1991.

[7] J. C. González-Moreno, M. T. Hortalá-González, and M. Rodríguez-Artalejo. Polymorphic types in functional logic programming. *JFLP*, 2001(1), 2001.

[8] M. Hanus. A unified computation model for functional and logic programming. In *Proc. 24th POPL*, pages 80–93, 1997.

Table 8: $\mathrm{LCNC_d}$ with $\mathcal{R}$ versus $\mathrm{LNC_d}$ with $\mathcal{U}(\mathcal{R})$.

	H_1						H_2					
	$\mathrm{LCNC_d}$			$\mathrm{LNC_d}$			$\mathrm{LCNC_d}$			$\mathrm{LNC_d}$		
length	R	N	T	R	N	T	R	N	T	R	N	T
5	0	9	0.02	0	7	0.01	0	9	0.02	0	7	0.01
10	0	24	0.04	0	19	0.04	0	24	0.04	0	19	0.04
15	0	38	0.06	0	37	0.07	0	38	0.06	0	37	0.07
20	0	53	0.08	0	42	0.07	0	53	0.08	0	42	0.07
25	0	71	0.10	0	53	0.08	0	71	0.11	0	53	0.07
50	0	122	0.13	0	104	0.13	0	129	0.17	0	119	0.15
89	1	193	0.23	0	183	0.21	0	210	0.25	0	171	0.21
123				1	233	0.27	0	299	0.36	0	250	0.28
151							1	363	0.41	0	324	0.38
211										1	435	0.50

[9] M. Marin and F. Piroi. Deduction and presentation in ρLog. In *Proc. Mathematical Knowledge Management Symposium*, volume 93 of *ENTCS*, pages 161–182, 2004.

[10] A. Middeldorp and S. Okui. A deterministic lazy narrowing calculus. *JSC*, 25(6):733–757, 1998.

[11] A. Middeldorp, S. Okui, and T. Ida. Lazy narrowing: Strong completeness and eager variable elimination. *TCS*, 167(1,2):95–130, 1996.

[12] A. Middeldorp, T. Suzuki, and M. Hamada. Complete selection functions for a lazy conditional narrowing calculus. *JFLP*, 2002(3), March 2002.

[13] J. J. Moreno-Navarro and M. Rodriguez-Artalejo. Logic programming with functions and predicates: The language BABEL. *JLP*, 12(3-4):191–223, 1992.

[14] E. Ohlebusch. Termination of logic programs: Transformational methods revisited. *AAECC*, 12:73–116, 2001.

[15] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, 2002.

[16] C. Prehofer. *Solving Higher-Order Equations: From Logic to Programming*. Progress in Theoretical Computer Science. Birkhäuser, 1998.

[17] T. Suzuki. Standardization theorem revisited. In *Proc. 5th ALP*, volume 1139 of *LNCS*, pages 122–134, 1996.

[18] P. Viry. Elimination of conditions. *JSC*, 28:381–400, 1999.

[19] T. Yamada, J. Avenhaus, C. Loría-Sáenz, and A. Middeldorp. Logicality of conditional rewrite systems. *TCS*, 236(1,2):209–232, 2000.

APPENDIX

A. PROOF OF THEOREM 2

In this section we prove the completeness of LCNC for deterministic CTRSs. We start by defining some useful concepts. For a substitution θ and a set of variables X, we denote $(X \setminus \mathcal{D}(\theta)) \cup \mathcal{I}_X(\theta)$ by $\mathcal{V}\mathrm{ar}_X(\theta)$. Here $\mathcal{D}(\theta) = \{x \in \mathcal{V} \mid \theta(x) \neq x\}$ denotes the domain of θ, which is always assumed to be finite, and $\mathcal{I}_X(\theta) = \bigcup_{x \in \mathcal{D}(\theta) \cap X} \mathcal{V}\mathrm{ar}(x\theta)$ the set of variables introduced by the restriction of θ to X. The proof of the following lemma is straightforward.

LEMMA 3. *If G is an X-deterministic goal and σ a substitution then $G\sigma$ is $\mathcal{I}_X(\sigma)$-deterministic.* □

For a CTRS $\mathcal{R}$ we let $\mathcal{R}_+ = \mathcal{R} \cup \{x \approx x \to \mathtt{true}, x \rhd x \to \mathtt{true}\}$. We use the variant of conditional rewriting in which the list of instantiated conditions of the applied rewrite rule is explicitly added to the goal after every rewrite step. Formally, we use the *intermediate* rewrite relation $\rightarrowtail$ defined as follows: $G \rightarrowtail G'$ if $G = G_1, e, G_2$, $G' = G_1, c\theta, e', G_2$, and $e \to_{\mathcal{R}} e'$ by applying the rewrite rule $l \to r \Leftarrow c \in \mathcal{R}_+$ with substitution θ. The equation e' is called the *(immediate) descendant* of e in the intermediate rewrite step. The notion of descendant is generalized to arbitrary intermediate rewrite derivations in the obvious way. It is well-known that $\mathcal{R} \vdash G$ if and only if $G \rightarrowtail^* \top$.

Definition 2. A state is a quadruple $\langle G, \theta, \Pi, X\rangle$ consisting of a proper goal G, an X-normalized solution of G, and an intermediate rewrite sequence $\Pi\colon G\theta \rightarrowtail^* \top$ such that always the leftmost equation different from $\mathtt{true}$ is rewritten and only rewrite rules from $\mathcal{R}_0$ are applied to trivial equations. A state $\langle G, \theta, \Pi, X\rangle$ is *deterministic* if G is X-deterministic.

The two assumptions about rewrite proofs are harmless because it is easy to prove that every intermediate rewrite sequence $G\theta \rightarrowtail^* \top$ can be transformed into one which satisfies these assumptions. Without the second assumption, our definition of state coincides with the notion of normalized state in [12, Definition 3.1]. The intended meaning of a state $\langle G, \theta, \Pi, X\rangle$ is as follows:

> Given a goal G with X-normalized solution θ and rewrite proof $\Pi\colon G\theta \rightarrowtail^* \top$, find an LCNC_ℓ-refutation $G \Rightarrow^*_\sigma \square$ such that $\sigma \leqslant \theta\ [\mathcal{V}\mathrm{ar}(G)]$.

To solve this problem, we apply proof steps called *state transformations*. Each proof step takes as input a state $S = \langle G, \theta, \Pi, X\rangle$ and a finite set of variables W, and produces a simpler state $S' = \langle G', \theta', \Pi', X'\rangle$ and an LCNC_ℓ step $\pi\colon G \Rightarrow_\sigma G'$ such that $\theta = \sigma\theta'\ [W]$. We denote this operation by $\langle S', \sigma\rangle = \phi_{\mathrm{LCNC}_\ell}(S, W)$ and depict it as

$$S \xrightarrow[W]{\sigma} S'.$$

The notion of "simpler" is captured by a well-founded relation $\gg$ on states. In this setting, a successful transformation is a finite sequence of proof steps

$$\langle G, \theta, \Pi, X\rangle \xrightarrow[W_1]{\sigma_1} \langle G_1, \theta_1, \Pi_1, X_1\rangle \xrightarrow[W_2]{\sigma_2} \cdots \xrightarrow[W_n]{\sigma_n} \langle \square, \theta_n, \Pi_n, X_n\rangle$$

which we abbreviate to

$$\langle G,\theta,\Pi,X\rangle \overset{\sigma_1\sigma_2\cdots\sigma_n}{\underset{W_1,W_2,\ldots,W_n}{\longrightarrow}} \langle \Box,\theta_n,\Pi_n,X_n\rangle$$

where the sets $W_1,\ldots,W_n$ are chosen in such a way that $\theta = \sigma_1\sigma_2\cdots\sigma_n\theta_n$ $[\mathcal{V}\mathrm{ar}(G)]$. Then, by concatenating the LCNC$_\ell$-steps obtained along the transformation, we obtain the LCNC$_\ell$-refutation $G \Rightarrow^*_\sigma \Box$ with $\sigma = \sigma_1\sigma_2\cdots\sigma_n$, providing an answer to the problem posed by the initial state $\langle G,\theta,\Pi,X\rangle$.

The definition of the state transformation $\phi_{\mathrm{LCNC}_\ell}$ is based on five more elementary state transformations $\phi_{[\alpha]}(S,W)$ with $\alpha \in \{\mathtt{o},\mathtt{i},\mathtt{t},\mathtt{v},\mathtt{d}\}$. These transformations transform a state $S = \langle G,\theta,\Pi,X\rangle$ with $G \neq \Box$ and a finite set of variables W into a simpler state $S' = \langle G',\theta',\Pi',X'\rangle$ and a substitution σ such that $\theta = \sigma\theta'$ $[W]$. In the statements of the following five lemmata we assume that S is a state $\langle G,\theta,\Pi,X\rangle$ with $G = e,G_1$ and that W is a finite set of variables. We refrain from spelling out the proofs since they are obtained by straightforward modifications of the proofs of the transformation lemmata in [12, Lemmata 3.4–3.8].

LEMMA 4. *If $e = s \doteq t$ with $\doteq \in \{\approx,\rhd\}$ and a descendant of $e\theta$ is rewritten at position 1 or if $e = s \doteq t$ with $\doteq = \not\approx$ and a descendant of $e\theta$ is rewritten at position 2, and $l \to r \Leftarrow c \in \mathcal{R}$ is the employed rewrite rule in the first such step then there exists a state $S' = \langle G',\theta',\Pi',X\rangle$ with $G' = s \rhd l,c,r \doteq t,G_1$ such that $\theta' = \theta$ $[W]$ and in Π' no descendant of $s\theta' \rhd l\theta'$ is rewritten at position 1. We denote the pair $\langle S',\epsilon\rangle$ by $\phi_{[\mathtt{o}]}(S,W)$.* □

LEMMA 5. *If $e = f(s_1,\ldots,s_n) \doteq g(t_1,\ldots,t_m)$ with $\doteq \in \{\approx,\rhd\}$ such that no descendant of e is rewritten at position 1 or 2, then $f = g$, $m = n$, and there exists a state $S' = \langle G',\theta,\Pi',X\rangle$ with $G' = s_1 \doteq t_1,\ldots,s_n \doteq t_n,G_1$. We denote the pair $\langle S',\epsilon\rangle$ by $\phi_{[\mathtt{d}]}(S,W)$.* □

LEMMA 6. *If $e = f(s_1,\ldots,s_n) \doteq x$ with $\doteq \in \{\approx,\not\approx,\rhd,\lhd\}$ and $\mathrm{root}(x\theta) = f$ then there exists a state $S' = \langle G\sigma,\theta',\Pi',X'\rangle$ with $X' = \mathcal{I}_X(\sigma)$ such that $\theta = \sigma\theta'$ $[W]$. Here $\sigma = \{x \mapsto f(x_1,\ldots,x_n)\}$ and $x_1,\ldots,x_n \notin X \cup W$. We denote the pair $\langle S',\sigma\rangle$ by $\phi_{[\mathtt{i}]}(S,W)$.* □

LEMMA 7. *If $e = s \doteq x$ with $s \neq x$ and $\doteq \in \{\approx,\not\approx,\rhd,\lhd\}$ such that $e\theta$ is rewritten to* `true` *in the first step of Π then there exists a state $S' = \langle G_1\sigma,\theta',\Pi',X'\rangle$ with $\sigma = \{x \mapsto s\}$ and $X' = \mathcal{I}_X(\sigma)$ such that $\theta = \sigma\theta'$ $[W]$. We denote the pair $\langle S',\sigma\rangle$ by $\phi_{[\mathtt{v}]}(S,W)$.* □

LEMMA 8. *If $e = s \doteq s$ with $\doteq \in \{\approx,\rhd\}$ then there exists a state $S' = \langle G_1,\theta,\Pi',X\rangle$. We denote the pair $\langle S',\epsilon\rangle$ by $\phi_{[\mathtt{t}]}(S,W)$.* □

We denote by $\gg$ the well-founded ordering on states defined in [12, Definition 3.10].

LEMMA 9 ([12, LEMMA 3.11]). *Let S be a state and W a finite set of variables. If $\phi_{[\alpha]}(S,W) = \langle S',\sigma\rangle$ is defined then $S \gg S'$.* □

LEMMA 10. *If a deterministic state S satisfies the conditions of Lemma 4 then $s \notin \mathcal{V}$.*

PROOF. Because θ' is a solution of $G' = s \rhd l,c,r \doteq t,G_1$, we have $s\theta = s\theta' \to^*_{\mathcal{R}} l\theta \to_{\mathcal{R}} r\theta$, so $s\theta$ is not a normal form. Since S is deterministic, $\mathcal{V}\mathrm{ar}(s) \subseteq X$ and $\theta{\restriction_X}$ is normalized. Hence s cannot be a variable. □

LEMMA 11. *Let S be a deterministic state and W a finite set of variables. If $\phi_{[\alpha]}(S,W) = \langle S',\sigma\rangle$ is defined then S' is deterministic.*

PROOF. Let $S = \langle G,\theta,\Pi,X\rangle$ with $G = e,G_1$ and $S' = \langle G',\theta',\Pi',X'\rangle$. We distinguish five cases.

1. If $\alpha = \mathtt{o}$ then $X' = X$, $e = s \doteq t$ with $\doteq \in \{\approx,\not\approx,\rhd\}$ and $G' = s \rhd l,c,r \doteq t,G_1$ for some rewrite rule $l \to r \Leftarrow c \in \mathcal{R}$. We have $\mathcal{V}\mathrm{ar}(s) \subseteq X$ because G is X-deterministic. Since $\mathcal{R}$ is deterministic, c is $\mathcal{V}\mathrm{ar}(l)$-deterministic. We have $\mathcal{V}\mathrm{ar}(r) \subseteq \mathcal{V}\mathrm{ar}(l,c)$. If $\doteq = \rhd$ then G_1 is $X \cup \mathcal{V}\mathrm{ar}(s)$-deterministic and thus G' is X-deterministic. If $\doteq \in \{\approx,\not\approx\}$ then $\mathcal{V}\mathrm{ar}(t) \subseteq X$ and G_1 is $X \cup \mathcal{V}\mathrm{ar}(e)$-deterministic. Hence also in this case we conclude that G' is X-deterministic.
2. If $\alpha = \mathtt{d}$ then $X' = X$, $e = f(s_1,\ldots,s_n) \doteq f(t_1,\ldots,t_n)$ with $\doteq \in \{\approx,\rhd\}$ and $G' = s_1 \doteq t_1,\ldots,s_n \doteq t_n,G_1$. We have $\mathcal{V}\mathrm{ar}(s) \subseteq X$ because G is X-deterministic. If $\doteq = \approx$ then also $\mathcal{V}\mathrm{ar}(t) \subseteq X$ and G_1 is $X \cup \mathcal{V}\mathrm{ar}(e)$-deterministic. If $\doteq = \rhd$ then G_1 is $X \cup \mathcal{V}\mathrm{ar}(s)$-deterministic. In both cases we conclude that G' is X-deterministic.
3. If $\alpha = \mathtt{i}$ then $G' = G\sigma$ and $X' = \mathcal{I}_X(\sigma)$. Lemma 3 yields that G' is X'-deterministic.
4. If $\alpha = \mathtt{v}$ then $e = s \doteq x$, $\sigma = \{x \mapsto s\}$, $G' = G_1\sigma$ and $X' = \mathcal{I}_X(\sigma)$. From Lemma 3 we learn that $G\sigma = s \doteq s,G'$ is X'-deterministic. Since $\mathcal{V}\mathrm{ar}(s) \subseteq X'$, it follows that G' is X'-deterministic as well.
5. Finally, if $\alpha = \mathtt{t}$ then $e = s \doteq s$, $G' = G_1$ and $X' = X$. We have $\mathcal{V}\mathrm{ar}(s) \subseteq X$ because G is X-deterministic. Hence G' is deterministic, too. □

We are now ready to define our state transformation function $\phi_{\mathrm{LCNC}_\ell}$.

LEMMA 12. *Let $\mathcal{R}$ be a deterministic CTRS. For every deterministic state $S = \langle G,\theta,\Pi,X\rangle$ and finite set of variables W there exist a deterministic state $S' = \langle G',\theta',\Pi',X'\rangle$ and an LCNC$_\ell$-step $G \Rightarrow_\sigma G'$ such that $\theta = \sigma\theta'$ $[W]$, $X' = \mathcal{I}_X(\sigma)$, and $S \gg S'$. We denote the pair $\langle S',\sigma\rangle$ by $\phi_{\mathrm{LCNC}_\ell}(S,W)$.*

PROOF. We distinguish the following cases. In each case we assume that the preceding cases are not applicable.

1. If $e = s \doteq s$ with $\doteq \in \{\approx,\rhd\}$ then we let $\phi_{\mathrm{LCNC}_\ell}(S,W) = \phi_{[\mathtt{t}]}(S,W)$. Since $G \Rightarrow_{[\mathtt{t}]} G'$, the result follows from Lemma 8.
2. If $e = f(s_1,\ldots,s_n) \doteq g(t_1,\ldots,t_m)$ with $\doteq \in \{\approx,\rhd\}$ such that no descendant of e is rewritten at position 1 or 2, then we can apply Lemma 5. So we let $\phi_{\mathrm{LCNC}_\ell}(S,W) = \phi_{[\mathtt{d}]}(S,W)$. Since $f = g$ and $m = n$ we have $G \Rightarrow_{[\mathtt{d}]} G'$.
3. If $e = x \rhd t$ with $x \neq t$ then $x\theta \to^*_{\mathcal{R}} t\theta$ because θ is a solution of G. We have $x \in X$ because S is deterministic and thus $x\theta$ is a normal form with respect to $\mathcal{R}$. Hence $x\theta = t\theta$. Since $x \neq t$ this is only possible if $x \notin \mathcal{V}\mathrm{ar}(t)$. It follows that $e\theta$ is rewritten to `true` in the first step of Π. Hence we can apply Lemma 7. So we let $\phi_{\mathrm{LCNC}_\ell}(S,W) = \phi_{[\mathtt{v}]}(S,W)$. Since $x \notin \mathcal{V}\mathrm{ar}(t)$, we indeed have $G \Rightarrow_{[\mathtt{v}],\sigma} G'$.

4. Suppose $e = s \doteq t$ with $\doteq \in \{\approx, \rhd\}$ and a descendant of $e\theta$ is rewritten at position 1 or $e = s \doteq t$ with $\doteq = \asymp$ and a descendant of $e\theta$ is rewritten at position 2. Lemma 4 yields $\phi_{[\mathtt{o}]}(S, W) = \langle S'', \epsilon\rangle$ with $S'' = \langle G'', \theta', \Pi'', X\rangle$ and $G'' = s \rhd f(l_1, \ldots, l_n), c, r \doteq t, G_1$ for some rewrite rule $f(l_1, \ldots, l_n) \to r \Leftarrow c$ of $\mathcal{R}$ such that in Π'' no descendant of $s \rhd f(l_1, \ldots, l_n)$ is rewritten at position 1 or 2. The state S'' is deterministic by Lemma 11 and hence $s \notin \mathcal{V}$ according to Lemma 10. So we have $s = f(s_1, \ldots, s_n)$. Now we apply Lemma 5 to S'' and W. So we define $\phi_{\text{LCNC}_\ell}(S, W) = \phi_{[\mathtt{d}]}(S'', W)$. We have $G' = s_1 \rhd l_1, \ldots, s_n \rhd l_n, c, r \doteq t, G_1$ and thus $G \Rightarrow_{[\mathtt{o}]} G'$ as desired.

5. Let $e = s \doteq x$ such that $s\theta = x\theta$ and either $s \in \mathcal{V} \setminus \{x\}$ and $\doteq = \approx$ or $s \notin \mathcal{V}$ and $\doteq \in \{\approx, \asymp, \rhd\}$. So $e\theta$ is rewritten to `true` in the first step of Π. Since $s \neq x$, we can apply Lemma 7. So we let $\phi_{\text{LCNC}_\ell}(S, W) = \phi_{[\mathtt{v}]}(S, W)$. Since $x \notin \mathcal{V}\mathrm{ar}(s)$, $G \Rightarrow_{[\mathtt{v}],\sigma} G'$ is a valid LCNC$_\ell$-step.

6. In the remaining case we have $e = s \doteq x$ with $s \notin \mathcal{V}$, $\doteq \in \{\approx, \asymp, \rhd\}$, and $s\theta \neq x\theta$. If $\doteq \in \{\approx, \asymp\}$ then $x \in X$ and thus $x\theta$ is a normal form, so $s\theta \to^*_{\mathcal{R}} x\theta$. If $\doteq = \rhd$ then $s\theta \to^*_{\mathcal{R}} x\theta$ by definition. We may assume that all steps in $s\theta \to^*_{\mathcal{R}} x\theta$ take place below the root, since otherwise case 4 would apply. Hence $\mathrm{root}(x\theta) = \mathrm{root}(s)$. Lemma 6 yields $\phi_{[\mathtt{i}]}(S, W) = \langle S'', \sigma\rangle$ with $S'' = \langle G'', \theta', \Pi, X'\rangle$, $\sigma = \{x \mapsto f(x_1, \ldots, x_n)\}$, and $G'' = s\sigma \doteq f(x_1, \ldots, x_n), G_1\sigma$. Next we apply Lemma 5 to S'' and W. So we define $\phi_{\text{LCNC}_\ell}(S, W) = \phi_{[\mathtt{d}]}(S'', W)$ and we obtain $G \Rightarrow_{[\mathtt{i}],\sigma} G'$.

In all cases we obtained $\langle S', \sigma\rangle$ by one or two applications of Lemmata 4–8. So $S \gg S'$ according to Lemma 9. Note that if $\sigma = \epsilon$ then $X' = \mathcal{I}_X(\sigma) = X$. □

We are now ready for the proof of the completeness of LCNC$_\ell$.

THEOREM 2. *Let $\mathcal{R}$ be a deterministic CTRS and G an X-deterministic goal. If θ is an X-normalized solution of G then $G \Rightarrow^*_{\theta'} \square$ for some substitution θ' with $\theta' \leqslant \theta$ $[\mathcal{V}\mathrm{ar}(G)]$.*

PROOF. Since θ is a solution of G, there exists an intermediate rewrite sequence $\Pi\colon G\theta \rightarrowtail^* \top$. We may assume that Π satisfies the conditions stated in Definition 2. Hence $S = \langle G, \theta, \Pi, X\rangle$ is a state. We use induction with respect to the well-founded order $\gg$ on states. To make the induction work we prove that for any finite set of variables W there exists θ' with $G \Rightarrow^*_{\theta'} \square$ and $\theta' \leqslant \theta$ $[W]$. In the base case G is the empty goal and we take the empty LCNC$_\ell$-refutation. If $G \neq \square$ then we perform the following construction:

$$\langle G, \theta, \Pi, X\rangle \overset{\sigma_1}{\underset{W}{\dashrightarrow}} \langle G_1, \theta_1, \Pi_1, X_1\rangle \overset{\sigma_2}{\underset{W_1,\ldots}{\dashrightarrow}} \langle \square, \theta_2, \Pi_2, X_2\rangle$$

Lemma 12 induction hypothesis

where $W_1 = W \cup \mathcal{I}_W(\sigma_1)$ and $\sigma_2 \leqslant \theta_1$ $[W_1]$. Let $\theta' = \sigma_1\sigma_2$. Clearly $\theta' \leqslant \sigma_1\theta_1 = \theta$ $[W]$. We obtain the LCNC$_\ell$-refutation $G \Rightarrow_{\sigma_1} G' \Rightarrow^*_{\sigma_2} \square$. □

B. PROOF OF THEOREM 3

Definition 3. We call $S = \langle G, \theta, \Pi, X\rangle$ with a marked goal G a *marked* state if $\mathrm{u}(S) = \langle \mathrm{u}(G), \theta, \Pi, X\rangle$ is a state. A marked state $\langle G, \theta, \Pi, X\rangle$ is said to be *properly marked* if the following two conditions are satisfied:

1. every unmarked occurrence in G of a variable in X appears in a subterm with marked root symbol,
2. $\mathrm{u}(t)\theta$ is a normal form for every subterm t with marked root symbol that occurs in G.

THEOREM 3. *Let $\mathcal{R}$ be a deterministic CTRS and θ a normalized solution of a proper goal G. There exists an* LCNC$^\dagger_\ell$*-refutation $G^\dagger \Rightarrow^*_{\theta'} \square$ such that $\mathrm{u}(\theta') \leqslant \theta$ $[\mathcal{V}\mathrm{ar}(G)]$.*

PROOF. The key idea is that ϕ_{LCNC_ℓ} can be lifted to properly marked states. First we show that the initial marked state $\langle G^\dagger, \theta, \Pi, X_G\rangle$ is properly marked. The first condition of Definition 3 is satisfied by the definition of $G^\dagger$ and the second condition follows because the only marked subterms in $G^\dagger$ are of the form $x^\dagger$ with $x \in X_G$ and $\theta|_{X_G}$ is normalized by assumption. Next let $S = \langle G, \theta, \Pi, X\rangle$ a properly marked state and W a finite set of variables such that $\phi_{\text{LCNC}_\ell}(\mathrm{u}(S), W) = \langle S', \sigma\rangle$ with $S' = \langle G', \theta', \Pi', X'\rangle$. So $\mathrm{u}(G) \Rightarrow_\sigma G'$. We have to show the existence of a properly marked state $S'' = \langle G'', \theta', \Pi', X'\rangle$ and a marked substitution σ' such that $\mathrm{u}(S'') = S'$, $\mathrm{u}(\sigma') = \sigma$ and $G \Rightarrow_{\sigma'} G''$ is an LCNC$^\dagger_\ell$-step. We follow the case analysis in the proof of Lemma 12. Let $G = e, G_1$.

1. Because $[\mathtt{t}]$ is applicable to any marked version of a trivial equation, we clearly have $G \Rightarrow_{[\mathtt{t}],\epsilon} G''$ for some goal G'' with $\mathrm{u}(G'') = G'$.

2. In this case there is also no problem since in LCNC$^\dagger_\ell$ the rule $[\mathtt{d}]$ is defined for all possible combinations of marking the two root symbols.

3. We have $\mathrm{u}(e) = x \rhd t$ with $x \in X$. Hence $e = x^\dagger \rhd t'$ with $\mathrm{u}(t') = t$. If $t' \notin \mathcal{V} \cup \mathcal{V}^\dagger$ then we obtain $G \Rightarrow_{[\mathtt{v}],\sigma'} G''$ with $\sigma' = \{x, x^\dagger \mapsto t^\dagger\} \cup \{y \mapsto y^\dagger \mid y \in \mathcal{V}\mathrm{ar}(\mathrm{u}(t))\}$. Clearly $\mathrm{u}(G'') = G'$ and $\mathrm{u}(\sigma') = \sigma$. If $t \in \mathcal{V} \cup \mathcal{V}^\dagger$ then we can apply $[\mathtt{v}]$ with binding $\sigma' = \{t, t^\dagger \mapsto x^\dagger\}$, which corresponds to the binding $\sigma = \{\mathrm{u}(t) \mapsto x\}$ that is used in the LCNC$_\ell$-step $\mathrm{u}(G) \Rightarrow_\sigma G'$.

4. We have $\mathrm{u}(e) = f(s_1, \ldots, s_n) \doteq t$ and $f(s_1, \ldots, s_n)\theta$ is reducible. In order to lift the $[\mathtt{o}]$-step, we have to make sure that $e = f(s'_1, \ldots, s'_n) \doteq t'$. Suppose to the contrary that the displayed f in e is marked. Then, according to the second condition in Definition 3, the term $\mathrm{u}(f^\dagger(s'_1, \ldots, s'_n))\theta$ is a normal form. This is impossible as $\mathrm{u}(f^\dagger(s'_1, \ldots, s'_n))\theta = f(s_1, \ldots, s_n)\theta$.

5. In this case we reason as in case 3 above.

6. We have $\mathrm{u}(e) = s \doteq x$ with $s \notin \mathcal{V}$ and $\doteq \in \{\approx, \asymp, \rhd\}$. If $\doteq \in \{\approx, \asymp\}$ then $x \in X$ and thus $e = s' \doteq x^\dagger$ with $\mathrm{u}(s') = s$ and hence we can apply $[\mathtt{i}]$ with binding $\{x \mapsto f(x_1, \ldots, x_n), x^\dagger \mapsto f^\dagger(x_1, \ldots, x_n)\}$. If $\doteq = \rhd$ then there are two possibilities: $e = s' \rhd x^\dagger$ or $e = s' \rhd x$, but in both cases the imitation rule of LCNC$^\dagger_\ell$ is applicable.

We did not show yet that the resulting marked state $S'' = \langle G'', \theta', \Pi', X'\rangle$ is properly marked. Both conditions in Definition 3 are easily proved by inspecting the step $\mathrm{u}(G) \Rightarrow_\sigma G'$ and the inference rules of LCNC$^\dagger_\ell$. □

C. PROOF SKETCH OF THEOREM 4

First we recall the notion of standard intermediate rewrite sequence. In the following we use $Pos_{\mathcal{F}}(t)$ to denote the set of non-variable positions in the term t.

Definition 4. An intermediate rewrite sequence $G \rightarrowtail^* \top$ is called *standard* if $q \backslash p \in \mathcal{P}\mathrm{os}_{\mathcal{F}}(l')$ whenever $G \rightarrowtail^* \top$ can be written as follows:

$$\begin{aligned} G &\rightarrowtail^* \top, e, G' \rightarrowtail_p \top, c\theta, e[r\theta]_p, G' \\ &\rightarrowtail^* \underbrace{\top, e[r\theta]_p, G' \rightarrowtail^* \top, e', G'} \rightarrowtail_q \top, c'\theta', e'[r'\theta']_q, G' \\ &\rightarrowtail^* \top \end{aligned}$$

where $l \to r \Leftarrow c$ and $l' \to r' \Leftarrow c'$ are the rewrite rules used in the $\rightarrowtail_p$ and $\rightarrowtail_q$-steps, such that $\epsilon < q < p$ and in the underbraced part no rewrite rule is applied to a position above p.

THEOREM 6. *Let $\mathcal{R}$ be a left-linear fresh CTRS and G a goal. Every intermediate rewrite sequence $G \rightarrowtail^* \top$ can be transformed into a standard sequence from G to $\top$.*

PROOF. The proof is essentially the same as the proof of standardization for left-linear join CTRSs (Suzuki [17, Theorem 5.10]). The only complication is that we must show that the relation "$\Rightarrow$" for eliminating so-called anti-standard pairs is well-defined, which amounts to proving that rewriting can be performed at positions which are descendants of positions below the pattern position of a $\rightarrowtail$-step, cf. [17, Definition 5.3]. This follows from the simple observation that in fresh CTRSs there are no such descendants in the right-hand sides of oriented equations in the conditions. □

The following example shows the necessity of the freshness condition.

Example 4. Consider the left-linear deterministic CTRS $\mathcal{R}$ of Example 3 and the equation $\mathsf{g}(\mathsf{a}, \mathsf{f}(\mathsf{a})) \approx \mathsf{a}$. The intermediate rewrite sequence

$$\begin{aligned} \mathsf{g}(\mathsf{a}, \mathsf{f}(\mathsf{a})) \approx \mathsf{a} &\rightarrowtail \mathsf{g}(\mathsf{a}, \mathsf{a}) \approx \mathsf{a} \rightarrowtail \mathsf{a} \rhd \mathsf{a}, \mathsf{a} \approx \mathsf{a} \\ &\rightarrowtail \mathtt{true}, \mathsf{a} \approx \mathsf{a} \rightarrowtail \top \end{aligned}$$

is not standard because the first two steps form an anti-standard pair (since rewriting the second argument $\mathsf{f}(\mathsf{a})$ of $\mathsf{g}(\mathsf{a}, \mathsf{f}(\mathsf{a}))$ does not contribute to the creation of the redex pattern $\mathsf{g}(\Box, \Box)$ of the second step.) However, we cannot apply the second rule to $\mathsf{g}(\mathsf{a}, \mathsf{f}(\mathsf{a}))$ because the condition $\mathsf{a} \rhd \mathsf{f}(\mathsf{a})$ is invalid. It follows that there is no standard rewrite sequence $\mathsf{g}(\mathsf{a}, \mathsf{f}(\mathsf{a})) \rightarrowtail^* \top$.

Definition 5. A state $S = \langle G, \theta, \Pi, X \rangle$ is called standard if Π is standard and for every parameter-passing descendant $e = s \rhd t$ in G the following property holds: if a descendant of $e\theta$ is rewritten at position $1p$ and subsequent descendants are not rewritten at a position $\epsilon < q < 1p$ then $p \in \mathcal{P}\mathrm{os}_{\mathcal{F}}(t)$.

The next lemma states that $\phi_{\mathrm{LCNC}_\ell}$ preserves standardness. We omit the proof.

LEMMA 13. *Let $\mathcal{R}$ be a deterministic fresh CTRS and W a finite set of variables. If the state S is standard and $\phi_{\mathrm{LCNC}_\ell}(S, W) = \langle S', \sigma \rangle$ then S' is standard.* □

THEOREM 7. *Let $\mathcal{R}$ be a left-linear fresh deterministic CTRS. Then the calculus obtained from LCNC_ℓ by applying inference rule* [v] *eagerly to descendants $t \rhd x$ of parameter-passing equations with $x \notin \mathcal{V}\mathrm{ar}(t)$ is complete with respect to normalized solutions.*

PROOF. Let θ be a normalized solution of a proper goal G. So there exists a rewrite proof $\Pi\colon G\theta \rightarrowtail^* \top$ which we may assume to be standard according to Theorem 6. Since there are no parameter-passing descendants in G, the initial state $\langle G, \theta, \Pi, X_G \rangle$ is standard. We obtain an LCNC_ℓ-refutation as in the proof of Theorem 2. According to Lemma 13, all states produced along the way are standard. Now suppose that $S' = \langle G', \theta', \Pi', X' \rangle$ is one of these states such that $G' = t \rhd x, G''$ with $t \rhd x$ is a parameter-passing descendant satisfying $x \notin \mathcal{V}\mathrm{ar}(t)$. We have to show that the LCNC_ℓ-step produced in Lemma 12 uses the variable elimination rule. We claim that $x\theta = t\theta$. If $x\theta \neq t\theta$ then in the rewrite proof of the equation $(t \rhd x)\theta'$ in Π' a rewrite rule is applied to the left-hand side $t\theta$. Suppose the last such application is a position p. According to the definition of standard state we must have $p \in \mathcal{P}\mathrm{os}_{\mathcal{F}}(x)$, which is clearly impossible. Hence $x\theta = t\theta$. It follows that case 5 (case 1 if $t \in \mathcal{V}$) of the proof of Lemma 12 applies and thus the resulting LCNC_ℓ-step indeed uses the [v] rule. □

Note that Lemma 1 takes care of parameter-passing descendants of the form $x \rhd t$.

THEOREM 4. *Let $\mathcal{R}$ be a left-linear fresh deterministic CTRS and θ a normalized solution of G. There exists an $\mathrm{LCNC}_\ell^{\mathrm{eve}}$-refutation $G^\dagger \Rightarrow^*_{\theta'} \Box$ such that $\mathrm{u}(\theta') \leqslant \theta$ $[\mathcal{V}\mathrm{ar}(G)]$.*

PROOF. From the preceding theorem we obtain an LCNC_ℓ-refutation $G \Rightarrow^*_{\theta''} \Box$ such that $\theta'' \leqslant \theta$ $[\mathcal{V}\mathrm{ar}(G)]$ and in which all selected parameter-passing descendants $t \rhd x$ with $x \notin \mathcal{V}\mathrm{ar}(t)$ are subjected to the [v] rule. According to the proof of Theorem 3 this refutation can be lifted to an $\mathrm{LCNC}_\ell^\dagger$-refutation. Using $\blacktriangleright$ to denote descendants of parameter-passing equations, we finally obtain an $\mathrm{LCNC}_\ell^{\mathrm{eve}}$-refutation $G^\dagger \Rightarrow^*_{\theta'} \Box$ such that $\mathrm{u}(\theta') = \theta''$ and the selection strategy of Table 3 is respected. □