J. Symbolic Computation (1999) **25**, 733–757

A Deterministic Lazy Narrowing Calculus[†]

AART MIDDELDORP AND SATOSHI OKUI

Institute of Information Sciences and Electronics
University of Tsukuba, Tsukuba 305-8573, Japan
ami@score.is.tsukuba.ac.jp

Faculty of Engineering
Mie University, Tsu 514-8507, Japan
okui@cs.info.mie-u.ac.jp

In this paper we study the non-determinism between the inference rules of the lazy narrowing calculus LNC (Middeldorp *et al.*, 1996). We show that all non-determinism can be removed without losing the important completeness property by restricting the underlying term rewriting systems to left-linear confluent constructor systems and interpreting equality as strict equality. For the subclass of orthogonal constructor systems the resulting narrowing calculus is shown to have the nice property that solutions computed by different derivations starting from the same goal are incomparable.

1. Introduction

Besides being a general method for solving unification problems in equational theories that are presented by confluent term rewriting systems (TRSs for short), narrowing (Hullot, 1980) is the computation model of many functional logic programming languages (Hanus, 1994a). Since narrowing is a complicated operation, various calculi consisting of a small number of more elementary inference rules that simulate narrowing have been proposed, e.g. by Martelli *et al.* (1986, 1989), Hölldobler (1987, 1989), Snyder (1991), Hanus (1994b), Ida and Nakahara (1997). These calculi are highly non-deterministic: in general all choices of (1) the equation in the current goal, (2) the inference rule to be applied, and (3) the rewrite rule of the TRS (for certain inference rules) have to be considered in order to guarantee the desirable property of completeness. In this paper we address the second kind of non-determinism for the calculus LNC of (Middeldorp *et al.*, 1996). This calculus is the specialization of Hölldobler's calculus TRANS (1989), which is defined for general equational systems and based on paramodulation, to (confluent) TRSs and narrowing. The main reason for adopting LNC in this paper is that its completeness has been established for arbitrary confluent TRSs (Middeldorp *et al.*, 1996).

The lazy narrowing calculus LNC consists of the following five inference rules:

[†] A preliminary version of this paper appeared in the Proceedings of the Fuji International Workshop on Functional and Logic Programming, Susuno, World Scientific, pp. 104–118, 1995.

0747–7171/98/000000 + 00 $25.00/0/sy970000

[o] *outermost narrowing*

$$\frac{G', f(s_1, \ldots, s_n) \simeq t, G''}{G', s_1 \approx l_1, \ldots, s_n \approx l_n, r \approx t, G''}$$

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

[i] *imitation*

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

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

[d] *decomposition*

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

[v] *variable elimination*

$$\frac{G', s \simeq x, G''}{(G', G'')\theta}$$

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

[t] *removal of trivial equations*

$$\frac{G', x \approx x, G''}{G', G''}.$$

In the rules [o], [i], and [v], $s \simeq t$ stands for $s \approx t$ or $t \approx s$. Contrary to usual narrowing, the outermost narrowing rule [o] generates new *parameter-passing* equations $s_1 \approx l_1, \ldots, s_n \approx l_n$ besides the *body* equation $r \approx t$. These parameter-passing equations must eventually be solved, but we don't require that they are solved right away.

This paper is a continuation of (Middeldorp *et al.*, 1996). Familiarity with the basics of term rewriting and narrowing will be helpful in the sequel. Surveys can be found in (Dershowitz and Jouannaud, 1990; Klop, 1992). Below we recall some basic concepts.

The narrowing calculus NC consists of the following inference rule:

$$\frac{G', e, G''}{(G', e[r]_p, G'')\theta}$$

if there exist a fresh variant $l \to r$ of a rewrite rule in $\mathcal{R} \cup \{x \approx x \to \mathtt{true}\}$, a non-variable position p in e, and a most general unifier θ of $e_{|p}$ and l.

The rewrite rule $x \approx x \to \mathtt{true}$ is used to simulate syntactic unification. It is similar to reflection in other narrowing calculi. In the above situation we write $G', e, G'' \leadsto_\theta (G', e[r]_p, G'')\theta$. This is called an NC-step. We call $e[r]_p\theta$ the descendant of e. An equation e' in G', G'' has the corresponding equation $e'\theta$ in $(G', G'')\theta$ as descendant. A sequence $G_1 \leadsto_{\theta_1} \cdots \leadsto_{\theta_{n-1}} G_n$ of NC-steps is called an NC-derivation and abbreviated to $G_1 \leadsto_\theta^* G_n$ where $\theta = \theta_1 \cdots \theta_{n-1}$. We use the symbol Π (and its derivatives) to denote NC-derivations. The notion of descendant extends to NC-derivations in the obvious way. For an NC-derivation $\Pi\colon G \leadsto_\theta^* G'$, $\Pi\theta$ denotes the corresponding rewrite sequence $G\theta \to^* G'$. An NC-derivation which ends in $\top$—a generic notation for goals that consist entirely of `true`'s—is called an NC-refutation.

We extend these notions to the calculus LNC. If G and G' are the upper and lower goal in the inference rule [α] ($\alpha \in \{o, i, d, v, t\}$), we write $G \Rightarrow_{[\alpha]} G'$. This is called an LNC-step. The applied rewrite rule or substitution may be supplied as subscript, that is, we will write things like $G \Rightarrow_{[o],\, l \to r} G'$ and $G \Rightarrow_{[i],\, \theta} G'$. A finite LNC-derivation

$G_1 \Rightarrow_{\theta_1} \cdots \Rightarrow_{\theta_{n-1}} G_n$ may be abbreviated to $G_1 \Rightarrow^*_\theta G_n$ where $\theta = \theta_1 \cdots \theta_{n-1}$. An LNC-refutation is an LNC-derivation ending in the empty goal $\square$.

Let θ_1 and θ_2 be substitutions and V a set of variables. We write $\theta_1 \leqslant \theta_2\ [V]$ if there exists a substitution θ such that $x\theta_1\theta = x\theta_2$ for all $x \in V$. In the presence of a TRS $\mathcal{R}$, the relation $\leqslant$ is generalized to $\leqslant_\mathcal{R}$ as follows: $\theta_1 \leqslant_\mathcal{R} \theta_2\ [V]$ if there exists a substitution θ such that $x\theta_1\theta \leftrightarrow^*_\mathcal{R} x\theta_2$ for all $x \in V$. We say that θ_1 and θ_2 are independent on V if neither $\theta_1 \leqslant_\mathcal{R} \theta_2\ [V]$ nor $\theta_2 \leqslant_\mathcal{R} \theta_1\ [V]$. A substitution θ is a solution of a goal G if $s\theta \leftrightarrow^*_\mathcal{R} t\theta$ for every equation $s \approx t$ in G. Solutions θ_1 and θ_2 of G are called incomparable if they are independent on $\mathcal{V}\mathrm{ar}(G)$. For a substitution θ and a set of variables V, we denote $(V \setminus \mathcal{D}(\theta)) \cup \mathcal{I}(\theta|_V)$ by $\mathcal{V}\mathrm{ar}_V(\theta)$. Here $\mathcal{D}(\theta) = \{x \in \mathcal{V} \mid \theta(x) \neq x\}$ denotes the domain of θ and $\mathcal{I}(\theta) = \bigcup_{x \in \mathcal{D}(\theta)} \mathcal{V}\mathrm{ar}(x\theta)$ the set of variables introduced by θ. The empty substitution is denoted by ε.

The set of function symbols $\mathcal{F}$ of a TRS $\mathcal{R}$ is partitioned into disjoint sets $\mathcal{F}_\mathcal{D}$ and $\mathcal{F}_\mathcal{C}$ as follows: a function symbol f belongs to $\mathcal{F}_\mathcal{D}$ if there is a rewrite rule $l \to r$ in $\mathcal{R}$ such that $l = f(l_1, \ldots, l_n)$ for some terms $l_1, \ldots, l_n$, otherwise $f \in \mathcal{F}_\mathcal{C}$. Function symbols in $\mathcal{F}_\mathcal{C}$ are called constructors, those in $\mathcal{F}_\mathcal{D}$ defined symbols. A term built from constructors and variables is called a constructor term. A constructor system (CS for short) is a TRS with the property that the arguments $l_1, \ldots, l_n$ of every left-hand side $f(l_1, \ldots, l_n)$ of a rewrite rule are constructor terms. A left-linear TRS without critical pairs is called orthogonal.

The remainder of this paper is organized as follows. In the next section we discuss the non-determinism in LNC and recall some results of (Middeldorp *et al.*, 1996). In Section 3 we show under what restrictions the non-determinism between the inference rules of LNC for equations that descend from parameter-passing equations can be removed. In Section 4 we do the same for equations that descend from equations in the initial goal. In Section 5 we present our deterministic lazy narrowing calculus. We also prove a minimality result for the computed solutions by this calculus. In Section 6 our calculus is compared with the narrowing calculi of (Ida and Nakahara, 1997) and (González-Moreno *et al.*, 1996). A comparison of the performance of LNC and our deterministic lazy narrowing calculus on a small number of examples confirms our theoretical result.

2. Non-Determinism

There are three sources of non-determinism in LNC: the choice of the equation, the choice of the inference rule, and the choice of the rewrite rule $l \to r$ (in the case of $[o]$). In (Middeldorp *et al.*, 1996) it is shown that all three choices are *don't know* non-deterministic. This means that in general all possible choices have to be considered in order to guarantee completeness, resulting in a huge search space. In particular, LNC lacks *strong* completeness. In other words, completeness is not independent of selection functions. (This contradicts Corollary 7.3.9 in (Hölldobler, 1989). Middeldorp *et al.* showed that LNC is strongly complete whenever *basic* narrowing is complete (Hullot, 1980; Middeldorp and Hamoen, 1994).) Actually things are not that bad. In (Middeldorp *et al.*, 1996) we showed that for completeness it is sufficient to restrict attention to the selection function $\mathcal{S}_{\mathrm{left}}$ that selects the leftmost equation in every goal. So we may assume that G'—the sequence of equations to the left of the selected equation—in the inference rules of LNC is empty. A formal statement of completeness is given below. Observe that we don't require TRSs to be terminating.

THEOREM 2.1. (Middeldorp *et al.*, 1996) *Let $\mathcal{R}$ be a confluent TRS and G a goal. For every normalized solution θ of G there exists an* LNC*-refutation $G \Rightarrow^*_{\theta'} \square$ respecting $\mathcal{S}_{\text{left}}$ such that $\theta' \leqslant \theta\ [\mathcal{V}\text{ar}(G)]$.* $\square$

So the non-determinism of LNC due to the selection of the equation is avoided if we adopt the leftmost selection function. The structure of the proof of Theorem 2.1, to which we refer later, is as follows. First of all, every normalized solution of G is subsumed by a substitution produced by a *normal* NC-refutation starting from G. (This follows from Theorem 4 and Lemma 38 in (Middeldorp *et al.*, 1996).) An NC-refutation $\Pi\colon G \rightsquigarrow^*_\theta \top$ is said to be normal if it respects $\mathcal{S}_{\text{left}}$ and satisfies the following property: if narrowing is applied to the left-hand side (right-hand side) of a descendant of an equation $s \approx t$ in G then $\theta_2{\restriction}_{\mathcal{V}\text{ar}(s\theta_1)}$ ($\theta_2{\restriction}_{\mathcal{V}\text{ar}(t\theta_1)}$) is normalized. Here θ_1 and θ_2 are (uniquely) defined by writing Π as

$$G = G', s \approx t, G'' \quad \rightsquigarrow^*_{\theta_1} \quad \top, (s \approx t, G'')\theta_1 \quad \rightsquigarrow^*_{\theta_2} \quad \top.$$

The main part of the proof of Theorem 2.1 consists in showing that for every non-empty normal NC-refutation $\Pi\colon G \rightsquigarrow^+_\theta \top$ there exist an LNC-step $\Psi_1\colon G \Rightarrow_{\sigma_1} G_1$ respecting $\mathcal{S}_{\text{left}}$ and a normal NC-refutation $\Pi_1\colon G_1 \rightsquigarrow^*_{\theta_1} \top$ such that Π_1 is smaller than Π in some well-founded order $\gg$ (Definition 18 in (Middeldorp *et al.*, 1996)) on NC-refutations and $\sigma_1\theta_1 \leqslant \theta$ for a suitable set of variables V:

$$\begin{array}{llllll} \forall & \text{normal} & \Pi & : & G & \rightsquigarrow^+_\theta \top \\ \exists & & \Psi_1 & : & \Downarrow_{\sigma_1} & \text{such that } \sigma_1\theta_1 \leqslant \theta\ [V] \text{ and } \Pi \gg \Pi_1. \\ \exists & \text{normal} & \Pi_1 & : & G_1 & \rightsquigarrow^*_{\theta_1} \top \end{array}$$

The inference rule employed in the LNC-step Ψ_1 depends on what happens to the selected (leftmost) equation in the given NC-refutation Π, as shown below.

In the following five lemmata Π denotes a normal NC-refutation $G \rightsquigarrow^+_\theta \top$ with $G = s \approx t, G'$ and V denotes a finite set of variables that includes all variables in the initial goal G of Π. The numbers in parentheses refer to the statements in (Middeldorp *et al.*, 1996) whose combination imply the statements below.

LEMMA 2.2. (13, 32, 34) *Suppose narrowing is applied to a descendant of $s \approx t$ in Π at position 1. If $l \to r$ is the applied rewrite rule in the first such step then there exists a normal* NC*-refutation $\phi_{[o]}(\Pi)\colon s \approx l, r \approx t, G' \rightsquigarrow^*_{\theta_1} \top$ such that $\theta_1 = \theta\ [V]$.* $\square$

LEMMA 2.3. (14, 34) *Let $s = f(s_1, \ldots, s_n)$ and $t \in \mathcal{V}$. If $\text{root}(t\theta) = f$ then there exists a normal* NC*-refutation $\phi_{[i]}(\Pi)\colon G\sigma_1 \rightsquigarrow^*_{\theta_1} \top$ such that $\sigma_1\theta_1 = \theta\ [V]$ and $\Pi\theta = \phi_{[i]}(\Pi)\theta_1$. Here $\sigma_1 = \{t \mapsto f(x_1, \ldots, x_n)\}$ with $x_1, \ldots, x_n \notin V$.* $\square$

LEMMA 2.4. (15, 33, 34) *Let $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$, and suppose that narrowing is never applied to a descendant of $s \approx t$ in Π at position 1 or 2. There exists a normal* NC*-refutation $\phi_{[d]}(\Pi)\colon s_1 \approx t_1, \ldots, s_n \approx t_n, G' \rightsquigarrow^*_{\theta_1} \top$ such that $\theta_1 \leqslant \theta\ [V]$.* $\square$

LEMMA 2.5. (16, 34) *Let $t \in \mathcal{V}$, $s \neq t$, and suppose that the first step of Π takes place at the root position. There exists a normal* NC*-refutation $\phi_{[v]}(\Pi)\colon G'\sigma_1 \rightsquigarrow^*_{\theta_1} \top$ with $\sigma_1 = \{t \mapsto s\}$ such that $\sigma_1\theta_1 \leqslant \theta\ [V]$.* $\square$

LEMMA 2.6. (17, 34) *Let $t \in \mathcal{V}$, $s = t$, and suppose that the first step of Π takes place at the root position. There exists a normal* NC*-refutation $\phi_{[t]}(\Pi)\colon G' \leadsto^*_{\theta_1} \top$ such that $\theta_1 \leqslant \theta\ [V]$.* □

The transformations $\phi_{[o]}$ and $\phi_{[d]}$ in Lemmata 2.2 and 2.4 correspond to $\phi_{[o]} \circ \phi_1$ and $\phi_{[d]} \circ \phi_2$ in (Middeldorp *et al.*, 1996). The purpose of ϕ_1 and ϕ_2 is to reorder narrowing steps in such a way that subsequent applications of $\phi_{[o]}$ and $\phi_{[d]}$ (in (Middeldorp *et al.*, 1996)) result in NC-refutations that respect $\mathcal{S}_{\text{left}}$. Since we don't need to know the details of the transformations ϕ_1 and ϕ_2 here, we choose to incorporate them in the definitions of $\phi_{[o]}$ and $\phi_{[d]}$.

According to the next lemma the above transformations decrease the complexity of NC-refutations.

LEMMA 2.7. (20, 32, 33) *Let Π be a normal* NC*-refutation and $\alpha \in \{[o], [i], [d], [v], [t]\}$. We have $\Pi \gg \phi_\alpha(\Pi)$ whenever $\phi_\alpha(\Pi)$ is defined.* □

Lemma 2.8 explains why we don't need the symmetric versions of Lemmata 2.2, 2.3, and 2.5.

LEMMA 2.8. (21, 34) *For every normal* NC*-refutation $\Pi\colon s \approx t, G' \leadsto^*_\theta \top$ there exists a normal* NC*-refutation $\phi_{\text{swap}}(\Pi)\colon t \approx s, G' \leadsto^*_\theta \top$ with the same complexity.* □

The main lemma for establishing the completeness of LNC is now easily proved.

LEMMA 2.9. (36) *For every non-empty normal* NC*-refutation $\Pi\colon G \leadsto^+_\theta \top$ there exist an* LNC*-step $\Psi_1\colon G \Rightarrow_{\sigma_1} G_1$ respecting $\mathcal{S}_{\text{left}}$ and a normal* NC*-refutation $\Pi_1\colon G_1 \leadsto^*_{\theta_1} \top$ such that $\sigma_1\theta_1 \leqslant \theta\ [V]$ and $\Pi \gg \Pi_1$.*

PROOF. We distinguish the following cases, depending on what happens to the selected equation $e = \mathcal{S}_{\text{left}}(G)$ in Π. Let $G = e, G'$ and $e = s \approx t$.

(1) Suppose narrowing is never applied to a descendant of e at position 1 or 2. We distinguish four further cases.

   (a) Suppose $s, t \notin \mathcal{V}$. We may write $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_n)$. Note that the root symbols of s and t must be equal, for otherwise Π cannot be a refutation as narrowing is never applied to a descendant of $s \approx t$ at position 1 or 2. Let $G_1 = s_1 \approx t_1, \ldots, s_n \approx t_n, G'$. We have $\Psi_1\colon G \Rightarrow_{[d]} G_1$. Lemma 2.4 yields a normal NC-refutation $\Pi_1 = \phi_{[d]}(\Pi)\colon G_1 \leadsto^*_{\theta_1} \top$ such that $\theta_1 \leqslant \theta\ [V]$. Take $\sigma_1 = \varepsilon$.
   (b) Suppose $t \in \mathcal{V}$ and $s = t$. In this case the first step of Π_1 must take place at the root of e. Let $G_1 = G'$. We have $\Psi_1\colon G \Rightarrow_{[t]} G_1$. Lemma 2.6 yields a normal NC-refutation $\Pi_1 = \phi_{[t]}(\Pi)\colon G_1 \leadsto^*_{\theta_1} \top$ such that $\theta_1 \leqslant \theta\ [V]$. Take $\sigma_1 = \varepsilon$.
   (c) Suppose $t \in \mathcal{V}$ and $s \neq t$. We distinguish two further cases, depending on what happens to e in the first step of Π.

      (i) Suppose narrowing is applied to e at the root position. Let $\sigma_1 = \{t \mapsto s\}$ and $G_1 = G'\sigma_1$. We have $\Psi_1\colon G \Rightarrow_{[v],\,\sigma_1} G_1$. Lemma 2.5 yields a normal NC-refutation $\Pi_1 = \phi_{[v]}(\Pi)\colon G_1 \leadsto^*_{\theta_1} \top$ such that $\sigma_1\theta_1 \leqslant \theta\ [V]$.

Table 1. Case analysis in the proof of Lemma 2.9.

case	LNC-step	transformation(s)
(1)(a)	$\Rightarrow_{[d]}$	$\phi_{[d]}$
(1)(b)	$\Rightarrow_{[t]}$	$\phi_{[t]}$
(1)(c)(i)	$\Rightarrow_{[v]}$	$\phi_{[v]}$
(1)(c)(ii)	$\Rightarrow_{[i]}$	$\phi_{[d]} \circ \phi_{[i]}$
(1)(d)	$\Rightarrow_{[v]}$ or $\Rightarrow_{[i]}$	$\phi_{[v]} \circ \phi_{\mathrm{swap}}$ or $\phi_{[d]} \circ \phi_{[i]} \circ \phi_{\mathrm{swap}}$
(2)	$\Rightarrow_{[o]}$	$\phi_{[d]} \circ \phi_{[o]}$
(3)	$\Rightarrow_{[o]}$	$\phi_{[d]} \circ \phi_{[o]} \circ \phi_{\mathrm{swap}}$

(ii) Suppose narrowing is not applied to e at the root position. This is only possible if $s \notin \mathcal{V}$. Hence we may write $s = f(s_1, \ldots, s_n)$. Let $\sigma_1 = \{t \mapsto f(x_1, \ldots, x_n)\}$, $G_1 = (s_1 \approx x_1, \ldots, s_n \approx x_n, G')\sigma_1$, and $G_2 = G\sigma_1$. Here $x_1, \ldots, x_n$ are fresh variables. We have $\Psi_1 : G \Rightarrow_{[i],\,\sigma_1} G_1$. From Lemma 2.3 we obtain an NC-refutation $\Pi_2 = \phi_{[i]}(\Pi) : G_2 \rightsquigarrow^*_{\theta_2} \top$ such that $\sigma_1\theta_2 = \theta\ [V]$. Let $V_2 = V \cup \{x_1, \ldots, x_n\}$. Clearly $\mathcal{V}\mathrm{ar}(G_2) \subseteq V_2$. An application of Lemma 2.4 to Π_2 results in an NC-refutation $\Pi_1 = \phi_{[d]}(\Pi_2) : G_1 \rightsquigarrow^*_{\theta_1} \top$ such that $\theta_1 \leqslant \theta_2\ [V_2]$. Using the inclusion $\mathcal{V}\mathrm{ar}_V(\sigma_1) \subseteq V_2$ we obtain $\sigma_1\theta_1 \leqslant \sigma_1\theta_2 = \theta\ [V]$.

(d) In the remaining case we have $t \notin \mathcal{V}$ and $s \in \mathcal{V}$. This case reduces to case (1)(c) by an appeal to Lemma 2.8.

(2) Suppose narrowing is applied to a descendant of e at position 1. Let $l = f(l_1, \ldots, l_n) \to r$ be the used rewrite rule the first time this happens. Because Π is normal, s cannot be a variable. Hence we may write $s = f(s_1, \ldots, s_n)$. Let $G_1 = s_1 \approx l_1, \ldots, s_n \approx l_n, r \approx t, G'$ and $G_2 = s \approx l, r \approx t, G'$. We have $\Psi_1 : G \Rightarrow_{[o]} G_1$. From Lemma 2.2 we obtain an NC-refutation $\Pi_2 = \phi_{[o]}(\Pi) : G_2 \rightsquigarrow^*_{\theta_2} \top$ such that $\theta_2 = \theta\ [V]$. Let $V_2 = V \cup \mathcal{V}\mathrm{ar}(l)$. Clearly $\mathcal{V}\mathrm{ar}(G_2) \subseteq V_2$. An application of Lemma 2.4 to Π_2 results in an NC-refutation $\Pi_1 = \phi_{[d]}(\Pi_2) : G_1 \rightsquigarrow^*_{\theta_1} \top$ such that $\theta_1 \leqslant \theta_2\ [V_2]$. Using $V \subseteq V_2$ we obtain $\theta_1 \leqslant \theta\ [V]$. Take $\sigma_1 = \varepsilon$.

(3) Suppose narrowing is applied to a descendant of e at position 2. This case reduces to the previous one by an appeal to Lemma 2.8.

In all cases we obtain Π_1 from Π by applying one or more transformations $\phi_{[o]}$, $\phi_{[i]}$, $\phi_{[d]}$, $\phi_{[v]}$, $\phi_{[t]}$ together with an additional application of ϕ_{swap} in case (1)(d) and (3). According to Lemmata 2.7 and 2.8 Π_1 has smaller complexity than Π. □

The case analysis in the above proof is summarized in Table 1. The proof of Theorem 2.1 is completed by a straightforward induction argument.

In the present paper we address the non-determinism of LNC due to the selection of the inference rule. Part of the work has been done in (Middeldorp *et al.*, 1996) where the *eager variable elimination problem* is addressed. The non-deterministic application of the various inference rules to selected equations causes LNC to generate many redundant derivations. At several places in the literature it is mentioned that this type of redundancy can be greatly reduced by applying the variable elimination rule $[v]$ prior to other applicable inference rules. In (Middeldorp *et al.*, 1996) it is shown that a restricted version of

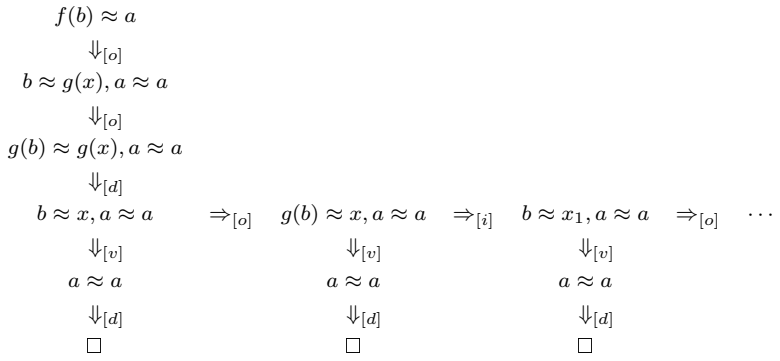

Figure 1. The LNC-refutations starting from $f(b) \approx a$ that respect $\mathcal{S}_{\text{left}}$.

the eager variable elimination strategy is complete (with respect to $\mathcal{S}_{\text{left}}$) for orthogonal TRSs. The definition of the strategy relies on a notion of *descendants* for LNC-derivations. The selected equation $f(s_1, \ldots, s_n) \simeq t$ in the outermost narrowing rule $[o]$ has the body equation $r \approx t$ as only one-step descendant. In the imitation rule $[i]$ all equations $s_i\theta \approx x_i$ $(1 \leqslant i \leqslant n)$ are one-step descendants of the selected equation $f(s_1, \ldots, s_n) \simeq x$. The selected equation $f(s_1, \ldots, s_n) \approx f(t_1, \ldots, t_n)$ in the decomposition rule $[d]$ has all equations $s_1 \approx t_1, \ldots, s_n \approx t_n$ as one-step descendants. Finally, the selected equations in $[v]$ 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 LNC-derivation descends from either a parameter-passing equation or an equation in the initial goal. An equation of the form $x \simeq t$, with $x \notin \mathcal{V}\text{ar}(t)$, is called *solved*. An LNC-derivation is called *eager* if the variable elimination rule $[v]$ is applied to all selected solved equations that are descendants of a parameter-passing equation. This is equivalent to saying that the variable elimination rule $[v]$ is applied to all selected solved equations that do not descend from an equation in the initial goal.

THEOREM 2.10. (Middeldorp *et al.*, 1996) *Let $\mathcal{R}$ be an orthogonal TRS and G a goal. For every normalized solution θ of G there exists an eager* LNC*-refutation $G \Rightarrow^*_{\theta'} \square$ respecting $\mathcal{S}_{\text{left}}$ such that $\theta' \leqslant \theta$ $[\mathcal{V}\text{ar}(G)]$.* $\square$

Consider the orthogonal TRS

$$\mathcal{R} = \left\{ \begin{array}{rcl} f(g(x)) & \to & a \\ b & \to & g(b) \end{array} \right\}$$

and the goal $f(b) \approx a$. Of the infinitely many LNC-refutations that respect the selection function $\mathcal{S}_{\text{left}}$ shown in Figure 1, only the leftmost one is eager.

We conclude this section by sketching the proof of Theorem 2.10. The proof consists of two parts. First it is shown that, as a consequence of the standardization theorem of Huet and Lévy (1991) for orthogonal TRSs, every normalized solution of G is subsumed by a substitution produced by an *outside-in* normal NC-refutation starting from G. An NC-refutation $\Pi\colon G \rightsquigarrow^*_\theta \top$ is said to be outside-in if for every equation $e \in G$ the rewrite

sequence $e\theta \to_{p_1, l_1 \to r_1} \cdots \to_{p_n, l_n \to r_n} \mathtt{true}$ in $\Pi\theta$ satisfies the following condition: for all $1 \leqslant i < n$, if there exists a j with $i < j \leqslant n$ such that $\epsilon < p_j < p_i$ then $p_i \backslash p_j \in \mathcal{P}\mathrm{os}_{\mathcal{F}}(l_j)$ for the least such j. The NC-refutation Π_1 constructed in the proof of Lemma 2.9 is outside-in whenever Π is outside-in:

$$\begin{array}{llllll} \forall & \text{normal outside-in} & \Pi & : & G & \leadsto_{\theta}^{+} \quad \top \\ \exists & & \Psi_1 & : & \Downarrow_{\sigma_1} & \\ \exists & \text{normal outside-in} & \Pi_1 & : & G_1 & \leadsto_{\theta_1}^{*} \quad \top \end{array}$$

In the second part of the proof a property $\mathcal{P}$ of equations in the initial goal of NC-refutations is defined. It is shown that parameter-passing equations introduced in the transformation proof of Lemma 2.9 satisfy this property (Lemma 49 in (Middeldorp *et al.*, 1996)), because we are dealing with outside-in NC-refutations. The property $\mathcal{P}$ is shown to be preserved by LNC-descendants obtained during the transformation proof (Lemma 51). Finally, it is shown that Ψ_1 in Lemma 2.9 consists of a $\Rightarrow_{[v]}$-step whenever the selected (leftmost) equation in Π is solved and satisfies the property $\mathcal{P}$ (Lemma 52).

Orthogonality in Theorem 2.10 is assumed only for allowing the use of the standardization theorem of Huet and Lévy. Recently we learned that standardization holds for arbitrary left-linear TRSs (Boudol, 1985; Suzuki, 1996). Hence we can strengthen Theorem 2.10 as follows.

THEOREM 2.11. *Let $\mathcal{R}$ be a left-linear confluent TRS and G a goal. For every normalized solution θ of G there exists an eager* LNC*-refutation $G \Rightarrow_{\theta'}^{*} \square$ respecting $\mathcal{S}_{\mathrm{left}}$ such that $\theta' \leqslant \theta\ [\mathcal{V}\mathrm{ar}(G)]$.* $\square$

Note that left-linear TRSs are in general not confluent. Consider the left-linear confluent TRS consisting of the rewrite rules

$$\begin{array}{llllllll} \bot \Rightarrow x & \to & \top & \neg\neg x & \to & x & x \vee y & \to & \neg x \Rightarrow y \\ x \Rightarrow \top & \to & \top & \neg\top & \to & \bot & x \wedge y & \to & \neg(\neg x \vee \neg y) \\ \top \Rightarrow \bot & \to & \bot & \neg\bot & \to & \top & & & \end{array}$$

and the goal $x \vee \neg x \approx \top$. One easily verifies that of the infinitely many LNC-refutations respecting $\mathcal{S}_{\mathrm{left}}$ starting from this goal only finitely many are eager.

3. Descendants of Parameter-Passing Equations

In this section we address the remaining non-determinism between the inference rules of LNC for descendants of parameter-passing equations. Consider the TRS

$$\mathcal{R} = \left\{ \begin{array}{rcl} f(a) & \to & f(b) \\ g(f(b)) & \to & c \end{array} \right\}$$

and the goal $g(f(x)) \approx c$. Only the outermost narrowing rule $[o]$ is applicable to this goal, resulting in the new goal $f(x) \approx f(b), c \approx c$. To the parameter-passing equation $f(x) \approx f(b)$ we can either apply the decomposition rule $[d]$ followed by the variable elimination rule $[v]$ or apply $[o]$ followed by $[v]$. In the former case we obtain the solution $\{x \mapsto b\}$ and in the latter the solution $\{x \mapsto a\}$. Since these solutions are incomparable (with respect to $\leqslant_{\mathcal{R}}$), we cannot eliminate the non-determinism between the outermost narrowing rule $[o]$ and the decomposition rule $[d]$ while retaining completeness.

Observe that $\mathcal{R}$ is not a CS because the defined symbol f occurs in the argument of the

left-hand side $g(f(b))$. In this section we will prove that for left-linear confluent CSs all non-determinism between the inferences rules of LNC can be eliminated for descendants of parameter-passing equations.

LEMMA 3.1. *Let $\mathcal{R}$ be a left-linear CS and $G \Rightarrow^* G', s \approx t, G''$ an* LNC*-derivation that respects $\mathcal{S}_{\text{left}}$. If the equation $s \approx t$ descends from a parameter-passing equation then*

(1) $\mathcal{V}\text{ar}(G', s) \cap \mathcal{V}\text{ar}(t) = \varnothing$,
(2) t is a constructor term.

PROOF. Let Π be the given LNC-derivation. We may write Π as

$$G \Rightarrow^* H \Rightarrow_{[o]} H_1, e', H_2 \Rightarrow^* G', s \approx t, G''$$

where e' is the parameter-passing equation of which $s \approx t$ is a descendant. We use induction on the length n of the subderivation $\Pi' \colon H_1, e', H_2 \Rightarrow^* G', s \approx t, G''$ of Π. In order to make the induction work we prove that (2) t is a *linear* constructor term. If $n = 0$ then $s \approx t$ is a parameter-passing equation. Let $s' \approx t'$ be the leftmost equation in H and let $f(l_1, \ldots, l_n) \to r$ be the rewrite rule employed in $H \Rightarrow_{[o]} G', s \approx t, G''$. So $s' = f(s_1, \ldots, s_n)$ or $t' = f(s_1, \ldots, s_n)$ for some terms $s_1, \ldots, s_n$. Because Π respects $\mathcal{S}_{\text{left}}$ we have $G' = s_1 \approx l_1, \ldots, s_{i-1} \approx l_{i-1}$ and $s \approx t = s_i \approx l_i$ for some $1 \leqslant i \leqslant n$. Freshness of the variables in the rewrite rule yields $\mathcal{V}\text{ar}(s_1, \ldots, s_i) \cap \mathcal{V}\text{ar}(l_i) = \varnothing$. The linearity of l implies that $\mathcal{V}\text{ar}(l_1, \ldots, l_{i-1}) \cap \mathcal{V}\text{ar}(l_i) = \varnothing$. Hence (1) is satisfied. For (2) we simply note that proper subterms of left-hand sides of rewrite rules in left-linear CSs are linear constructor terms. Suppose $n > 0$. Write Π' as

$$H_1, e', H_2 \Rightarrow^* H_1', s' \approx t', H_2' \Rightarrow G', s \approx t, G''$$

such that $s \approx t$ descends from $s' \approx t'$. According to the induction hypothesis we have (3) $\mathcal{V}\text{ar}(H_1', s') \cap \mathcal{V}\text{ar}(t') = \varnothing$ and (4) t' is a linear constructor term. First we consider the case that $s' \approx t'$ is selected in the last step of Π'. This is equivalent to $H_1' = \Box$. The last step of Π' cannot be an application of the inference rules $[v]$ and $[t]$ because then $s' \approx t'$ would have no descendants. So we consider the following three cases.

$[o]$ Because t' is a constructor term, its root symbol is not defined and hence we must have $s' = f(s_1, \ldots, s_n)$, $G' = s_1 \approx l_1, \ldots, s_n \approx l_n$, and $s \approx t = r \approx t'$ for some rewrite rule $f(l_1, \ldots, l_n) \to r$. Because the variables in the rewrite rule are fresh we obtain (1) from (3). As $t = t'$, (2) is an immediate consequence of (4).

$[i]$ We have $s' \approx t' = f(s_1, \ldots, s_n) \simeq x$, $G' = s_1\theta \approx x_1, \ldots, s_{i-1}\theta \approx x_{i-1}$, and $s \approx t = s_i\theta \approx x_i$ for fresh variables $x_1, \ldots, x_n$ and substitution $\theta = \{x \mapsto f(x_1, \ldots, x_n)\}$. Property (3) implies $x \notin \mathcal{V}\text{ar}(s_1, \ldots, s_n)$ and thus $s_i\theta = s_i$ for all $1 \leqslant i \leqslant n$. Freshness of the variables $x_1, \ldots, x_n$ implies the desired (1). Term t' is a variable, hence (2) is trivially satisfied.

$[d]$ In this case we have $s' = f(s_1, \ldots, s_n)$, $t' = f(t_1, \ldots, t_n)$, $G' = s_1 \approx t_1, \ldots, s_{i-1} \approx t_{i-1}$, and $s \approx t = s_i \approx t_i$ for some $1 \leqslant i \leqslant n$. The linearity of t' implies that $\mathcal{V}\text{ar}(t_1, \ldots, t_{i-1}) \cap \mathcal{V}\text{ar}(t_i) = \varnothing$. From (3) we infer that $\mathcal{V}\text{ar}(s_1, \ldots, s_i) \cap \mathcal{V}\text{ar}(t_i) = \varnothing$. Hence (1) holds. Because t_i is a (proper) subterm of t', (2) follows from (4).

Next we consider the case that $s' \approx t'$ is not selected in the last step of Π'. Let θ be the produced substitution in the last step of Π'. We have $s \approx t = s'\theta \approx t'\theta$. If the applied inference rule is $[i]$ or $[v]$ then $\mathcal{D}(\theta) \subseteq \mathcal{V}\text{ar}(H_1')$, otherwise $\theta = \varepsilon$ and thus also $\mathcal{D}(\theta) \subseteq \mathcal{V}\text{ar}(H_1')$. From (3) we obtain $t = t'$ and thus (2) follows from (4). We

Table 2. Selection of inference rule for descendant $s \approx t$ of parameter-passing equation.

$\mathrm{root}(s) \backslash \mathrm{root}(t)$	$\mathcal{V}$	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$
$\mathcal{V}$	$[v]$	$[v]$	$\times$
$\mathcal{F}_\mathcal{C}$	$[v]$	$[d]$	$\times$
$\mathcal{F}_\mathcal{D}$	$[v]$	$[o]$	$\times$

either have $\mathcal{V}\mathrm{ar}(G', s') \subseteq \mathcal{V}\mathrm{ar}(H'_1, s')$ (when the applied inference rule is $[d]$, $[v]$, or $[t]$) or $\mathcal{V}\mathrm{ar}(G', s') \subseteq \mathcal{V}\mathrm{ar}(H'_1, s') \cup V$ for some set V of fresh variables (when the applied inference rule is $[o]$ or $[i]$). Hence in all cases we obtain (1) from (3). □

One might think that Lemma 3.1(1) holds for arbitrary left-linear TRSs. This is not true. Consider for example the TRS

$$\mathcal{R} = \left\{ \begin{array}{rcl} f(g(x)) & \to & a \\ g(x) & \to & h(x,x) \end{array} \right\}$$

and the LNC-derivation

$$\begin{array}{lll} f(h(x,x)) \approx a & \Rightarrow_{[o]} & h(x,x) \approx g(x_1), a \approx a \\ & \Rightarrow_{[o]} & x_1 \approx x_2, h(x_2,x_2) \approx h(x,x), a \approx a \\ & \Rightarrow_{[v]} & h(x_2,x_2) \approx h(x,x), a \approx a \\ & \Rightarrow_{[d]} & x_2 \approx x, x_2 \approx x, a \approx a \\ & \Rightarrow_{[v]} & x \approx x, a \approx a. \end{array}$$

The equation $x \approx x$ descends from the parameter-passing equation $h(x,x) \approx g(x_1)$ generated in the first $\Rightarrow_{[o]}$-step.

The first part of Lemma 3.1 implies in particular that $\mathcal{V}\mathrm{ar}(s) \cap \mathcal{V}\mathrm{ar}(t) = \varnothing$ for every descendant $s \approx t$ of a parameter-passing equation. Hence we can forget about the occur-check in the variable elimination rule $[v]$ when dealing with such equations. The second part of the lemma implies that the outermost narrowing rule $[o]$ is only applicable to the left-hand side of descendants of parameter-passing equations. Moreover, if $[o]$ can be applied, then the decomposition rule $[d]$ is not applicable. Combining these observations with Theorem 2.11 yields complete determinism in the choice of inference rule for descendants of parameter-passing equations, provided of course we are dealing with left-linear confluent CSs. Table 2 shows how the inference rule is completely determined by the root symbols of both sides of the selected descendant $s \approx t$ of a parameter-passing equation.

The case $\mathrm{root}(t) \in \mathcal{F}_\mathcal{D}$ is impossible according to Lemma 3.1(2). Observe that the imitation rule $[i]$ is never applied to descendants of parameter-passing equations. This is because if $[i]$ is applicable then, according to Lemma 3.1(1), so is the variable elimination rule $[v]$ and by Theorem 2.11 the latter is given precedence.

Incorporating the above observations into LNC gives rise to the calculus $\mathrm{LNC}_{\mathrm{pp}}$ whose inference rules are presented below. To distinguish descendants of parameter-passing equations from descendants of initial equations, we use $\triangleright$ rather than $\approx$ to denote the former. The first group of rules are designed for descendants of initial equations. The only difference with the inference rules of LNC specialized to $\mathcal{S}_{\mathrm{left}}$ is that in the outermost narrowing rule we mark the parameter-passing equations using $\triangleright$:

[o] *outermost narrowing*

$$\frac{f(s_1,\ldots,s_n) \simeq t, G}{s_1 \triangleright l_1, \ldots, s_n \triangleright l_n, r \approx t, G}$$

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

[i] *imitation*

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

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

[d] *decomposition*

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

[v] *variable elimination*

$$\frac{s \simeq x, G}{G\theta}$$

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

[t] *removal of trivial equations*

$$\frac{x \approx x, G}{G}$$

The second group of inference rules of $\mathrm{LNC}_{\mathrm{pp}}$ deals with descendants of parameter-passing equations:

[o]$_\triangleright$ *outermost narrowing for parameter-passing equations*

$$\frac{f(s_1,\ldots,s_n) \triangleright t, G}{s_1 \triangleright l_1, \ldots, s_n \triangleright l_n, r \triangleright t, G}\ t \notin \mathcal{V}$$

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

[d]$_\triangleright$ *decomposition for parameter-passing equations*

$$\frac{f(s_1,\ldots,s_n) \triangleright f(t_1,\ldots,t_n), G}{s_1 \triangleright t_1, \ldots, s_n \triangleright t_n, G}\ f \in \mathcal{F}_\mathcal{C}$$

[v]$_\triangleright$ *variable elimination for parameter-passing equations*

$$\frac{s \triangleright x, G}{G\theta} \quad \text{and} \quad \frac{x \triangleright s, G}{G\theta}\ s \notin \mathcal{V}$$

if $\theta = \{x \mapsto s\}$.

THEOREM 3.2. *Let $\mathcal{R}$ be a left-linear confluent CS and G a goal. For every normalized solution θ of G there exists an $\mathrm{LNC}_{\mathrm{pp}}$-refutation $G \Rightarrow^*_{\theta'} \square$ such that $\theta' \leqslant \theta\ [\mathcal{V}\mathrm{ar}(G)]$.* $\square$

In the next section we turn our attention to descendants of equations in the initial goal.

4. Descendants of Initial Equations

Whereas the restriction to left-linear confluent CSs is sufficient to remove all non-determinism in the choice of inference rule for descendants of parameter-passing equations, this is not the case for descendants of initial equations. Consider for example the

(left-linear confluent) CS

$$\mathcal{R} = \left\{ \begin{array}{lcl} f(a) & \to & f(b) \end{array} \right\}$$

and the goal $f(x) \approx f(b)$. This goal has the two incomparable solutions $\{x \mapsto a\}$ and $\{x \mapsto b\}$. The first solution can only be obtained if we apply the outermost narrowing rule $[o]$. The second solution requires an application of the decomposition rule $[d]$.

There is also non-determinism in the outermost narrowing rule $[o]$ itself. Consider for example the CS

$$\mathcal{R} = \left\{ \begin{array}{lcl} f(a) & \to & g(a) \\ g(b) & \to & f(b) \end{array} \right\}$$

and the goal $f(x) \approx g(x)$. The solution $\{x \mapsto a\}$ can only be obtained if we apply the outermost narrowing rule $[o]$ to the left-hand side of $f(x) \approx g(x)$, but in order to obtain the incomparable solution $\{x \mapsto b\}$ it is essential that we apply $[o]$ to its right-hand side.

In functional logic programming it is customary to consider two expressions to be equal if and only if they reduce to the same ground constructor normal form. This so-called *strict equality* is adopted to model non-termination correctly (Giovannetti *et al.*, 1991; Antoy *et al.*, 1994). If we interpret $\approx$ as strict equality then the non-determinism in the above examples disappears: neither $\{x \mapsto a\}$ nor $\{x \mapsto b\}$ are (strict) solutions of the goals $f(x) \approx f(b)$ and $f(x) \approx g(x)$. In the following definition we are slightly less restrictive.

Definition 4.1. Let $\mathcal{R}$ be a TRS and G a goal. A substitution θ is said to be a *strict solution* of G if for every equation $s \approx t$ in G there exists a constructor term u such that $s\theta \to_{\mathcal{R}}^* u$ and $t\theta \to_{\mathcal{R}}^* u$.

Note that we don't require that the constructor term u in the above definition is ground. Also note that a strict solution may substitute non-constructor terms for variables. (See however Lemma 5.3.) In this section we will prove that for confluent TRSs all non-determinism between the inference rules of LNC can be eliminated for descendants of initial equations with strict semantics. We would like to stress that we don't need the restriction to left-linear CSs here. The structure of the proof is similar to that of the second half of the proof of Theorem 2.10: we define a property of initial equations in NC-refutations (Definition 4.2), we show that initial equations of the LNC-refutation obtained in the transformation proof of Theorem 2.1 satisfy this property (Lemma 4.3), we show that the property is preserved by LNC-descendants obtained during the transformation in the proof of Theorem 2.1 (Lemma 4.4), and finally we show that there is no non-determinism between the inference rules for selected equations that have the property (Lemma 4.5).

Definition 4.2. Let $\Pi\colon G \rightsquigarrow_\theta^* \top$ be an NC-refutation and $e \in G$. We say that e is *strictly solved* in Π if the sequence starting from $e\theta$ in the corresponding rewrite sequence $\Pi\theta\colon G\theta \to^* \top$ is of the form $e\theta \to^* t \approx t \to \mathtt{true}$ with t a constructor term.

Lemma 4.3. *Let G be a goal and θ a substitution such that $\theta|_{\mathcal{V}\mathrm{ar}(G)}$ is normalized. If θ is a strict solution of G then there exists a normal* NC*-refutation $\Pi\colon G \rightsquigarrow_{\theta'}^* \top$ such that $\theta' \leqslant \theta\ [\mathcal{V}\mathrm{ar}(G)]$ and every equation in G is strictly solved in Π.*

Proof. By assumption there exists a rewrite sequence $G\theta \to^* \top$ with the property that

for every $e \in G$ the corresponding subsequence $e\theta \to^* \texttt{true}$ is of the form $e\theta \to^* t \approx t \to \texttt{true}$ with $t \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$. Because steps in different equations of $G\theta$ are independent of each other, we may assume that this rewrite sequence respects $\mathcal{S}_{\text{left}}$. An application of the lifting lemma for NC results in an NC-refutation $\Pi\colon G \rightsquigarrow^*_{\theta'} \top$ respecting $\mathcal{S}_{\text{left}}$ such that $\theta' \leqslant \theta\ [\mathcal{V}\text{ar}(G)]$. Because $\theta|_{\mathcal{V}\text{ar}(G)}$ is normalized, so is $\theta'|_{\mathcal{V}\text{ar}(G)}$. It follows that Π is a normal NC-refutation. It remains to show that every equation $e \in G$ is strictly solved in Π. Because $G\theta \to^* \top$ is an instance of $\Pi\theta'$, the rewrite sequence starting from $e\theta'$ in $\Pi\theta'$ must be of the form $e\theta' \to^* t' \approx t' \to \texttt{true}$ with $t' \leqslant t$ for some $t \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$. This implies that $t' \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$. □

In the following two lemmata, Π_1 and Ψ_1 refer to the NC-refutation and the LNC-step constructed in Lemma 2.9.

LEMMA 4.4. *Suppose $\Pi\colon G \rightsquigarrow^+ \top$ is a normal* NC*-refutation and let e be an equation in G that is strictly solved in Π. If e' is a descendant of e with respect to Ψ_1 then e' is strictly solved in Π_1.*

PROOF. First we consider the case that $e = s \approx t$ is the selected equation in G. Consider the case analysis in the proof of Lemma 2.9. In cases (1)(b) and (1)(c)(i) there is nothing to show as e has no descendants in G_1. In case (1)(a) we have $s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$, and $e' = s_i \approx t_i$ for some $1 \leqslant i \leqslant n$. Because e is strictly solved in Π, the equation $e\theta$ is rewritten in $\Pi\theta$ to an equation of the form $f(u_1, \ldots, u_n) \approx f(u_1, \ldots, u_n)$ with $f(u_1, \ldots, u_n) \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$. Hence, by construction of $\phi_{[d]}$, in $\Pi_1\theta_1$ the equation $e'\theta_1$ is rewritten to $u_i \approx u_i$. Since subterms of constructor terms are constructor terms, we conclude that e' is strictly solved in Π_1. In case (1)(c)(ii) we have $s = f(s_1, \ldots, s_n)$, $t \in \mathcal{V}$, and $e' = s_i \approx x_i$ for some $1 \leqslant i \leqslant n$. From $\phi_{[i]}(\Pi)\theta_2 = \Pi\theta$ we infer that $e\sigma_1$ is strictly solved in $\Pi_2 = \phi_{[i]}(\Pi)$. By the same reasoning as in case (1)(a)—note that e' is a descendant of $e\sigma_1$ with respect to the LNC-step $G_2 \Rightarrow_{[d]} G_1$—we conclude that e' is strictly solved in $\Pi_1 = \phi_{[d]}(\Pi_2)$. In case (2) the rewrite sequences starting from $e\theta$ in $\Pi\theta$ and $(r \approx t)\theta_2$ in $\phi_{[o]}(\Pi)\theta_2$ can be written as†

$$\begin{array}{lllll} (s \approx t)\theta & \to^* & l\theta' \approx t\theta & & \\ & & \downarrow & & \\ (r \approx t)\theta_2 & \to^* & r\theta' \approx t\theta & \to^* & \texttt{true} \end{array}$$

for some substitution θ'. Because by assumption $s \approx t$ is strictly solved in Π, the sequence from $r\theta' \approx t\theta$ to $\texttt{true}$ has the form $r\theta' \approx t\theta \to^* u \approx u \to \texttt{true}$ with $u \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$. Hence $r \approx t$ is strictly solved in $\Pi_2 = \phi_{[o]}(\Pi)$. Since the rewrite sequence starting from $(r \approx t)\theta_2$ in $\phi_{[o]}(\Pi)\theta_2$ is an instance‡ of the rewrite sequence starting from $(r \approx t)\theta_1$ in $\phi_{[d]}(\Pi_2)\theta_1$, the equation $r \approx t$ is also strictly solved in $\Pi_1 = \phi_{[d]}(\Pi_2)$. Observe that $r \approx t$ is the (only) descendant of e with respect to $\Psi_1\colon G \Rightarrow_{[o]} G_1$. Finally, case (3) reduces to case (2) because strict solvability is trivially preserved by ϕ_{swap}.

Next we consider the case that $e \in G$ is not selected in the first step of Π. The (unique) descendant $e' = e\sigma_1 \in G_1$ of e inherits strict solvability from e as the rewrite sequence starting from $e\theta$ in $\Pi\theta$ is an instance‡ of the rewrite sequence from $e'\theta_1$ in $\Pi_1\theta_1$. □

† This follows from Lemma 32 and the proof of Lemma 13 in (Middeldorp *et al.*, 1996).

‡ Modulo an inconsequential reordering of rewrite steps due to the transformations ϕ_1 and ϕ_2 which are part of $\phi_{[o]}$ and $\phi_{[d]}$, cf. the text following Lemma 2.6.

It is interesting to note that parameter-passing equations generated in Lemma 2.9 are in general not strictly solved. Consider for example the TRS

$$\mathcal{R} = \left\{ \begin{array}{rcl} f(x) & \to & a \\ b & \to & b \end{array} \right\}$$

and the normal NC-refutation $\Pi\colon f(b) \approx a \rightsquigarrow a \approx a \rightsquigarrow \mathtt{true}$. The equation $f(b) \approx a$ is strictly solved in Π because a is a constructor. Since in the first step of Π narrowing is applied at position 1, we obtain

$$\begin{array}{lcccccc} \Pi & : & f(b) \approx a & \rightsquigarrow & a \approx a & \rightsquigarrow & \mathtt{true} \\ \Psi_1 & : & \Downarrow_{[o]} & & & & \\ \Pi_1 & : & b \approx x, a \approx a & \rightsquigarrow & \mathtt{true}, a \approx a & \rightsquigarrow & \top \end{array}$$

from Lemma 2.9. The parameter-passing equation $b \approx x$ is not strictly solved in Π_1 as b is a defined symbol.

LEMMA 4.5. *Let $\Pi\colon G \rightsquigarrow^+ \top$ be a normal* NC*-refutation and suppose that the selected (leftmost) equation $e = s \approx t$ is strictly solved in Π.*

(1) Ψ_1 consists of an $\Rightarrow_{[o]}$-step if and only if $\mathrm{root}(s) \in \mathcal{F}_\mathcal{D}$ or $\mathrm{root}(t) \in \mathcal{F}_\mathcal{D}$. Moreover, if $[o]$ is applied to the right-hand side t of e then $\mathrm{root}(s) \notin \mathcal{F}_\mathcal{D}$.
(2) Ψ_1 consists of a $\Rightarrow_{[v]}$-step if and only if e is solved and $s, t \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$.

PROOF. The following observations follow easily from the fact that e is strictly solved in Π: if $\mathrm{root}(s) \in \mathcal{F}_\mathcal{D}$ then narrowing is applied to a descendant of e in Π at position 1, if $\mathrm{root}(t) \in \mathcal{F}_\mathcal{D}$ then narrowing is applied to a descendant of e in Π at position 2, and if $s \notin \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$ or $t \notin \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$ then narrowing is not applied at the root position in the first step of Π.

(1) According to the proof of Lemma 2.9 Ψ_1 consists of an $\Rightarrow_{[o]}$-step only in cases (2) and (3). In case (2) we have $\mathrm{root}(s) \in \mathcal{F}_\mathcal{D}$ and in case (3) $\mathrm{root}(t) \in \mathcal{F}_\mathcal{D}$. Conversely, if $\mathrm{root}(s) \in \mathcal{F}_\mathcal{D}$ then, according to one of the above observations, narrowing is applied to a descendant of e in Π at position 1. This means that we are in case (2) of the proof of Lemma 2.9, and thus Ψ_1 consists of an $\Rightarrow_{[o]}$-step. The case $\mathrm{root}(t) \in \mathcal{F}_\mathcal{D}$ is similar. Since case (2) precedes case (3) in the proof of Lemma 2.9, $[o]$ is applied to the right-hand side t of e only if case (2) does not apply, so $\mathrm{root}(s) \notin \mathcal{F}_\mathcal{D}$.
(2) According to the proof of Lemma 2.9 Ψ_1 consists of a $\Rightarrow_{[v]}$-step only in cases (1)(c)(i) and (1)(d). In both cases narrowing is applied at the root position in the first step of Π. This is only possible if the equation e is solved and, according to one of the above observations, $s, t \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$. Conversely, assume that e is solved and $s, t \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$. This is only possible if narrowing is applied at the root position in the first step of Π, hence we are in case (1)(a), (1)(b), (1)(c)(i), or (1)(d) of the proof of Lemma 2.9. In cases (1)(a) and (1)(b) the equation e is not solved. In the other two cases Ψ_1 indeed consists of a $\Rightarrow_{[v]}$-step.

□

Lemmata 4.3–4.5 imply that for completeness (with respect to strict solutions) it suffices to apply a single inference rule of LNC to each goal whose selected leftmost equation

Table 3. Selection of inference rule for strictly solved equation $s \approx t$.

$\mathrm{root}(s) \backslash^{\mathrm{root}(t)}$	$\mathcal{V}$	$\mathcal{F}_\mathcal{C}$	$\mathcal{F}_\mathcal{D}$
$\mathcal{V}$	$[v]/[t]^a$	$[v]/[i]^b$	$[o]$
$\mathcal{F}_\mathcal{C}$	$[v]/[i]^c$	$[d]$	$[o]$
$\mathcal{F}_\mathcal{D}$	$[o]$	$[o]$	$[o]^d$

[a] $[t]$ is applied if and only if $s = t$.
[b] $[v]$ is applied if and only if $t \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$ and $s \notin \mathcal{V}\mathrm{ar}(t)$.
[c] $[v]$ is applied if and only if $s \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$ and $t \notin \mathcal{V}\mathrm{ar}(s)$.
[d] $[o]$ is applied to the left-hand side s.

descends from an equation in the initial goal. According to Lemma 4.5 the unique inference rule is completely determined by the form of the selected equation. Table 3 shows how the inference rule depends on the selected strictly solved equation $s \approx t$.

It is interesting to note that the resulting strategy is almost the opposite of eager variable elimination: conflicts between the variable elimination rule $[v]$ and the outermost narrowing rule $[o]$ are always resolved by giving preference to the latter and often the imitation rule $[i]$ is selected even if $[v]$ is applicable.

We describe a further restriction on the applicability of the imitation rule $[i]$ for descendants of initial equations in the case of confluent TRSs. Consider the goal $x \approx c(x)$ with c a constructor. Applying $[i]$, the only applicable inference rule, results in a variant of $x \approx c(x)$. Hence LNC produces an infinite derivation. Since the goal $x \approx c(x)$ has no strict solution, we might as well stop the generation of new goals. More generally, a goal $x \simeq t$ with $x \neq t$ has no strict solution if there is an occurrence of x in t such that only constructors appear along the path from the root of t to this occurrence. This motivates the following definition.

DEFINITION 4.6. Let t be a term. We define the subset $\mathcal{V}\mathrm{ar}_\mathcal{C}(t)$ of $\mathcal{V}\mathrm{ar}(t)$ inductively as follows:

$$\mathcal{V}\mathrm{ar}_\mathcal{C}(t) = \begin{cases} \{t\} & \text{if } t \in \mathcal{V}, \\ \bigcup_{i=1}^{n} \mathcal{V}\mathrm{ar}_\mathcal{C}(t_i) & \text{if } t = f(t_1, \ldots, t_n) \text{ with } f \in \mathcal{F}_\mathcal{C}, \\ \varnothing & \text{otherwise.} \end{cases}$$

LEMMA 4.7. *Let $\mathcal{R}$ be a confluent TRS. An equation $x \simeq t$ with $x \neq t$ has no strict solution if $x \in \mathcal{V}\mathrm{ar}_\mathcal{C}(t)$.*

PROOF. There exists a position $p \in \mathcal{P}\mathrm{os}_\mathcal{V}(t)$ such that $t_{|p} = x$ and $\mathrm{root}(t_{|q}) \in \mathcal{F}_\mathcal{C}$ for all positions $q < p$. Note that $p \neq \epsilon$. Suppose to the contrary that θ is a strict solution of $x \simeq t$. So $t\theta$ and $x\theta$ both reduce to some constructor term u. Because in the rewrite sequence $t\theta \to_\mathcal{R}^* u$ no steps takes place at a position above p, we have $x\theta = t\theta_{|p} \to_\mathcal{R}^* u_{|p}$. However, as $p \neq \epsilon$, u and $u_{|p}$ are different normal forms of $x\theta$, contradicting confluence. □

So we require that $x \notin \mathcal{V}\mathrm{ar}_{\mathcal{C}}(t)$ when we apply the imitation rule to a descendant $x \simeq t$ of an initial equation. The same condition appears in the "partial binding of variables to constructor terms" of (Hanus, 1994b) in the setting of ground confluent and terminating CSs.

Note that a goal $x \simeq t$ with $x \neq t$ and $x \in \mathcal{V}\mathrm{ar}_{\mathcal{C}}(t)$ may have a solution. For instance, $\{x \mapsto a\}$ is a (non-strict) solution of the goal $x \approx c(x)$ with respect to the confluent CS $\{a \to c(a)\}$. Nevertheless, it is not difficult to show that LNC is not able to compute solutions to such goals. (This doesn't contradict the completeness of LNC for arbitrary confluent TRSs, which is stated only for *normalized*, or equivalently, *normalizable* solutions.) Hence we can impose the condition $x \notin \mathcal{V}\mathrm{ar}_{\mathcal{C}}(t)$ also when applying the imitation rule to a descendant $x \simeq t$ of a parameter-passing equation, without affecting the completeness of LNC. Since in the final calculus described in the next section there is no imitation rule for descendants of parameter-passing equations, we chose not to do so.

Incorporating the above observations into LNC gives rise to the calculus $\mathrm{LNC_i}$ whose inference rules are presented below. We use $\equiv$ to denote descendants of initial equations. The first group of rules are designed for descendants of initial equations. Concerning the formulation of $[i]_{\equiv}$, note that $x \notin \mathcal{V}\mathrm{ar}_{\mathcal{C}}(f(s_1,\dots,s_n))$ and either $x \in \mathcal{V}\mathrm{ar}(f(s_1,\dots,s_n))$ or $f(s_1,\dots,s_n) \notin \mathcal{T}(\mathcal{F}_{\mathcal{C}},\mathcal{V})$ is equivalent to $x \notin \mathcal{V}\mathrm{ar}_{\mathcal{C}}(f(s_1,\dots,s_n))$ and $f(s_1,\dots,s_n) \notin \mathcal{T}(\mathcal{F}_{\mathcal{C}},\mathcal{V})$.

$[o]_{\equiv}$ *outermost narrowing for initial equations*

$$\frac{f(s_1,\dots,s_n) \equiv t, G}{s_1 \approx l_1,\dots,s_n \approx l_n, r \equiv t, G} \quad \text{and} \quad \frac{t \equiv f(s_1,\dots,s_n), G}{s_1 \approx l_1,\dots,s_n \approx l_n, r \equiv t, G}\ \mathrm{root}(t) \notin \mathcal{F}_{\mathcal{D}}$$

if there exists a fresh variant $f(l_1,\dots,l_n) \to r$ of a rewrite rule in $\mathcal{R}$,

$[i]_{\equiv}$ *imitation for initial equations*

$$\frac{f(s_1,\dots,s_n) \equiv x, G}{(s_1 \equiv x_1,\dots,s_n \equiv x_n, G)\theta}\ f \in \mathcal{F}_{\mathcal{C}} \quad \text{and} \quad \frac{x \equiv f(s_1,\dots,s_n), G}{(s_1 \equiv x_1,\dots,s_n \equiv x_n, G)\theta}\ f \in \mathcal{F}_{\mathcal{C}}$$

if $x \notin \mathcal{V}\mathrm{ar}_{\mathcal{C}}(f(s_1,\dots,s_n))$, $f(s_1,\dots,s_n) \notin \mathcal{T}(\mathcal{F}_{\mathcal{C}},\mathcal{V})$, and $\theta = \{x \mapsto f(x_1,\dots,x_n)\}$ with $x_1,\dots,x_n$ fresh variables,

$[d]_{\equiv}$ *decomposition for initial equations*

$$\frac{f(s_1,\dots,s_n) \equiv f(t_1,\dots,t_n), G}{s_1 \equiv t_1,\dots,s_n \equiv t_n, G}\ f \in \mathcal{F}_{\mathcal{C}}$$

$[v]_{\equiv}$ *variable elimination for initial equations*

$$\frac{s \equiv x, G}{G\theta} \quad \text{and} \quad \frac{x \equiv s, G}{G\theta}\ s \notin \mathcal{V}$$

if $x \notin \mathcal{V}\mathrm{ar}(s)$, $s \in \mathcal{T}(\mathcal{F}_{\mathcal{C}},\mathcal{V})$, and $\theta = \{x \mapsto s\}$,

$[t]_{\equiv}$ *removal of trivial initial equations*

$$\frac{x \equiv x, G}{G}$$

The second group of inference rules of $\mathrm{LNC_i}$ deals with descendants of parameter-passing equations. They are the same as the inference rules of LNC with respect to $\mathcal{S}_{\mathrm{left}}$.

$[o]$ *outermost narrowing*

$$\frac{f(s_1,\dots,s_n) \simeq t, G}{s_1 \approx l_1,\dots,s_n \approx l_n, r \approx t, G}$$

if there exists a fresh variant $f(l_1,\dots,l_n) \to r$ of a rewrite rule in $\mathcal{R}$,

[*i*] *imitation*

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

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

[*d*] *decomposition*

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

[*v*] *variable elimination*

$$\frac{s \simeq x, G}{G\theta}$$

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

[*t*] *removal of trivial equations*

$$\frac{x \approx x, G}{G}$$

THEOREM 4.8. *Let $\mathcal{R}$ be a confluent TRS and G a goal. For every strict and normalized solution θ of G there exists an* LNC$_\mathrm{i}$*-refutation $G \Rightarrow^*_{\theta'} \square$ such that $\theta' \leqslant \theta$ $[\mathcal{V}\mathrm{ar}(G)]$.*

Note that in the above theorem G is assumed to be a sequence $s_1 \equiv t_1,\ldots,s_n \equiv t_n$ of initial equations. The calculus LNC$_\mathrm{i}$ can be slightly improved by replacing the selected equation $f(s_1,\ldots,s_n) \simeq t$ in the outermost narrowing rule [*o*] (for parameter-passing equations) by $f(s_1,\ldots,s_n) \approx t$, i.e., there is no need to apply outermost narrowing to right-hand sides of descendants of parameter-passing equations. (The reason for this is that the proof of Lemma 49 in (Middeldorp *et al.*, 1996) reveals that the first part of property $\mathcal{P}$ holds for arbitrary normal NC-refutations.)

5. The Final Calculus

The results of the preceding two sections give rise to the *deterministic* lazy narrowing calculus, LNC$_\mathrm{d}$ for short, whose inference rules are presented below. The first group of rules are designed for descendants of initial equations:

[o]$_\equiv$ *outermost narrowing for initial equations*

$$\frac{f(s_1,\ldots,s_n) \equiv t, G}{s_1 \triangleright l_1,\ldots,s_n \triangleright l_n, r \equiv t, G} \quad\text{and}\quad \frac{t \equiv f(s_1,\ldots,s_n), G}{s_1 \triangleright l_1,\ldots,s_n \triangleright l_n, r \equiv t, G}\ \mathrm{root}(t) \notin \mathcal{F}_\mathcal{D}$$

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

[i]$_\equiv$ *imitation for initial equations*

$$\frac{f(s_1,\ldots,s_n) \equiv x, G}{(s_1 \equiv x_1,\ldots,s_n \equiv x_n, G)\theta}\ f \in \mathcal{F}_\mathcal{C} \quad\text{and}\quad \frac{x \equiv f(s_1,\ldots,s_n), G}{(s_1 \equiv x_1,\ldots,s_n \equiv x_n, G)\theta}\ f \in \mathcal{F}_\mathcal{C}$$

if $x \notin \mathcal{V}\mathrm{ar}_\mathcal{C}(f(s_1,\ldots,s_n))$, $f(s_1,\ldots,s_n) \notin \mathcal{T}(\mathcal{F}_\mathcal{C},\mathcal{V})$, and $\theta = \{x \mapsto f(x_1,\ldots,x_n)\}$ with $x_1,\ldots,x_n$ fresh variables,

[d]$_\equiv$ *decomposition for initial equations*

$$\frac{f(s_1,\ldots,s_n) \equiv f(t_1,\ldots,t_n), G}{s_1 \equiv t_1,\ldots,s_n \equiv t_n, G}\ f \in \mathcal{F}_\mathcal{C}$$

$[v]_{\equiv}$ *variable elimination for initial equations*

$$\frac{s \equiv x, G}{G\theta} \quad \text{and} \quad \frac{x \equiv s, G}{G\theta}\ s \notin \mathcal{V}$$

if $x \notin \mathcal{V}\mathrm{ar}(s)$, $s \in \mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$, and $\theta = \{x \mapsto s\}$,

$[t]_{\equiv}$ *removal of trivial initial equations*

$$\frac{x \equiv x, G}{G}$$

The second group of inference rules of $\mathrm{LNC_d}$ deals with descendants of parameter-passing equations:

$[o]_{\triangleright}$ *outermost narrowing for parameter-passing equations*

$$\frac{f(s_1, \ldots, s_n) \triangleright t, G}{s_1 \triangleright l_1, \ldots, s_n \triangleright l_n, r \triangleright t, G}\ t \notin \mathcal{V}$$

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

$[d]_{\triangleright}$ *decomposition for parameter-passing equations*

$$\frac{f(s_1, \ldots, s_n) \triangleright f(t_1, \ldots, t_n), G}{s_1 \triangleright t_1, \ldots, s_n \triangleright t_n, G}\ f \in \mathcal{F}_\mathcal{C}$$

$[v]_{\triangleright}$ *variable elimination for parameter-passing equations*

$$\frac{s \triangleright x, G}{G\theta} \quad \text{and} \quad \frac{x \triangleright s, G}{G\theta}\ s \notin \mathcal{V}$$

if $\theta = \{x \mapsto s\}$.

Observe that there is no non-determinism between the inference rules of $\mathrm{LNC_d}$. This does not imply that every goal admits at most one $\mathrm{LNC_d}$-refutation because the outermost narrowing rules $[o]_{\equiv}$ and $[o]_{\triangleright}$ depend on the choice of the rewrite rule $f(l_1, \ldots, l_n) \to r$. This latter non-determinism cannot be avoided because goals in general have different incomparable (strict) solutions.

Combining the results of the preceding sections yields the following completeness theorem.

THEOREM 5.1. *Let $\mathcal{R}$ be a left-linear confluent CS and G a goal. For every strict and normalized solution θ of G there exists an $\mathrm{LNC_d}$-refutation $G \Rightarrow^*_{\theta'} \Box$ such that $\theta' \leqslant \theta\ [\mathcal{V}\mathrm{ar}(G)]$.*

PROOF. According to Theorem 2.1 there exists an LNC-refutation $\Psi\colon G \Rightarrow^*_{\theta'} \Box$ respecting $\mathcal{S}_{\mathrm{left}}$ such that $\theta' \leqslant \theta\ [\mathcal{V}\mathrm{ar}(G)]$. According to the proof of Theorem 3.2 Ψ is an $\mathrm{LNC_{pp}}$-refutation. According to the proof of Theorem 4.8 Ψ is an $\mathrm{LNC_i}$-refutation. We conclude that Ψ is an $\mathrm{LNC_d}$-refutation. $\Box$

We would like to emphasize that the results in Sections 3 and 4 are independent of each other. So if we are faced with a confluent TRS that is not a left-linear CS, we can still remove all non-determinism due to the selection of the inference rule for descendants of initial equations, provided we adopt strict semantics. Only if both restrictions (left-linear CS and strict semantics) are fulfilled, we can use the fully deterministic calculus $\mathrm{LNC_d}$.

In the remainder of this section we show that substitutions computed by different $\mathrm{LNC_d}$ derivations are incomparable, provided we are dealing with *orthogonal* CSs. Hence for

this subclass of TRSs LNC_d can be seen as a minimal unification procedure. First we show that the substitutions produced by LNC_d map variables in the initial goal to constructor terms, provided we are dealing with left-linear CSs. This is not entirely obvious as the $[v]_\triangleright$ inference rule may yield non-constructor substitutions. Actually, this can only happen with the first half of $[v]_\triangleright$. Hence we find it convenient to split the $[v]_\triangleright$ inference rule into two parts:

$$[v]^1_\triangleright \quad \frac{s \triangleright x, G}{G\theta} \qquad \text{and} \qquad [v]^2_\triangleright \quad \frac{x \triangleright s, G}{G\theta}\ s \notin \mathcal{V}$$

with $\theta = \{x \mapsto s\}$.

DEFINITION 5.2. Goals appearing in an LNC_d-derivation can be written as

$$G = s_1 \triangleright t_1, \ldots, s_m \triangleright t_m, u_1 \equiv v_1, \ldots, u_n \equiv v_n$$

for some $m, n \geqslant 0$. We partition $\mathcal{V}\text{ar}(G)$ into two sets: $\mathcal{V}\text{ar}_\text{pp}(G) = \mathcal{V}\text{ar}(t_1, \ldots, t_m)$ and $\mathcal{V}\text{ar}_\text{i}(G) = \mathcal{V}\text{ar}(G) \setminus \mathcal{V}\text{ar}_\text{pp}(G)$.

LEMMA 5.3. *Let $\mathcal{R}$ be a left-linear CS and G a goal. If $G \Rightarrow^*_\theta G'$ is an LNC_d-derivation then*

(1) $\mathcal{V}\text{ar}_\text{i}(G') \subseteq \mathcal{V}\text{ar}(G\theta)$,
(2) $\mathcal{V}\text{ar}_\text{pp}(G') \cap \mathcal{V}\text{ar}(G\theta) = \varnothing$,
(3) $\theta{\restriction_{\mathcal{V}\text{ar}(G)}}$ *is a constructor substitution.*

PROOF. We use induction on the length of the given LNC_d-derivation Ψ. The case of zero length is trivial: $G' = G$ and $\theta = \varepsilon$ and thus $\mathcal{V}\text{ar}_\text{i}(G') = \mathcal{V}\text{ar}(G)$ and $\mathcal{V}\text{ar}_\text{pp}(G') = \varnothing$. Let $\Psi\colon G \Rightarrow^*_{\theta_1} G_1 \Rightarrow_{[\alpha],\,\theta_2} G'$ with $\theta = \theta_1\theta_2$. The induction hypothesis yields $\mathcal{V}\text{ar}_\text{i}(G_1) \subseteq \mathcal{V}\text{ar}(G\theta_1)$, $\mathcal{V}\text{ar}_\text{pp}(G_1) \cap \mathcal{V}\text{ar}(G\theta_1) = \varnothing$, and $\theta_1{\restriction_{\mathcal{V}\text{ar}(G)}}$ is a constructor substitution. If $[\alpha]$ is $[d]_\equiv$ or $[d]_\triangleright$ then $\mathcal{V}\text{ar}_\text{i}(G') = \mathcal{V}\text{ar}_\text{i}(G_1)$, $\mathcal{V}\text{ar}_\text{pp}(G') = \mathcal{V}\text{ar}_\text{pp}(G_1)$, and $\theta_2 = \varepsilon$. Hence $\theta = \theta_1$ and thus $\mathcal{V}\text{ar}_\text{i}(G') \subseteq \mathcal{V}\text{ar}(G\theta)$, $\mathcal{V}\text{ar}_\text{pp}(G') \cap \mathcal{V}\text{ar}(G\theta) = \varnothing$, and $\theta{\restriction_{\mathcal{V}\text{ar}(G)}}$ is a constructor substitution. If $[\alpha]$ is $[o]_\equiv$ or $[o]_\triangleright$ then $\mathcal{V}\text{ar}_\text{i}(G') = \mathcal{V}\text{ar}_\text{i}(G_1)$, $\mathcal{V}\text{ar}_\text{pp}(G') = \mathcal{V}\text{ar}_\text{pp}(G_1) \cup V$ where V are the (fresh) variables in the employed rewrite rule, and $\theta_2 = \varepsilon$. Since $V \cap \mathcal{V}\text{ar}(G\theta_1) = \varnothing$ we obtain $\mathcal{V}\text{ar}_\text{i}(G') \subseteq \mathcal{V}\text{ar}(G\theta)$, $\mathcal{V}\text{ar}_\text{pp}(G') \cap \mathcal{V}\text{ar}(G\theta) = \varnothing$, and $\theta{\restriction_{\mathcal{V}\text{ar}(G)}}$ is a constructor substitution, as before. In the remaining cases we reason as follows:

$[i]_\equiv$ We have $\mathcal{V}\text{ar}_\text{pp}(G') = \varnothing$, $\mathcal{V}\text{ar}_\text{i}(G') = (\mathcal{V}\text{ar}_\text{i}(G_1) \setminus \{x\}) \cup \{x_1, \ldots, x_n\}$, and $\theta_2 = \{x \mapsto f(x_1, \ldots, x_n)\}$ with $f \in \mathcal{F}_\mathcal{C}$. Since $\mathcal{V}\text{ar}_\text{i}(G_1) = \mathcal{V}\text{ar}(G_1)$ we obtain $\mathcal{V}\text{ar}_\text{i}(G') = \mathcal{V}\text{ar}(G_1\theta_2) \subseteq \mathcal{V}\text{ar}(G\theta)$. Note that θ_2 and hence $\theta{\restriction_{\mathcal{V}\text{ar}(G)}} = (\theta_1\theta_2){\restriction_{\mathcal{V}\text{ar}(G)}}$ is a constructor substitution.

$[v]_\equiv$ We have $\mathcal{V}\text{ar}_\text{pp}(G') = \varnothing$, $\mathcal{V}\text{ar}_\text{i}(G') \subseteq \mathcal{V}\text{ar}_\text{i}(G_1\theta_2)$ and $\theta_2 = \{x \mapsto s\}$ for some constructor term s. From $\mathcal{V}\text{ar}_\text{i}(G_1) = \mathcal{V}\text{ar}(G_1)$ we obtain $\mathcal{V}\text{ar}_\text{i}(G') \subseteq \mathcal{V}\text{ar}(G_1\theta_2) \subseteq \mathcal{V}\text{ar}(G\theta)$. Since θ_2 is constructor substitution, so is $\theta{\restriction_{\mathcal{V}\text{ar}(G)}}$.

$[t]_\equiv$ We have $\mathcal{V}\text{ar}_\text{i}(G') \subseteq \mathcal{V}\text{ar}_\text{i}(G_1)$, $\mathcal{V}\text{ar}_\text{pp}(G') = \varnothing$, and $\theta_2 = \varepsilon$. From this we immediately obtain the desired result.

$[v]^1_\triangleright$ We have $\theta_2 = \{x \mapsto s\}$ with $x \in \mathcal{V}\text{ar}_\text{pp}(G_1)$. Since $\mathcal{V}\text{ar}_\text{pp}(G_1) \cap \mathcal{V}\text{ar}(G\theta_1) = \varnothing$ we obtain $G\theta = G\theta_1$ and hence $\theta{\restriction_{\mathcal{V}\text{ar}(G)}} = \theta_1{\restriction_{\mathcal{V}\text{ar}(G)}}$ is a constructor substitution. From Lemma 3.1(1) we learn that $\mathcal{V}\text{ar}_\text{pp}(G') = \mathcal{V}\text{ar}_\text{pp}(G_1) \setminus \{x\}$, which easily yields $\mathcal{V}\text{ar}_\text{pp}(G') \cap \mathcal{V}\text{ar}(G\theta) = \varnothing$. Again with help of Lemma 3.1(1) we obtain $\mathcal{V}\text{ar}_\text{i}(G') \subseteq \mathcal{V}\text{ar}_\text{i}(G_1)$ and thus $\mathcal{V}\text{ar}_\text{i}(G') \subseteq \mathcal{V}\text{ar}(G\theta_1) = \mathcal{V}\text{ar}(G\theta)$.

$[v]^2_{\triangleright}$ According to Lemma 3.1(2) the substitution $\theta_2 = \{x \mapsto s\}$ in this case is a constructor substitution. Hence so is $\theta\restriction_{\mathcal{V}\mathrm{ar}(G)}$. From Lemma 3.1(1) we learn that $\mathcal{V}\mathrm{ar}_{\mathrm{pp}}(G') = \mathcal{V}\mathrm{ar}_{\mathrm{pp}}(G_1)\backslash\mathcal{V}\mathrm{ar}(s)$. Since $x \in \mathcal{V}\mathrm{ar}_{\mathrm{i}}(G_1) \subseteq \mathcal{V}\mathrm{ar}(G\theta_1)$ we obtain $\mathcal{V}\mathrm{ar}(G\theta) = (\mathcal{V}\mathrm{ar}(G\theta_1)\backslash \{x\}) \cup \mathcal{V}\mathrm{ar}(s)$. Combining these observations with $\mathcal{V}\mathrm{ar}_{\mathrm{pp}}(G_1) \cap \mathcal{V}\mathrm{ar}(G\theta_1) = \varnothing$ yields $\mathcal{V}\mathrm{ar}_{\mathrm{pp}}(G') \cap \mathcal{V}\mathrm{ar}(G\theta) = \varnothing$. It remains to show that $\mathcal{V}\mathrm{ar}_{\mathrm{i}}(G') \subseteq \mathcal{V}\mathrm{ar}(G\theta)$. This is an easy consequence of the inclusion $\mathcal{V}\mathrm{ar}_{\mathrm{i}}(G') \subseteq (\mathcal{V}\mathrm{ar}_{\mathrm{i}}(G_1) \setminus \{x\}) \cup \mathcal{V}\mathrm{ar}(s)$, which follows with help of Lemma 3.1(1).

□

Note that the restriction to CSs in the above lemma is essential: for the TRS $\{f(f(a)) \to a\}$ and the goal $f(x) \equiv a$, $\mathrm{LNC_d}$ computes the non-constructor substitution $\{x \mapsto f(a)\}$.

The next lemma states that when comparing constructor substitutions with respect to a confluent TRS, we can ignore the TRS.

LEMMA 5.4. *Let $\mathcal{R}$ be a confluent TRS and V a set of variables. If $\theta_1\restriction_V$ and $\theta_2\restriction_V$ are constructor substitutions then $\theta_1 \leqslant_{\mathcal{R}} \theta_2\ [V]$ if and only if $\theta_1 \leqslant \theta_2\ [V]$.*

PROOF. Clearly $\theta_1 \leqslant \theta_2\ [V]$ implies $\theta_1 \leqslant_{\mathcal{R}} \theta_2\ [V]$. Suppose $\theta_1 \leqslant_{\mathcal{R}} \theta_2\ [V]$. So there exists a substitution θ such that $x\theta_1\theta \leftrightarrow^*_{\mathcal{R}} x\theta_2$ for all $x \in V$. Without loss of generality we assume that $\mathcal{D}(\theta) \subseteq \mathcal{V}\mathrm{ar}_V(\theta_1)$. Since $x\theta_2$ is a normal form and $\mathcal{R}$ is confluent we have $x\theta_1\theta \to^*_{\mathcal{R}} x\theta_2$, for all $x \in V$. We define a substitution θ' that satisfies $x\theta_1\theta' = x\theta_2$ for all $x \in V$. For every $y \in \mathcal{D}(\theta)$ there exists an $x \in V$ and a position $p \in \mathcal{P}\mathrm{os}(x\theta_1)$ such that $y = x\theta_{1|p}$. Moreover, because $x\theta_1$ is a constructor term, $y\theta \to^*_{\mathcal{R}} x\theta_{2|p}$. We define $\theta'(y) = x\theta_{2|p}$. It remains to show that $\theta'(y)$ does not depend on the choice of x and p. If $y = x'\theta_{1|p'}$ for some $x' \in V$ and position $p' \in \mathcal{P}\mathrm{os}(x'\theta_1)$ then $y\theta \to^*_{\mathcal{R}} x'\theta_{2|p'}$. Since $\theta_2\restriction_V$ is a constructor substitution, $x\theta_{2|p}$ and $x'\theta_{2|p'}$ are normal forms of $y\theta$. Confluence yields $x\theta_{2|p} = x'\theta_{2|p'}$. Hence θ' is well-defined. We have $\theta_1\theta' = \theta_2\ [V]$ by construction of θ'. □

We need one more result, before we are in a position to prove our minimality result.

LEMMA 5.5. *Let $\mathcal{R}$ be a left-linear CS. Consider an $\mathrm{LNC_d}$-derivation $G \Rightarrow^*_\theta e, G'$.*

(1) If $e = s \equiv t$ then $\mathcal{V}\mathrm{ar}(e) \subseteq \mathcal{V}\mathrm{ar}(G\theta)$.
(2) If $e = s \triangleright t$ then $\mathcal{V}\mathrm{ar}(s) \subseteq \mathcal{V}\mathrm{ar}(G\theta)$.

PROOF. Easy consequence of Lemmata 5.3(1) and 3.1(1). □

THEOREM 5.6. *Let $\mathcal{R}$ be an orthogonal CS and G a goal. If $G \Rightarrow^*_\theta \square$ and $G \Rightarrow^*_{\theta'} \square$ are different $\mathrm{LNC_d}$-refutations then θ and θ' are incomparable.*

PROOF. Let us denote the two $\mathrm{LNC_d}$-refutations by Ψ and Ψ'. Suppose to the contrary that θ and θ' are not independent on $\mathcal{V}\mathrm{ar}(G)$. So either $\theta \leqslant_{\mathcal{R}} \theta'\ [\mathcal{V}\mathrm{ar}(G)]$ or $\theta' \leqslant_{\mathcal{R}} \theta\ [\mathcal{V}\mathrm{ar}(G)]$. Without loss of generality we assume the former. According to Lemma 5.4 $\theta \leqslant \theta'\ [\mathcal{V}\mathrm{ar}(G)]$. Since the only non-determinism in $\mathrm{LNC_d}$ is the choice of the rewrite rule in the inference rules $[o]_\equiv$ and $[o]_\triangleright$, we may write $\Psi\colon G \Rightarrow^*_{\theta_1} G_1 \Rightarrow_{[\alpha]} G_2 \Rightarrow^*_{\theta_2} \square$ and $\Psi'\colon G \Rightarrow^*_{\theta_1} G_1 \Rightarrow_{[\alpha]} G'_2 \Rightarrow^*_{\theta'_2} \square$ where $\theta = \theta_1\theta_2$, $\theta' = \theta_1\theta'_2$, and either $[\alpha] = [o]_\equiv$ or $[\alpha] = [o]_\triangleright$. From $\theta \leqslant \theta'\ [\mathcal{V}\mathrm{ar}(G)]$ we infer that $\theta_2 \leqslant \theta'_2\ [\mathcal{V}\mathrm{ar}(G\theta_1)]$. So there exists a substitution ρ such that $\theta_2\rho = \theta'_2\ [\mathcal{V}\mathrm{ar}(G\theta_1)]$. First we consider the

case that $[\alpha] = [o]_{\triangleright}$. Write $G_1 = f(s_1, \ldots, s_n) \triangleright t, G_3$ and let $l = f(l_1, \ldots, l_n) \to r$ and $l' = f(l'_1, \ldots, l'_n) \to r'$ be the employed rewrite rules in Ψ and Ψ'. We have $G_2 = s_1 \triangleright l_1, \ldots, s_n \triangleright l_n, r \triangleright t, G_3$ and $G'_2 = s_1 \triangleright l'_1, \ldots, s_n \triangleright l'_n, r' \triangleright t, G_3$. From $G_2 \Rightarrow^*_{\theta_2} \square$ we obtain $s_i\theta_2 \leftrightarrow^*_{\mathcal{R}} l_i\theta_2$ for $1 \leqslant i \leqslant n$ and thus $f(s_1, \ldots, s_n)\theta_2 \leftrightarrow^*_{\mathcal{R}} l\theta_2$. Likewise, $G'_2 \Rightarrow^*_{\theta'_2} \square$ implies $f(s_1, \ldots, s_n)\theta'_2 \leftrightarrow^*_{\mathcal{R}} l'\theta'_2$. From Lemma 5.5(2) we obtain $\mathcal{V}\mathrm{ar}(f(s_1, \ldots, s_n)) \subseteq \mathcal{V}\mathrm{ar}(G\theta_1)$. Hence $f(s_1, \ldots, s_n)\theta'_2 = f(s_1, \ldots, s_n)\theta_2\rho$ and therefore $l\theta_2\rho \leftrightarrow^*_{\mathcal{R}} l'\theta'_2$. Since no rewrite step in $l\theta_2\rho \leftrightarrow^*_{\mathcal{R}} l'\theta'_2$ takes place at the root position, we easily derive a contradiction with the orthogonality assumption. The case $[\alpha] = [o]_{\equiv}$ follows along the same lines. The only significant difference is an appeal to Lemma 5.5(1) instead of Lemma 5.5(2). $\square$

Theorem 5.6 essentially relies on orthogonality. Consider for example the left-linear confluent CS

$$\mathcal{R} = \left\{ \begin{array}{rcl} f(a, x) & \to & a \\ f(x, a) & \to & a \\ g(a) & \to & a \end{array} \right\}$$

and the goal $G = f(g(x), g(x)) \equiv a$. There are two $\mathrm{LNC_d}$-refutations, both computing (when restricted to the variable x) the (unique) solution $\{x \mapsto a\}$.

6. Concluding Remarks

Ida and Nakahara (1997) study a lazy narrowing calculus OINC which is similar to LNC with the leftmost selection function. The essential difference is that OINC lacks the imitation inference rule. Because of this OINC is complete only for orthogonal TRSs and goals that satisfy the restriction that the right-hand side of every equation is a ground normal form. Ida and Nakahara incorporated strict semantics into OINC, resulting in the calculus S-OINC which is very similar to our $\mathrm{LNC_d}$ in that descendants of parameter-passing equations and descendants of initial equations are distinguished. However, S-OINC is not deterministic in the choice of inference rule, neither for descendants of parameter-passing equations nor for descendants of initial equations. We have shown in this paper that determinism in the choice of inference rule for descendants of initial equations can be achieved for *arbitrary* confluent TRSs with strict semantics. Moreover, we feel that the imitation rule for descendants of initial equation in S-OINC appears out of the blue as it is not present in the calculus OINC.

González-Moreno *et al.* (1996) present the constructor-based lazy narrowing calculus CLNC. Although the underlying theory is different—a rewriting logic for modelling non-deterministic functions by non-confluent systems (CRWL) versus standard equational logic—CLNC and $\mathrm{LNC_d}$ have many features in common. The inference rules of CLNC for both descendants of parameter-passing equations (which are called approximation conditions in (González-Moreno *et al.*, 1996)) and descendants of initial equations (joinability conditions) are fully deterministic but, whereas for descendants of parameter-passing equations we always give preference to the variable elimination rule $[v]$, in CLNC conflicts between $[v]$ (OB and IB in (González-Moreno *et al.*, 1996)), the outermost narrowing rule $[o]$ (NR2) and the imitation rule $[i]$ (IIM) are resolved by giving preference to the latter two in case the variable is *demanded* (Giovannetti *et al.*, 1991). (Giving unrestricted preference to $[v]$ would result in a calculus that may compute incorrect solutions with respect to CRWL semantics; cf. Example 2 in (González-Moreno *et al.*, 1996). Giving

unrestricted preference to $[o]$ and $[i]$ would result in an incomplete calculus, as can be seen from Figure 1.) However, unlike LNC$_{\mathrm{d}}$ the calculus CLNC is *not* strongly complete (nor complete with respect to leftmost selection) because the selection of an equation in a goal may have to be postponed before an inference rule can be applied to it. Nevertheless, González-Moreno *et al.* show that the choice of (applicable) inference rules is don't care non-deterministic. (They call this latter property strong completeness.)

The question how to reduce non-determinism while preserving completeness has also been addressed for the standard definition of narrowing (cf. NC in Section 2). We mention basic (Hullot, 1980) and LSE (Bockmayr *et al.*, 1995) narrowing for terminating TRSs. Efficient refinements of standard narrowing for functional logic programs (viewed as non-terminating TRSs that satisfy additional syntactic restrictions—like orthogonality—to ensure confluence) are described in (Antoy *et al.*, 1994, 1997; Hanus, 1994a; Loogen *et al.*, 1993; Loogen and Winkler, 1995; Moreno-Navarro and Rodríguez-Artalejo, 1992). Since there is no non-determinism due to the selection of the inference rule in standard narrowing, there is no basis for a comparison with the work reported in this paper. We do however include some remarks on needed narrowing. Antoy *et al.* (1994) define and prove the completeness of needed narrowing for inductively sequential TRSs. They present two optimality results for needed narrowing. First of all, only independent[†] solutions are computed by needed narrowing. Since the class of inductively sequential TRSs coincides with the class of strongly sequential (Huet and Lévy, 1991) orthogonal CSs, Theorem 5.6 shows that strong sequentiality is not essential for obtaining the independence of computed solutions. The second optimality result presented in (Antoy *et al.*, 1994) states that needed narrowing derivations have minimal length. This result has no counterpart in LNC$_{\mathrm{d}}$.

Concerning future work, we would like to extend the results presented in this paper to conditional TRSs. This, however, is a non-trivial matter. Of the three completeness results for LNC presented in (Middeldorp *et al.*, 1996) only one so-far has been extended to the conditional case: LCNC—the conditional version of LNC—is complete whenever basic conditional narrowing is complete (Hamada and Middeldorp, 1997). At present it is unknown whether LCNC with leftmost selection is complete for arbitrary confluent conditional TRSs. (The technical reason is that Lemma 31 in (Middeldorp *et al.*, 1996) doesn't extend to conditional TRSs.) It is also unclear whether the eager variable elimination result described in (Middeldorp *et al.*, 1996) extends to LCNC. Since these two results form the foundation of the present work, the extension to conditional CTRSs is problematic indeed.

We conclude this section by comparing the performance of LNC with leftmost selection and LNC$_{\mathrm{d}}$ on a small number of examples. All numbers below are based on an implementation in SICStus Prolog 2.1#6 running on a Sun Ultra1 with 128M memory. Consider

[†] In (Antoy *et al.*, 1994) a rather peculiar definition of independence is used: substitutions θ and θ' are called independent on a set of variables V if there exists an $x \in V$ such that $x\theta$ and $x\theta'$ are not unifiable. This definition is not equivalent to the standard definition we gave in Section 1, even when $\mathcal{R} = \varnothing$: the substitutions $\{x \mapsto a\}$ and $\{y \mapsto b\}$ are independent but not according to the above definition; likewise, the substitutions ε and $\{x \mapsto c(x)\}$ are independent according to the above definition but clearly $\varepsilon \leqslant \{x \mapsto c(x)\}$. However, it seems that for substitutions computed by needed narrowing derivations starting from the same goal, the independence definition of (Antoy *et al.*, 1994) reduces to the standard definition.

Table 4. The goals $x + y \equiv \mathtt{S}^n(\mathtt{0})$.

n	LNC nodes	LNC seconds	LNC_d nodes	LNC_d seconds
0	27	0.008	8	0.002
1	197	0.054	16	0.004
2	1225	0.34	24	0.007
3	7401	2.1	32	0.011
4	44465	14	40	0.015

Table 5. The goals $x + y \equiv z$ and $\mathtt{qsort}([\mathtt{S}(\mathtt{0}),\, \mathtt{0}]) \equiv x$.

depth	LNC nodes	LNC seconds	LNC_d nodes	LNC_d seconds	LNC nodes	LNC seconds	LNC_d nodes	LNC_d seconds
1	5	0.001	3	0.000	5	0.001	3	0.000
3	23	0.005	7	0.001	12	0.003	5	0.001
5	55	0.016	11	0.003	35	0.010	7	0.002
7	131	0.035	15	0.004	107	0.032	10	0.003
9	307	0.087	19	0.006	339	0.11	13	0.003
11	635	0.18	23	0.007	1083	0.34	15	0.004
13	1499	0.42	27	0.008	2835	0.81	17	0.004
15	3019	0.87	31	0.010	8035	2.2	25	0.007
17	6987	1.97	35	0.012	26515	8.7	29	0.008
19	14571	4.4	39	0.012	80275	29	31	0.009
		$x + y \equiv z$				$\mathtt{qsort}([\mathtt{S}(\mathtt{0}),\, \mathtt{0}]) \equiv x$		

the orthogonal CS

$$
\begin{array}{lcllcl}
\mathtt{if}(\mathtt{true}, x, y) & \to & x & \mathtt{0} + y & \to & y \\
\mathtt{if}(\mathtt{false}, x, y) & \to & y & \mathtt{S}(x) + y & \to & \mathtt{S}(x + y) \\
\mathtt{append}([\,], z) & \to & z & \mathtt{0} \leqslant y & \to & \mathtt{true} \\
\mathtt{append}([x|y], z) & \to & [x|\mathtt{append}(y, z)] & \mathtt{S}(x) \leqslant \mathtt{0} & \to & \mathtt{false} \\
\mathtt{qsort}([\,]) & \to & [\,] & \mathtt{S}(x) \leqslant \mathtt{S}(y) & \to & x \leqslant y \\
\mathtt{qsort}([x|y]) & \to & \mathtt{qsort}'(x, \mathtt{split}(x, y, [\,], [\,])) & & & \\
\mathtt{qsort}'(x, \langle y, z\rangle) & \to & \mathtt{append}(\mathtt{qsort}(y), [x|\mathtt{qsort}(z)]) & & & \\
\mathtt{split}(x, [\,], y, z) & \to & \langle y, z\rangle & & & \\
\mathtt{split}(x, [y'|z'], y, z) & \to & \mathtt{if}(y' \leqslant x, \mathtt{split}(x, z', [y'|y], z), \mathtt{split}(x, z', y, [y'|z])) & & &
\end{array}
$$

First consider the goals $x + y \equiv \mathtt{S}^n(\mathtt{0})$ for $n \geqslant 0$. Table 4 shows for various values of n the size of the (finite) search tree and the runtime in seconds for both calculi. Next consider the goal $x + y \equiv z$. This goal has infinitely many incomparable (strict) solutions. The numbers in Table 5 correspond to finite approximations of the infinite search tree. Finally, consider the goal $\mathtt{qsort}([\mathtt{S}(\mathtt{0}),\, \mathtt{0}]) \equiv x$ which admits the unique solution $\theta = \{x \mapsto [\mathtt{0},\, \mathtt{S}(\mathtt{0})]\}$. It takes 0.038 seconds to compute the (finite) LNC_d search tree for this goal. The total number of nodes is 120 and the depth of the tree is 64. Table 5 shows, for various depths, the performance of LNC and LNC_d. Since every LNC_d-derivation is an LNC-derivation, LNC will also compute θ, but we haven't been able to verify whether its search tree is finite or not.

Acknowledgements

We are grateful to Koji Nakagawa for implementing LNC and LNC_d and performing the experiments described above. The detailed comments of the referees helped to improve the paper. In particular, the minimality result described in Section 5 answers a question posed by one of the referees. Aart Middeldorp is partially supported by Grant-in-Aids for Scientific Research for Encouraging Young Researchers A 09780235 and for Basic Research B 07558152 of the Ministry of Education, Science, Sports and Culture of Japan, and by the Okawa Foundation for Information and Telecommunications.

References

Antoy, S., Echahed, R., Hanus, M. (1994). A needed narrowing strategy. In *Proceedings of the 21st ACM Symposium on Principles of Programming Languages*, pages 268–279.

Antoy, S., Echahed, R., Hanus, M. (1997). Parallel evaluation strategies for functional logic languages. In *Proceedings of the 14th International Conference on Logic Programming*, pages 138–152. The MIT Press.

Bockmayr, A., Krischer, S., Werner, A. (1995). Narrowing strategies for arbitrary canonical systems. *Fundamenta Informaticae*, **24**(1,2):125–155.

Boudol, G. (1985). Computational semantics of term rewriting systems. In Nivat, M., Reynolds, J., editors, *Algebraic Methods in Semantics*, pages 169–236. Cambridge University Press.

Dershowitz, N., Jouannaud, J.-P. (1990). Rewrite systems. In *Handbook of Theoretical Computer Science*, volume B, pages 243–320. Elsevier.

Giovannetti, E., Levi, G., Moiso, C., Palamidessi, C. (1991). Kernel-leaf: A logic plus functional language. *Journal of Computer and System Sciences*, **42**(2):139–185.

González-Moreno, J., Hortalá-González, M., López-Fraguas, F., Rodríguez-Artalejo, M. (1996). A rewriting logic for declarative programming. In *Proceedings of the 6th European Symposium on Programming*, volume 1058 of *LNCS*, pages 156–172.

Hamada, M., Middeldorp, A. (1997). Strong completeness of a lazy conditional narrowing calculus. In *Proceedings of the 2nd Fuji International Workshop on Functional and Logic Programming*, pages 14–32. World Scientific.

Hanus, M. (1994a). The integration of functions into logic programming: From theory to practice. *Journal of Logic Programming*, **19 & 20**:583–628.

Hanus, M. (1994b). Lazy unification with simplification. In *Proceedings of the 5th European Symposium on Programming*, volume 788 of *LNCS*, pages 272–286.

Hölldobler, S. (1987). A unification algorithm for confluent theories. In *Proceedings of the 14th International Colloquium on Automata, Languages and Programming*, volume 267 of *LNCS*, pages 31–41.

Hölldobler, S. (1989). *Foundations of Equational Logic Programming*, volume 353 of *LNAI*. Springer Verlag.

Huet, G., Lévy, J.-J. (1991). Computations in orthogonal rewriting systems, I and II. In Lassez, J.-L., Plotkin, G., editors, *Computational Logic, Essays in Honor of Alan Robinson*, pages 396–443. The MIT Press.

Hullot, J.-M. (1980). Canonical forms and unification. In *Proceedings of the 5th Conference on Automated Deduction*, volume 87 of *LNCS*, pages 318–334.

Ida, T., Nakahara, K. (1997). Leftmost outside-in narrowing calculi. *Journal of Functional Programming*, **7**(2):129–161.

Klop, J. (1992). Term rewriting systems. In Abramsky, S., Gabbay, D., Maibaum, T., editors, *Handbook of Logic in Computer Science, Vol. 2*, pages 1–116. Oxford University Press.

Loogen, R., López-Fraguas, F., Rodríguez-Artalejo, M. (1993). A demand driven computation strategy for lazy narrowing. In *Proceedings of the 5th International Symposium on Programming Language Implementation and Logic Programming*, volume 714 of *LNCS*, pages 184–200.

Loogen, R., Winkler, S. (1995). Dynamic detection of determinism in functional logic languages. *Theoretical Computer Science*, **142**:59–87.

Martelli, A., Moiso, C., Rossi, G. (1989). Lazy unification algorithms for canonical rewrite systems. In Aït-Kaci, H., Nivat, M., editors, *Resolution of Equations in Algebraic Structures, Vol. II, Rewriting Techniques*, pages 245–274. Academic Press.

Martelli, A., Rossi, G., Moiso, C. (1986). An algorithm for unification in equational theories. In *Proceedings of the 1986 Symposium on Logic Programming*, pages 180–186.

Middeldorp, A., Hamoen, E. (1994). Completeness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computing*, **5**:213–253.

Middeldorp, A., Okui, S., Ida, T. (1996). Lazy narrowing: Strong completeness and eager variable elimination. *Theoretical Computer Science*, **167**(1,2):95–130.

Moreno-Navarro, J., Rodríguez-Artalejo, M. (1992). Logic programming with functions and predicates: The language babel. *Journal of Logic Programming*, **12**:191–223.
Snyder, W. (1991). *A Proof Theory for General Unification.* Birkhäuser.
Suzuki, T. (1996). Standardization theorem revisited. In *Proceedings of the 5th International Conference on Algebraic and Logic Programming*, volume 1139 of *LNCS*, pages 122–134.

Originally received 20 September 1996
Accepted 6 October 1997