Lazy Narrowing Calculi: Strong Completeness, Eager Variable Elimination, Nondeterminism, Optimality

Aart Middeldorp

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

Satoshi Okui

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

Abstract

Narrowing is an important method for solving unification problems in equational theories that are presented by confluent term rewriting systems. Because narrowing is a rather complicated operation, several authors studied calculi in which narrowing is replaced by more simple inference rules. In this paper we give an overview of the results that have been obtained by Middeldorp *et al.* [19, 18] for the lazy narrowing calculus LNC.

1 Introduction

Narrowing [13] is general method for solving unification problems in equational theories that are presented by confluent term rewrite systems (TRSs for short). Consider e.g. the TRS $\mathcal{R}$ consisting of the following three rewrite rules, specifying subtraction on natural numbers:

$$\begin{array}{lcl} 0 - y & \to & 0 \\ x - 0 & \to & x \\ \mathsf{s}(x) - \mathsf{s}(y) & \to & x - y \end{array}$$

Suppose we want to solve the equation $\mathsf{s}(\mathsf{s}(x-y)) \approx \mathsf{s}(x)$, i.e., we want to find a substitution for the variables x and y such that both terms become equal in the equational theory generated by $\mathcal{R}$. The following rewrite sequence, where in each step the selected redex is underlined, shows that the substitution $\theta = \{x \mapsto \mathsf{s}(0), y \mapsto \mathsf{s}(0)\}$ is a solution:

$$\begin{aligned} \mathsf{s}(\mathsf{s}(\underline{\mathsf{s}(0)-\mathsf{s}(0)})) \approx \mathsf{s}(\mathsf{s}(0)) &\rightarrow \mathsf{s}(\mathsf{s}(\underline{0-0})) \approx \mathsf{s}(\mathsf{s}(0)) \\ &\rightarrow \mathsf{s}(\mathsf{s}(0)) \approx \mathsf{s}(\mathsf{s}(0)) \end{aligned}$$

In term rewriting a subterm can be reduced only if it matches a left-hand side of a rewrite rule. In narrowing unification rather than matching is used, so variables in the term at hand may be instantiated before a rewrite step is performed. For instance, the equation $\mathsf{s}(\mathsf{s}(x-y)) \approx \mathsf{s}(x)$ cannot be reduced by rewriting but if we substitute $\mathsf{s}(x_1)$ for x and $\mathsf{s}(y_1)$ for y then the resulting equation $\mathsf{s}(\mathsf{s}(\mathsf{s}(x_1)-\mathsf{s}(y_1))) \approx \mathsf{s}(\mathsf{s}(x_1))$ can be reduced by applying the third rewrite rule to the subterm $\mathsf{s}(x_1)-\mathsf{s}(y_1)$. In order to guarantee a finitely branching search space, variables are instantiated in a minimal way such that a rewrite step can be performed. Continuing the above example, a possible narrowing computation is

$$\begin{aligned} \mathsf{s}(\mathsf{s}(\underline{x-y})) \approx \mathsf{s}(x) &\rightsquigarrow_{\theta_1} \mathsf{s}(\mathsf{s}(\underline{x_1-y_1})) \approx \mathsf{s}(\mathsf{s}(x_1)) \\ &\rightsquigarrow_{\theta_2} \mathsf{s}(\mathsf{s}(x_1)) \approx \mathsf{s}(\mathsf{s}(x_1)) \end{aligned}$$

with $\theta_1 = \{x \mapsto \mathsf{s}(x_1), y \mapsto \mathsf{s}(y_1)\}$ and $\theta_2 = \{y_1 \mapsto 0\}$. Since the two sides of the final equation are identical, the composition of θ_1 and θ_2 gives a solution to the initial equation $\mathsf{s}(\mathsf{s}(x-y)) \approx \mathsf{s}(x)$, as one easily verifies. The property that successful (i.e., ending in an equation whose sides are syntactically unifiable) narrowing derivations produce solutions is known as the *soundness* of narrowing. Narrowing is known to be *complete* for confluent TRSs with respect to normalized solutions. (A substitution is normalized if it substitutes normal forms for the variables.) That is, for every confluent TRS $\mathcal{R}$, for every equation $e = s \approx t$, and for every normalized solution θ of e, there exists a successful narrowing derivation starting from e which produces a substitution θ' that is, when restricted to the variables in e, at least as general θ. The restriction to confluent TRSs is essential as shown by the equation $e = \mathsf{b} \approx \mathsf{c}$ with respect to the non-confluent TRS $\{\mathsf{a} \rightarrow \mathsf{b}, \mathsf{a} \rightarrow \mathsf{c}\}$: the empty substitution is a solution (as b and c are equal in the induced equational theory) but narrowing is not applicable to e. The equation $e = x \approx \mathsf{f}(x)$ with respect to the one-rule TRS $\{\mathsf{a} \rightarrow \mathsf{f}(\mathsf{a})\}$ shows the necessity of the normalization requirement: the non-normalized substitution $\{x \mapsto \mathsf{a}\}$ is a solution of e but narrowing is not applicable.

Narrowing is the underlying computation mechanism of many programming language that integrate the functional and logic programming paradigms (Hanus [7]). Since narrowing is a complicated operation, numerous calculi consisting of a small number of more elementary inference rules that simulate narrowing have been proposed, e.g. by Martelli *et al.* [16, 15], Hölldobler [10, 11], Snyder [20], Dershowitz *et al.* [3], Hanus [8, 9], Ida and Nakahara [14], and Bütow *et al.* [2]. These calculi are highly nondeterministic: 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 completeness. In this paper we investigate under what conditions which of these choices can be eliminated without affecting completeness. So the aim of our work is to reduce the huge search space by minimizing the nondeterminism while retaining completeness. We do this for the *lazy narrowing calculus* (LNC for short), which is the specialization to confluent TRSs and narrowing of the Hölldobler's calculus TRANS ([11]), which is defined for general equational systems and based on paramodulation. LNC consists of the following five inference rules:

[o] *outermost narrowing*

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

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

[i] *imitation*

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

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

[d] *decomposition*

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

[v] *variable elimination*

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

if $x \notin \mathcal{V}\text{ar}(s)$ 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 $s_1 \approx l_1,\dots,s_n \approx l_n$ besides the *body* equation $r \approx t$. These parameter-passing equations must eventually be solved, but we do not require that they are solved right away. If G and G' are the upper and lower goal in the inference rule $[\alpha]$ ($\alpha \in \{\texttt{o},\texttt{i},\texttt{d},\texttt{v},\texttt{t}\}$), we write $G \Rightarrow_{[\alpha]} G'$. This is called an LNC-step. The applied rewrite rule or substitution may be supplied as subscript, i.e., we will write things like $G \Rightarrow_{[\texttt{o}], l\to r} G'$ and $G \Rightarrow_{[\texttt{i}],\theta} G'$. 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 $\square$.

In the remainder of the paper we summarize the results obtained in [18, 19]. In the next section we address the nondeterminism due to the choice of the equation in the inference rules. In Sections 3 and 4 we consider the nondeterminism due to the choice of the inference rule. We make some concluding remarks in Section 5. Due to lack of space, we omit several results and all proofs. The interested reader is referred to [18, 19] for details.

2 Strong Completeness

Consider again the TRS $\mathcal{R}$ defined in the beginning of the previous section. We have the following LNC-derivation:

$$\begin{array}{cl}
\mathsf{s}(\mathsf{s}(x-y)) \approx \mathsf{s}(x) & \\
\Downarrow_{[\mathtt{d}]} & \\
\mathsf{s}(x-y) \approx x & \\
\Downarrow_{[\mathtt{i}]} & \{x \mapsto \mathsf{s}(x_1)\} \\
\mathsf{s}(x_1) - y \approx x_1 & \\
\Downarrow_{[\mathtt{o}]} & x_2 - \mathsf{0} \to x_2 \\
\mathsf{s}(x_1) \approx x_2, y \approx \mathsf{0}, x_2 \approx x_1 & \\
\Downarrow_{[\mathtt{v}]} & \{x_2 \mapsto \mathsf{s}(x_1)\} \\
y \approx \mathsf{0}, \mathsf{s}(x_1) \approx x_1 & \\
\Downarrow_{[\mathtt{v}]} & \{y \mapsto \mathsf{0}\} \\
\mathsf{s}(x_1) \approx x_1 & \\
\Downarrow_{[\mathtt{i}]} & \{x_1 \mapsto \mathsf{s}(x_3)\} \\
\mathsf{s}(x_3) \approx x_3 &
\end{array}$$

Since the only applicable inference rule at this point is [i], it is clear that this derivation will not produce any solution. To obtain a solution we have to use a different rewrite rule in the $\Rightarrow_{\mathtt{o}}$-step:

$$\begin{array}{cl}
\mathsf{s}(x_1) - y \approx x_1 & \\
\Downarrow_{[\mathtt{o}]} & \mathsf{s}(x_2) - \mathsf{s}(y_2) \to x_2 - y_2 \\
\mathsf{s}(x_1) \approx \mathsf{s}(x_2), y \approx \mathsf{s}(y_2), x_2 - y_2 \approx x_1 & \\
\Downarrow_{[\mathtt{d}]} & \\
x_1 \approx x_2, y \approx \mathsf{s}(y_2), x_2 - y_2 \approx x_1 & \\
\Downarrow_{[\mathtt{v}]} & \{x_1 \mapsto x_2\} \\
y \approx \mathsf{s}(y_2), x_2 - y_2 \approx x_2 & \\
\Downarrow_{[\mathtt{v}]} & \{y \mapsto \mathsf{s}(y_2)\} \\
x_2 - y_2 \approx x_2 & \\
\Downarrow_{[\mathtt{o}]} & \mathsf{0} - x_3 \to \mathsf{0} \\
x_2 \approx \mathsf{0}, y_2 \approx x_3, \mathsf{0} \approx x_2 & \\
\Downarrow_{[\mathtt{v}]} & \{x_2 \mapsto \mathsf{0}\}
\end{array}$$

$$\begin{array}{ll} y_2 \approx x_3, 0 \approx 0 & \\ \quad \Downarrow_{[\mathsf{v}]} & \{y_2 \mapsto x_3\} \\ \quad 0 \approx 0 & \\ \quad \Downarrow_{[\mathsf{d}]} & \\ \quad \Box & \end{array}$$

In this refutation, which computes the solution $\{x \mapsto \mathsf{s}(0), y \mapsto \mathsf{s}(y_3)\}$, the leftmost equation in every goal is selected. We will later see that is always safe to select the leftmost equation. In other words, LNC with respect to leftmost selection is complete (for confluent TRSs and normalized solutions). One may be tempted to think that the selection of equations in goals never matters, but this is not the case.

Consider the TRS $\mathcal{R}$ consisting of the three rewrite rules

$$\begin{array}{rcl} \mathsf{f}(x) & \to & \mathsf{g}(\mathsf{h}(x), x) \\ \mathsf{g}(x, x) & \to & \mathsf{a} \\ \mathsf{b} & \to & \mathsf{h}(\mathsf{b}) \end{array}$$

and the equation $e = \mathsf{f}(\mathsf{b}) \approx \mathsf{a}$. Confluence of $\mathcal{R}$ can be proved by a routine induction argument on the structure of terms and some case analysis. The (normalized) empty substitution ϵ is a solution of e because

$$\mathsf{f}(\mathsf{b}) \to \mathsf{g}(\mathsf{h}(\mathsf{b}), \mathsf{b}) \to \mathsf{g}(\mathsf{h}(\mathsf{b}), \mathsf{h}(\mathsf{b})) \to \mathsf{a}$$

There is essentially only one LNC-derivation starting from e in which always the rightmost equation is selected:

$$\begin{array}{lll} \mathsf{f}(\mathsf{b}) \approx \mathsf{a} & \Rightarrow_{[\mathsf{o}], \mathsf{f}(x) \to \mathsf{g}(\mathsf{h}(x), x)} & \mathsf{b} \approx x, \mathsf{g}(\mathsf{h}(x), x) \approx \mathsf{a} \\ & \Rightarrow_{[\mathsf{o}], \mathsf{g}(x_1, x_1) \to \mathsf{a}} & \mathsf{b} \approx x, \mathsf{h}(x) \approx x_1, x \approx x_1, \mathsf{a} \approx \mathsf{a} \\ & \Rightarrow_{[\mathsf{d}]} & \mathsf{b} \approx x, \mathsf{h}(x) \approx x_1, x \approx x_1 \\ & \Rightarrow_{[\mathsf{v}], \{x_1 \mapsto x\}} & \mathsf{b} \approx x, \mathsf{h}(x) \approx x \\ & \Rightarrow_{[\mathsf{i}], \{x \mapsto \mathsf{h}(x_2)\}} & \mathsf{b} \approx \mathsf{h}(x_2), \mathsf{h}(x_2) \approx x_2 \\ & \Rightarrow_{[\mathsf{i}], \{x_2 \mapsto \mathsf{h}(x_3)\}} & \cdots \end{array}$$

This is clearly not a refutation. (The alternative binding $\{x \mapsto x_1\}$ in the $\Rightarrow_{[\mathsf{v}]}$-step results in a variable renaming of the above LNC-derivation.) Hence completeness of LNC is not independent of *selection functions*. In other words, LNC lacks the so-called *strong* completeness property (contradicting Hölldobler [11, Corollary 7.3.9]). In [19] it is shown that LNC is strongly complete whenever *basic* narrowing is complete. Basic narrowing (Hullot [13]) is a restriction of narrowing with the property that narrowing steps are never applied to a subterm introduced by a previous narrowing substitution. Basic narrowing has a much smaller search space than narrowing. Hence additional requirements are needed to ensure its completeness. Three such conditions are mentioned in the literature (Hullot [13], Middeldorp and Hamoen [17]): termination, right-linearity, and orthogonality (under the additional condition stated below). Hence we obtain the following strong completeness results for LNC.

Theorem 2.1 *Let $\mathcal{R}$ be a confluent TRS, $\mathcal{S}$ a selection function, and G a goal. For every normalized solution θ of G there exists an* LNC*-refutation $G \Rightarrow^*_{\theta'} \square$ respecting $\mathcal{S}$ such that $\theta' \leqslant \theta$ $[\mathcal{V}\mathrm{ar}(G)]$, provided one of the following conditions is satisfied:*

1. *$\mathcal{R}$ is terminating,*
2. *$\mathcal{R}$ is right-linear, or*
3. *$\mathcal{R}$ is orthogonal and $G\theta$ has a normal form.*

$\square$

The above counterexample against strong completeness does not refute the completeness of LNC. The goal $\mathsf{f(b)} \approx \mathsf{a}$ can be solved by always selecting the leftmost equation:

$$\begin{array}{lll}
\mathsf{f(b)} \approx \mathsf{a} & \Rightarrow_{[\mathrm{o}],\, \mathsf{f}(x) \to \mathsf{g}(\mathsf{h}(x),x)} & \mathsf{b} \approx x, \mathsf{g}(\mathsf{h}(x),x) \approx \mathsf{a} \\
& \Rightarrow_{[\mathrm{v}],\, \{x \mapsto \mathsf{b}\}} & \mathsf{g}(\mathsf{h}(\mathsf{b}),\mathsf{b}) \approx \mathsf{a} \\
& \Rightarrow_{[\mathrm{o}],\, \mathsf{g}(x_1,x_1) \to \mathsf{a}} & \mathsf{h}(\mathsf{b}) \approx x_1, \mathsf{b} \approx x_1, \mathsf{a} \approx \mathsf{a} \\
& \Rightarrow_{[\mathrm{v}],\, \{x_1 \mapsto \mathsf{h}(\mathsf{b})\}} & \mathsf{b} \approx \mathsf{h}(\mathsf{b}), \mathsf{a} \approx \mathsf{a} \\
& \Rightarrow_{[\mathrm{o}],\, \mathsf{b} \to \mathsf{h}(\mathsf{b})} & \mathsf{h}(\mathsf{b}) \approx \mathsf{h}(\mathsf{b}), \mathsf{a} \approx \mathsf{a} \\
& \Rightarrow_{[\mathrm{d}]} & \mathsf{b} \approx \mathsf{b}, \mathsf{a} \approx \mathsf{a} \\
& \Rightarrow_{[\mathrm{d}]} & \mathsf{a} \approx \mathsf{a} \\
& \Rightarrow_{[\mathrm{d}]} & \square
\end{array}$$

This is not a coincidence, because we have the following completeness theorem ([19]).

Theorem 2.2 *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 $\mathcal{S}_{\mathrm{left}}$ such that $\theta' \leqslant \theta$ $[\mathcal{V}\mathrm{ar}(G)]$.* $\square$

Here $\mathcal{S}_{\mathrm{left}}$ is the selection function that always selects the leftmost equation. So the nondeterminism of LNC due to the selection of the equation is avoided if we adopt the leftmost equation, which we do from now on. Hence in the remainder of the paper we assume that the sequence of equations G' to the left of the selected equation in the inference rules of LNC is empty.

$$\begin{tikzcd}
\mathsf{f(b)} \approx \mathsf{a} \arrow[d, Rightarrow, "{[\mathsf{o}]}"] & & & \\
\mathsf{b} \approx \mathsf{g}(x), \mathsf{a} \approx \mathsf{a} \arrow[d, Rightarrow, "{[\mathsf{o}]}"] & & & \\
\mathsf{g(b)} \approx \mathsf{g}(x), \mathsf{a} \approx \mathsf{a} \arrow[d, Rightarrow, "{[\mathsf{d}]}"] & & & \\
\mathsf{b} \approx x, \mathsf{a} \approx \mathsf{a} \arrow[r, Rightarrow, "{[\mathsf{o}]}"] \arrow[d, Rightarrow, "{[\mathsf{v}]}"] & \mathsf{g(b)} \approx x, \mathsf{a} \approx \mathsf{a} \arrow[r, Rightarrow, "{[\mathsf{i}]}"] \arrow[d, Rightarrow, "{[\mathsf{v}]}"] & \mathsf{b} \approx x_1, \mathsf{a} \approx \mathsf{a} \arrow[r, Rightarrow, "{[\mathsf{o}]}"] \arrow[d, Rightarrow, "{[\mathsf{v}]}"] & \cdots \\
\mathsf{a} \approx \mathsf{a} \arrow[d, Rightarrow, "{[\mathsf{d}]}"] & \mathsf{a} \approx \mathsf{a} \arrow[d, Rightarrow, "{[\mathsf{d}]}"] & \mathsf{a} \approx \mathsf{a} \arrow[d, Rightarrow, "{[\mathsf{d}]}"] & \\
\square & \square & \square &
\end{tikzcd}$$

Figure 1: The LNC-refutations starting from $\mathsf{f(b)} \approx \mathsf{a}$.

3 Eager Variable Elimination

Next we turn out attention to the nondeterminism of LNC due to the selection of the inference rule. The nondeterministic application of the various inference rules to selected (leftmost) equations causes LNC to generate many redundant derivations. Consider for example the TRS consisting of the two rewrite rules

$$\begin{aligned} \mathsf{f}(\mathsf{g}(x)) &\to \mathsf{a} \\ \mathsf{b} &\to \mathsf{g(b)} \end{aligned}$$

Figure 1 shows all LNC-refutations starting from the goal $\mathsf{f(b)} \approx \mathsf{a}$. There are infinitely many such refutations, all computing the empty substitution. Hence for completeness it is suffices to compute only one of them. 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, although to the best of our knowledge there is no supporting proof of this so-called *eager variable elimination problem* for the general case of confluent systems.

In [19, 18] it is shown that a restricted version of the eager variable elimination strategy is complete for left-linear confluent 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. For example, in Figure 1 the equation $\mathsf{b} \approx x$ descends from the parameter-passing equation $\mathsf{b} \approx \mathsf{g}(x)$ introduced in the first $\Rightarrow_{[\mathsf{o}]}$-step.

An equation of the form $x \simeq t$, with $x \notin \mathcal{V}\mathrm{ar}(t)$, 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.

Theorem 3.1 *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 $G \Rightarrow^*_{\theta'} \square$ such that $\theta' \leqslant \theta$ $[\mathcal{V}\mathrm{ar}(G)]$.* $\square$

4 Other Nondeterminism

In this section we address the remaining nondeterminism between the inference rules of LNC. First we consider descendants of parameter-passing equations. Consider the TRS $\mathcal{R}$:

$$\begin{aligned} \mathsf{f}(\mathsf{a}) &\to \mathsf{f}(\mathsf{b}) \\ \mathsf{g}(\mathsf{f}(\mathsf{b})) &\to \mathsf{c} \end{aligned}$$

and the goal $\mathsf{g}(\mathsf{f}(x)) \approx \mathsf{c}$. Only the outermost narrowing rule [o] is applicable to this goal, resulting in the new goal $\mathsf{f}(x) \approx \mathsf{f}(\mathsf{b}), \mathsf{c} \approx \mathsf{c}$. To the parameter-passing equation $\mathsf{f}(x) \approx \mathsf{f}(\mathsf{b})$ 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 $\{x \mapsto \mathsf{b}\}$ and in the latter the solution $\{x \mapsto \mathsf{a}\}$. Since these solutions are incomparable (with respect to subsumption modulo $\mathcal{R}$), we cannot eliminate the nondeterminism between the outermost narrowing rule [o] and the decomposition rule [d] while retaining completeness.

The above situation cannot occur when we are dealing with left-linear *constructor systems.* 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. 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 $l = f(l_1, \ldots, l_n)$, 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. Observe that the above $\mathcal{R}$ is not a CS because the defined symbol f occurs in the argument of the left-hand side $\mathsf{g}(\mathsf{f}(\mathsf{b}))$.

In [18] it is shown that for left-linear confluent CSs all nondeterminism between the inferences rules of LNC can be eliminated for descendants of parameter-passing equations. The proof is a consequence of the following lemma.

Table 1: Selection of inference rule for descendant $s \approx t$ of parameter-passing equation.

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

Lemma 4.1 *Let $\mathcal{R}$ be a left-linear CS and $G \Rightarrow^* G', s \approx t, G''$ an* LNC*-derivation. If the equation $s \approx t$ descends from a parameter-passing equation then $\mathcal{V}\text{ar}(G', s) \cap \mathcal{V}\text{ar}(t) = \varnothing$ and t is a constructor term.* □

The first part of Lemma 4.1 implies in particular that for every descendant $s \approx t$ of a parameter-passing equation, s and t have no variables in common. 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 3.1 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 4 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 $\text{root}(t) \in \mathcal{F}_\mathcal{D}$ is impossible according to the second part of Lemma 4.1. 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 the first part of Lemma 4.1, so is the variable elimination rule [v] and by Theorem 3.1 the latter is given precedence.

Next we turn our attention to descendants of equations in the initial goal. The following example shows that the restriction to left-linear confluent CSs is insufficient to remove all non-determinism in the choice of inference rule for descendants of initial equations. Consider the (left-linear confluent) CS

$$\mathsf{f(a)} \rightarrow \mathsf{f(b)}$$

and the goal $\mathsf{f}(x) \approx \mathsf{f(b)}$. This goal has the two incomparable solutions $\{x \mapsto \mathsf{a}\}$ and $\{x \mapsto \mathsf{b}\}$. 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

$$\begin{aligned} \mathsf{f(a)} &\rightarrow \mathsf{g(a)} \\ \mathsf{g(b)} &\rightarrow \mathsf{f(b)} \end{aligned}$$

and the goal $\mathsf{f}(x) \approx \mathsf{g}(x)$. The solution $\{x \mapsto \mathsf{a}\}$ can only be obtained if we apply the

Table 2: Selection of inference rule for descendant $s \approx t$ of initial equation.

root(s)\root(t)	$\mathcal{V}$	$\mathcal{F}_C$	$\mathcal{F}_D$
$\mathcal{V}$	[v]/[t]	[v]/[i][a]	[o]
$\mathcal{F}_C$	[v]/[i][b]	[d]	[o]
$\mathcal{F}_D$	[o]	[o]	[o][c]

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

outermost narrowing rule [o] to the left-hand side of $\mathsf{f}(x) \approx \mathsf{g}(x)$, but in order to obtain the incomparable solution $\{x \mapsto \mathsf{b}\}$ 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 [4, 1]. If we interpret $\approx$ as strict equality then the non-determinism in the above examples disappears: neither $\{x \mapsto \mathsf{a}\}$ nor $\{x \mapsto \mathsf{b}\}$ are (strict) solutions of the goals $\mathsf{f}(x) \approx \mathsf{f}(\mathsf{b})$ and $\mathsf{f}(x) \approx \mathsf{g}(x)$. A substitution θ is said to be a *strict solution* of a goal G if for every equation $s \approx t$ in G there exists a constructor term u such that $s\theta \rightarrow^* u$ and $t\theta \rightarrow^* u$.

Note that we do not 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. In [18] it is shown that for confluent TRSs all nondeterminism between the inference rules of LNC can be eliminated for descendants of initial equations with strict semantics. We would like to stress that the restriction to left-linear CSs is not needed here. Table 2 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.

5 Concluding Remarks

The results of the preceding two section can be directly incorporated into the inference rules of LNC. This gives rise to the *deterministic* lazy narrowing calculus, LNC$_\mathrm{d}$ for short, whose inference rules can be found in [18]. In LNC$_\mathrm{d}$ there is no nondeterminism between the inference rules. In other words, at most one inference rule is applicable to every goal.

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

In [18] it is further shown that substitutions computed by different LNC$_d$ derivations are *incomparable*, provided we are dealing with *orthogonal* CSs. Hence for this subclass of TRSs solutions to goals are computed only once by LNC$_d$.

A similar result has been obtained for *needed narrowing*. Antoy *et al.* [1] define and prove the completeness of needed narrowing for inductively sequential TRSs. They present two optimality results for needed narrowing. First of all, only incomparable solutions are computed by needed narrowing. Since the class of inductively sequential TRSs coincides with the class of strongly sequential ([12]) orthogonal CSs, our result shows that strong sequentiality is not essential for obtaining the incomparability of computed solutions. The second optimality result presented in [1] states that needed narrowing derivations have minimal length. This result has no counterpart in LNC$_d$.

We refer to [18] for a more thorough discussion of related work. We conclude this paper by mentioning that some of the results presented above have been extended to the more complicated setting of conditional term rewriting, see [5] and [6] for details.

References

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

[2] B. Bütow, R. Giegerich, E. Ohlebusch, and S. Thesing. Semantic matching for left-linear convergent rewrite systems. *Journal of Functional and Logic Programming*, 1999. To appear.

[3] N. Dershowitz, S. Mitra, and G. Sivakumar. Decidable matching for convergent systems. In *Proceedings of the 11th International Conference on Automated Deduction*, volume 607 of *LNAI*, pages 589–602, 1992.

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

[5] M. Hamada and A. Middeldorp. 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, 1997.

[6] M. Hamada, A. Middeldorp, and T. Suzuki. Completeness results for a lazy conditional narrowing calculus. In *Combinatorics, Computation and Logic: Proceedings of 2nd Discrete Mathematics and Theoretical Computer Science Conference and the 5th Australasian Theory Symposium*, pages 217–231. Springer-Verlag Singapore, 1999.

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

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

[9] M. Hanus. Lazy narrowing with simplification. *Computer Languages*, 23(2–4):61–85, 1997.

[10] S. Hölldobler. 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, 1987.

[11] S. Hölldobler. *Foundations of Equational Logic Programming*, volume 353 of *LNAI*. Springer Verlag, 1989.

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

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

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

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

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

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

[18] A. Middeldorp and S. Okui. A deterministic lazy narrowing calculus. *Journal of Symbolic Computation*, 25(6):733–757, 1998.

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

[20] W. Snyder. *A Proof Theory for General Unification*. Birkhäuser, 1991.

Solutions Week 2

2. No, the relation $\subset$ is not irreflexive. Consider for instance the infinite sequence $\mathbf{t} = \mathsf{a}$, $\mathsf{f}(\mathsf{a})$, $\mathsf{f}(\mathsf{f}(\mathsf{a}))$, ... Since t_i is a proper subterm of t_{i+1} for all $i \geqslant 1$, we have $\mathbf{t} \subset \mathbf{t}$ by definition.

3. Consider a signature $\mathcal{F}$ consisting of a constant 0, a unary function symbol s, and an n-ary function symbol c. We code natural numbers as terms in $\mathcal{T}(\{0, \mathsf{s}\})$ via the following mapping ϕ:

$$\phi(n) = \begin{cases} 0 & \text{if } n = 0, \\ \mathsf{s}(\phi(n-1)) & \text{if } n > 0. \end{cases}$$

This mapping is extended to n-tuples of natural numbers by defining

$$\phi((x_1, \ldots, x_n)) = \mathsf{c}(\phi(x_1), \ldots, \phi(x_n)).$$

Now consider an infinite sequence $\mathbf{e}$ of n-tuples of natural numbers. By construction $\phi(\mathbf{e})$ is an infinite sequence of terms in $\mathcal{T}(\{0, \mathsf{s}, \mathsf{c}\})$. According to Kruskal's Tree Theorem there exist $i < j$ such that $\phi(e_i) \trianglelefteq_{\mathsf{emb}} \phi(e_j)$. It is not difficult to see that the latter is equivalent to $e_i \leqslant_n e_j$.

4. Because not every infinite sequence of arbitrary terms is self-embedding. The simplest counterexample enumerates the countably infinite set of variables: x_1, x_2, x_3, ...

5. Consider e.g. the TRS $\mathcal{R} = \{\mathsf{f}(\mathsf{f}(x)) \to \mathsf{g}(x), \mathsf{g}(x) \to \mathsf{f}(x)\}$. Its termination cannot be shown by the lexicographic path order since the first rule requires $\mathsf{f} \succ \mathsf{g}$ and the second rule requires $\mathsf{g} \succ \mathsf{f}$. Since all function symbols are unary, there is no difference between the lexicographic path order and any other order in the family of recursive path orders. The polynomial interpretations $\mathsf{f}_{\mathbb{N}}(x) = x + 2$ and $\mathsf{g}_{\mathbb{N}}(x) = x + 3$ for all $x \in \mathbb{N}$ show that $\mathcal{R}$ is polynomially terminating.

6. (a) Let $\mathcal{R}$ be a looping TRS. So $s \to_{\mathcal{R}}^{+} t$ for some terms s and t such that t encompasses s, i.e., $t = C[s\sigma]$ for some context C and substitution σ. Since $\to_{\mathcal{R}}^{+}$ is a rewrite relation, we obtain the infinite rewrite sequence $s \to_{\mathcal{R}}^{+} C[s\sigma] \to_{\mathcal{R}}^{+} C[C[s\sigma]\sigma] \to_{\mathcal{R}}^{+} C[C[C[s\sigma]\sigma]\sigma] \to_{\mathcal{R}}^{+} \cdots$.

   (b) Consider the TRS

$$\mathcal{R} = \begin{cases} \mathsf{f}(0, y) & \to \quad \mathsf{f}(\mathsf{s}(y), 0) \\ \mathsf{f}(\mathsf{s}(x), y) & \to \quad \mathsf{f}(x, \mathsf{s}(y)) \end{cases}$$

   We show that $\mathcal{R}$ is not terminating. Define for every $i \geqslant 0$ the term $t_i = \mathsf{f}(0, \mathsf{s}^i(0))$. We have $t_i \to^{+} t_{i+1}$ by one application of the first rewrite rule followed by $i + 1$ applications of the second rewrite rule. Hence $\mathcal{R}$ admits the infinite rewrite sequence $t_0 \to^{+} t_1 \to^{+} t_2 \to^{+} \cdots$. The proof that $\mathcal{R}$ is non-looping will be given next week.