Leftmost Outside-In Conditional Narrowing for Functional-logic Programming Languages

Tetsuo Ida [†] [‡], Koichi Nakahara [††] and Makoto Hamana [‡‡]

March, 1994

ISE-TR-94-108

Abstract

We present a new method of conditional narrowing called LOI (Leftmost Outside-In) conditional narrowing for orthogonal conditional term rewriting systems with strict equality. We show that LOI conditional narrowing is complete. Furthermore we present a calculus that realizes LOI conditional narrowing. The calculus shows that LOI conditional narrowing can be realized by several inference rules that perform basic operations of narrowing. Those inference rules are easy to implement. The calculus has been used to implement a new functional-logic programming language based on applicative conditional term rewriting systems with extra variables. The language has a feature of higher-order programming.

[†]Inst. of Information Sciences and Electronics, Univ. of Tsukuba, Tsukuba, Ibaraki 305, Japan

[‡]ida@is.tsukuba.ac.jp

[††]koh@softlab.is.tsukuba.ac.jp

[‡‡]hamana@softlab.is.tsukuba.ac.jp

1 Introduction

Narrowing has become an important computing mechanism for functional-logic programming languages. It comprises reduction of functional programming and term unification of logic programming. A conditional term rewriting system with narrowing is a natural computation model for functional-logic programming languages. To design efficient conditional narrowing that enjoys the property of completeness is, therefore, a research not only of theoretical interest but also of practical importance. In this paper we propose leftmost outside-in (LOI in short) conditional narrowing. The conditional LOI narrowing is defined for orthogonal conditional term rewriting systems. Orthogonal systems are appropriate models for functional-logic programming languages since most of proposed functional-logic programming languages can be viewed as a syntactically sugared version of orthogonal conditional term rewriting systems.

Various methods of narrowing have been proposed and their completeness has been studied. Among them innermost narrowing[4], outer narrowing[17], LSE narrowing [2] and basic narrowing [9, 13, 16] have been well studied. These methods are originally presented for (unconditional) term rewriting systems. In a separate paper we proposed LOI narrowing for (unconditional) term rewriting systems and showed its completeness[10]. LOI narrowing is based on the notion of leftmost outside-in reduction that was presented by Huet and Lévy[8]. In this paper we extend LOI narrowing for conditional term rewriting systems and show its application to the design and implementation of a functional-logic programming language based on applicative term rewriting.

The organization of the paper is as follows. In Section 2 we summarize basic notations that are used in this paper. We assume readers' familiarity with the basics of term rewriting systems. We introduce conditional term rewriting systems in Section 3, and relate them by example to functional-logic programs. In Section 4 we formally define conditional narrowing. In Section 5 we introduce conditional narrowing and discuss the correspondence between narrowing and reduction derivations. In Section 6 we define LOI conditional narrowing, and in Section 7 we introduce a calculus **LNC** that performs LOI narrowing. In Section 8 we give a new functional-logic language based on applicative rewriting systems. Programs of this language are evaluated by LOI narrowing. The completeness of the evaluation is guaranteed by the completeness result given in Section 7.

2 Preliminaries

Let $\mathcal{F}$ be a set of function symbols, and $\mathcal{V}$ a set of variables, satisfying $\mathcal{F} \cap \mathcal{V} = \emptyset$. Terms are defined as usual over a set of alphabet $\mathcal{F} \cup \mathcal{V}$. The set of terms is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$, or simply by $\mathcal{T}$. A set $\mathcal{F}$ is divided into disjoint sets $\mathcal{F}_\mathcal{C}$ and $\mathcal{F}_\mathcal{D}$; $\mathcal{F}_\mathcal{C}$ is a set of constructors and $\mathcal{F}_\mathcal{D}$ is a set of defined function symbols. When there is no danger of confusion we call a constructor symbol simply a constructor and a defined function symbol simply a function symbol. A term whose root symbol is a constructor is called a constructor term, and a term in $\mathcal{T}(\mathcal{F}_\mathcal{C}, \mathcal{V})$ is called a data term.

$\mathcal{V}(\mathcal{A})$ denotes a set of variables occurring in a syntactic object $\mathcal{A}$. $\mathcal{O}(t)$ denotes a set of

positions of a term t. A subterm of t at position u is denoted by $t|_u$. A position u in $\mathcal{O}(t)$ is called a non-variable position if $t|_u$ is not a variable. The set of non-variable positions of t is denoted by $\overline{\mathcal{O}}(t)$. A term obtained from t by replacing $t|_u$, where $u \in \mathcal{O}(t)$, by a term s is denoted by $t[s]_u$. An equation $s = t$, where $s = t \in \mathcal{T}$, is a special term whose root symbol is $=$ (used as an infix operator, and allowed only at the root position).

A substitution is a mapping from $\mathcal{V}$ to $\mathcal{T}$. The domain of a substitution θ is defined as $\mathcal{D}\theta = \{x \mid \theta x \not\equiv x, x \in \mathcal{V}\}$, and the codomain of θ as $Cod\ \theta = \{\theta x \mid x \in \mathcal{D}\theta\}$. We identify a substitution θ with the set $\{x \mapsto \theta x \mid x \in \mathcal{D}\theta\}$. An empty substitution is defined as the empty set $\emptyset$. A substitution is extended to an endomorphism over $\mathcal{T}$ as usual. Let $V \subseteq \mathcal{V}$. The restriction of θ to V is denoted by $\theta\lceil_V$. We write $\theta_1 = \theta_2[V]$ when $\theta_1\lceil_V = \theta_2\lceil_V$ holds. The composition of θ_2 and θ_1, (first apply θ_1, then θ_2) is denoted by $\theta_2\theta_1$. When $\sigma\theta_1 = \theta_2$ for some substitution σ, we write $\theta_1 \preceq \theta_2$. When $\sigma\theta_1 = \theta_2[V]$ holds for some substitution σ, we write $\theta_1 \preceq \theta_2[V]$. A set of substitutions is denoted by Θ.

3 Functional-logic program and conditional term rewriting system

A conditional term rewriting system (CTRS in short, hereafter) is a set of rewrite rules $\{l_i \to r_i \Leftarrow Q_i \mid i \in I\}$, where $l_i, r_i \in \mathcal{T}$ and Q_i is a sequences of equations. Q_i is called a condition of a rewrite rule $R_i \triangleq l_i \to r_i \Leftarrow Q_i$. A term $s \equiv t$ is called a strict equation. An infix function symbol $\equiv$ defines a strict equality, i.e., $s \equiv t$ is true iff s and t are reduced to the same ground data term. A CTRS $\mathcal{R}$ is called an s-CTRS if all the conditions of the rewrite rules in $\mathcal{R}$ are sequences of strict equations. We denote by $\mathcal{R}_{\equiv}$ the CTRS $\mathcal{R}$ extended with rewrite rules for strict equations.

Most proposed functional-logic programming languages[5, 7, 12, 14] are modeled as a CTRS. For example, the following program R, which we use throughout this paper, is a functional-logic program that is regarded as a CTRS $\mathcal{R}$.

Example 3.1 Appending two lists if they consists of natural numbers.

$$\mathtt{R} = \left\{ \begin{array}{l} \mathtt{append([\,],ys)} \to \mathtt{ys} \Leftarrow \mathtt{nats(ys)} \equiv \mathtt{tt}, \\ \mathtt{append(x:xs,ys)} \to \mathtt{x:append(xs,ys)} \Leftarrow \mathtt{nat(x)} \equiv \mathtt{tt}, \\ \mathtt{nat(0)} \to \mathtt{tt}, \\ \mathtt{nat(s(x))} \to \mathtt{nat(x)}, \\ \mathtt{nats([\,])} \to \mathtt{tt}, \\ \mathtt{nats(x:xs)} \to \mathtt{nats(xs)} \Leftarrow \mathtt{nat(x)} \equiv \mathtt{tt}. \end{array} \right.$$

where

- x, xs and ys are variables, and
- [ ] is an empty list and : is an infix operator of cons.

The above functional-logic program is regarded also as an s-CTRS.

A CTRS $\mathcal{R} \triangleq \{l_i \to r_i \Leftarrow Q_i \mid i \in I\}$ is called orthogonal if $\mathcal{U}(\mathcal{R}) \triangleq \{l_i \to r_i \mid i \in I\}$ is orthogonal. An orthogonal CTRS is abbreviated as OCTRS, hereafter. If $\mathcal{R}$ is orthogonal,

$\mathcal{R}_{\equiv}$ is also made orthogonal as we will see in the next section. A CTRS $\mathcal{R}$ is called 1-CTRS if $\mathcal{V}(l_i) \supseteq \mathcal{V}(r_i) \cup \mathcal{V}(Q_i)$ for every $i \in I$, and 2-CTRS if $\mathcal{V}(l_i) \supseteq \mathcal{V}(r_i)$ for every $i \in I$[13].

In the case of 2-CTRS, let $\mathrm{Ext}(R_i) \triangleq \mathcal{V}(Q_i) - \mathcal{V}(l_i)$. A variable in $\mathrm{Ext}(R_i)$ is called an extra variable. A 1s-CTRS is 1-CTRS and s-CTRS, and likewise for 2s-CTRS. The program R in Example 3.1 defines a 1s-OCTRS.

4 Conditional narrowing

Given a functional-logic program R the evaluator of functional-logic programs solves a sequence of strict equations, called goal, by conditional narrowing. The goal is conditionally narrowed repeatedly until it eventually becomes a sequence of true's. Take a goal `append([s(0)],w) ≡ y:[0]` for example, assuming that a program R of Example 3.1 is given. Let $\mathcal{R}_{\equiv} = \mathcal{R} \cup \mathcal{R}_c$ where

$$\mathcal{R}_c = \left\{ \begin{array}{l} [\,] \equiv [\,] \rightarrow \mathrm{true}, \\ (u_1 : u_2) \equiv (v_1 : v_2) \rightarrow \mathrm{true} \Leftarrow u_1 \equiv v_1,\ u_2 \equiv v_2, \\ \mathrm{tt} \equiv \mathrm{tt} \rightarrow \mathrm{true}. \end{array} \right.$$

The goal is conditionally narrowed for the $\mathcal{R}_{\equiv}$, and a solution $\{w \mapsto [0], y \mapsto s(0)\}$ is obtained.

For theoretical treatment we define a goal either an empty sequence, denoted by $\Box$, or a sequence consisting of strict equations and true's. We then define conditional narrowing as follows. Let $G[e := S]$ denote a goal G with the strict equation e in G replaced by a goal S.

Definition 4.1 Let $\mathcal{R}$ be a CTRS. The single-step conditional narrowing $\rightsquigarrow$ over goals is defined as follows. Suppose G and G' are goals. $G \rightsquigarrow G'$ if there exist a strict equation e in G, a position $u \in \overline{\mathcal{O}}(e)$, a new variant $l \rightarrow r \Leftarrow Q$ of a rewrite rule in $\mathcal{R}_{\equiv}$, and a substitution σ such that

- $\sigma e|_u \equiv \sigma l$,
- $G' \equiv \sigma G[e := Q, e[r]_u]$.

We may also write $G \rightsquigarrow_\theta G'$, where θ is the substitution which is formed in the single-step narrowing[1] and whose domain is restricted to $\mathcal{V}(G)$. A multiple-step narrowing is written $G \rightsquigarrow\!\!\!\!\rightsquigarrow_\theta G'$. The substitution θ is a composition of substitutions formed in the $\rightsquigarrow\!\!\!\!\rightsquigarrow$ steps, and whose domain is restricted to $\mathcal{V}(G)$. A symbol $\top$ generically represents a sequence of zero or more true's.

The evaluation process of a goal G_0 can be described by a narrowing derivation

$$G_0 \rightsquigarrow_{\theta_1} G_1 \rightsquigarrow \cdots \rightsquigarrow_{\theta_n} G_n.$$

The narrowing derivation ending with $\top$ is said to be successful. The goal G_0 is solvable if there exists a successful narrowing derivation starting from G_0. A successful narrowing

[1]Hereafter, we omit the word 'conditional' in conditional narrowing since narrowing is always conditional. When we want to emphasize 'conditional', we write conditional narrowing explicitly, however.

derivation starting from G_0 gives a solution $\theta \triangleq (\theta_n \cdots \theta_1)\restriction_{\mathcal{V}(G_0)}$ of the goal G_0, and we write $G_0 \rightsquigarrow\!\!\!\!\rightarrow_\theta \top$.

The process of solving the goal $\mathrm{append}([s(0)], w) \equiv y : [0]$ is describe by the following narrowing derivation.

$$
\begin{array}{ll}
 & \mathrm{append}([s(0)], w) \equiv y : [0] \\
\rightsquigarrow_{\{w \mapsto ys_1\}} & \mathrm{nat}(s(0)) = \mathrm{tt}, s(0) : \mathrm{append}([s(0)], w) \equiv y : [0] \\
\rightsquigarrow_{\emptyset} & \mathrm{true}, s(0) : \mathrm{append}([], ys) \equiv y : [0] \\
\rightsquigarrow_{\{ys_1 \mapsto ys_2\}} & \mathrm{nat}(ys_2) \equiv \mathrm{tt}, s(0) : ys_2 \equiv y : [0] \\
\rightsquigarrow_{\{ys_2 \mapsto x_2 : xs_2\}} & \mathrm{true}, \mathrm{nat}(x_2) \equiv \mathrm{tt}, \mathrm{nat}(xs_2) \equiv \mathrm{tt}, s(0) : (x_2 : xs_2) \equiv y : [0] \\
\rightsquigarrow_{\{x_2 \mapsto 0, xs_2 \mapsto [\,]\}} & \mathrm{true}, \mathrm{true}, \mathrm{true}, s(0) : (0 : [\,]) \equiv y : [0] \\
\rightsquigarrow_{\{y \mapsto s(0)\}} & \top
\end{array}
$$

Hence, we obtain a solution $\{w \mapsto [0], y \mapsto s(0)\}$ as expected.

The use of strict equations is advocated by several researchers in functional-logic programming[5, 14]. We also adopt strict equations in the condition of rewrite rules and goals for the following reasons.

- We need strict equality in actual programming. We want to compare two terms to see if they are reduced to the same data term.
- The strict equations together with orthogonality condition of CTRSs guarantee the data term solutions, i.e, the solution whose codomain is a set of data terms.

In order to make conditional narrowing practical for the computing mechanism of functional-logic programs, we need to design a specific narrowing where we can locate narrowable subterms efficiently. We are interested in the narrowing that performs lazy narrowing, and still enjoys the completeness of narrowing with respect to data term solutions. A (specific) narrowing is said to be complete for a certain class of a solution σ if for a given goal G, every substitution σ in that class such that $\sigma G \twoheadrightarrow \top$ can be found by that narrowing. The leftmost outside-in conditional narrowing is the one that meets our requirements. To define the leftmost outside-in conditional narrowing, we first discuss the correspondence between narrowing and reduction derivations.

5 Intermediate reduction

Intuitively, narrowing is a combination of two operations; first instantiate a term to be narrowed by a most general substitution and then reduce the term by a rewrite rule whose left-hand side matches with the term. It is clear from this view that narrowing and reduction derivations are made to correspond each other by a suitable substitution. The correspondence is used to analyze the properties of narrowing derivations.

5.1 Correspondence between narrowing and reduction derivations

Let $\mathcal{R} \vdash G$ denote the statement that the reduction $G \twoheadrightarrow_{\mathcal{R}} \top$ holds. The reduction relation induced by a CTRS $\mathcal{R}$ is defined via the notion of level reduction.

$$\mathcal{R}_0 = \emptyset$$

$$\mathcal{R}_{n+1} = \{\sigma l \to \sigma r \mid l \to r \Leftarrow Q \in \mathcal{R}, \sigma \in \Theta, \text{such that } \mathcal{R}_n \vdash \sigma Q\}$$

These TRSs $\mathcal{R}_n$ induce reduction relation $\to_{\mathcal{R}}$ Relation $\to_{\mathcal{R}}$ is defined as $\cup_n \to_{\mathcal{R}_n}$. We call the reduction $s \to_{\mathcal{R}} t$ n-level if $s \to_{\mathcal{R}_n} t$. A CTRS $\mathcal{R}$ is level-confluent if each $\mathcal{R}_n$ is confluent.

The reduction derivation with respect to the reduction relation $\to_{\mathcal{R}}$ does not correspond to narrowing derivation since rewriting of the goals originating in the conditions of the rewrite rules used during the reduction derivation is not recorded in the derivation.

In order to make the reduction derivation correspond to the narrowing derivation, we need a notion of intermediate reduction. The notion is originally due to Bockmayr[3].[2]

Definition 5.1 Let $\mathcal{R}$ be an s-CTRS. The single-step intermediate reduction $\rightarrowtail$ over goals is defined as follows. Suppose G and G' are goals. $G \rightarrowtail G'$ if there exist a strict equation e in S, a position $u \in \mathcal{O}(e)$, a new variant $l \to r \Leftarrow Q$ of a rewrite rule in $\mathcal{R}_{\equiv}$, and a substitution σ such that

- $e|_u \equiv \sigma l$,
- $G' \equiv G[e := \sigma Q, e[\sigma r]_u]$,
- $\mathcal{R}_{\equiv} \vdash \sigma Q$.

Intermediate reduction is abbreviated as i-reduction hereafter.

We can now have a correspondence between the narrowing derivation

$$G_0 \rightsquigarrow_{\theta_1} G_1 \rightsquigarrow \cdots \rightsquigarrow G_{n-1} \rightsquigarrow_{\theta_n} G_n$$

and the i-reduction derivation

$$\sigma_0 G_0 \rightarrowtail \sigma_1 G_1 \twoheadrightarrow \sigma_{n-1} G_{n-1} \rightarrowtail G_n$$

where $\sigma_i = \theta_n \cdots \theta_{i+1}$ for $i = 0, \ldots, n-1$,

in which the same rewrite rules are employed at the same positions of corresponding goals in each step of the derivations.

[2] Bockmayr called the intermediate reduction *Reduktionssrelation ohne Auswertung der Prämisse* (reduction relation without evaluating conditions).

5.2 Lifting of derivations

From an i-reduction derivation $\theta G \to G'$ we can obtain a corresponding narrowing $G \rightsquigarrow_\sigma G'$ derivation by lifting lemmas. The lifting lemmas which we give in the following are crucial to the proof of completeness. The normalization conditions on the substitution θ in the lemmas below are essential to make the i-reduction derivation correspond to the narrowing derivation.

In the case of 2-CTRSs, a problematic situation occurs when extra variables get instantiated with a term that is level-normalized at the time of instantiation, but it becomes reducible when later i-reductions are performed at a higher level. We have to exclude such possibilities. The sufficient normalization condition and the notion of the restricted i-reduction are provided for that purpose. Refer to [13] for further details.

Lemma 5.1 (Lifting lemma for 1-CTRSs [3]) Let $\mathcal{R}$ be a 1-CTRS. Suppose we have goals S and T, a normalized substitution θ such that $\mathcal{D}\theta \subseteq \mathcal{V}(S)$ and $T \triangleq \theta S$. For an i-reduction derivation $T \twoheadrightarrow T'$ there exist a goal S', substitutions θ' and σ such that

- $S \rightsquigarrow_\sigma S'$,
- $\theta' S' \equiv T'$,
- $\theta'\sigma = \theta[\mathcal{V}(S)]$,
- θ' is a normalized substitution.

The narrowing derivation $S \rightsquigarrow_\sigma S'$ and the i-reduction derivation $T \twoheadrightarrow T'$ employ the same rewrite rules at the same positions in the corresponding goals.

The lifting lemma for 2-CTRSs is more complicated because of the presence of extra variable. We define a restricted i-reduction $\overset{n}{\twoheadrightarrow}$ over solvable goals as follows.

Definition 5.2 Let $\mathcal{R}$ be an arbitrary s-CTRS, and S and T be solvable goals.

1. $\overset{0}{\twoheadrightarrow} = \emptyset$,
2. $S \overset{n+1}{\twoheadrightarrow} T$ if there exist an equation e in S, a position $u \in \mathcal{O}(e)$, a new variant $\mathcal{R} \triangleq l \to r \Leftarrow Q$ of a rewrite rule in $\mathcal{R}$, and a substitution σ such that
   - $e|_u \equiv \sigma l$,
   - $T \equiv S[e := \sigma Q, e[\sigma r]_u]$
   - $R_n \vdash \sigma Q$,
   - $\sigma\restriction_{\mathrm{Ext}(R)}$ is $\to_{\mathcal{R}_n}$-normalized,
   - $n+1$ does not exceed the level of e.

$\twoheadrightarrow$ is defined as $\cup_{n \geq 0} \overset{n}{\twoheadrightarrow}$. The notion of the restricted i-reduction is due to Middeldorp and Hamoen[13]. A restricted i-reduction derivation can be lifted to a narrowing derivation, as shown below.

Definition 5.3 A solution σ of a goal G is called sufficiently normalized (normalizable) if $\sigma\restriction_{\mathcal{V}(e)}$ is $\mathcal{R}_n$-normalized ($\mathcal{R}_n$-normalizable) where n is the level of σe, for every equation e in G.

Lemma 5.2 (Lifting lemma for 2-CTRSs[13]) Let $\mathcal{R}$ be a level-confluent 2-CTRS. Suppose we have goals S and T, a sufficiently normalized solution θ such that $\mathcal{D}\theta \subseteq \mathcal{V}(S)$ and $T \triangleq \theta S$. For a restricted i-reduction derivation $T \twoheadrightarrow\!\!\!\!\circ\, T'$ there exist a goal S', substitutions θ' and σ such that

- $S \rightsquigarrow_\sigma S'$,
- $\theta' S' \equiv T'$,
- $\theta'\sigma = \theta[\mathcal{V}(S)]$,
- θ' is a sufficiently normalized solution of S'.

The narrowing derivation $S \rightsquigarrow_\sigma S'$ and the i-reduction derivation $T \twoheadrightarrow\!\!\!\!\circ\, T'$ employ the same rewrite rules at the same positions in the corresponding goals.

6 Leftmost outside-in conditional narrowing

Having established the correspondence between conditional narrowing derivations and i-reduction derivations, we are ready to define a leftmost outside-in narrowing derivation via a leftmost outside-in reduction derivation. First we observe the following.

From an i-reduction derivation of a goal we can extract a sequence of (strict) equations that form a reduction derivation.

Example 6.1 We use Example 3.1. We have the following i-reduction derivation starting from a goal $\mathrm{nat}([0]) \equiv \mathrm{tt}$.

$$\mathrm{nats}([0]) \equiv \mathrm{tt} \twoheadrightarrow\!\!\!\circ\, \mathrm{nat}(0) \equiv \mathrm{tt}, \mathrm{nats}([\,]) \equiv \mathrm{tt} \twoheadrightarrow\!\!\!\circ\, \mathrm{tt} \equiv \mathrm{tt}, \mathrm{nats}([\,]) \equiv \mathrm{tt} \twoheadrightarrow\!\!\!\circ$$

$$\mathrm{true}, \mathrm{nats}([\,]) \equiv \mathrm{tt} \twoheadrightarrow\!\!\!\circ\, \mathrm{true}, \mathrm{tt} \equiv \mathrm{tt}, \twoheadrightarrow\!\!\!\circ\, \mathrm{true}, \mathrm{true}.$$

From the above i-reduction derivation, we can extract the following two reduction derivations.

- $\mathrm{nats}([0]) \equiv \mathrm{tt} \to_{\{1\}} \mathrm{nats}([\,]) \equiv \mathrm{tt}, \to_\emptyset \mathrm{nats}([\,]) \equiv \mathrm{tt} \to_\emptyset \mathrm{nats}([\,]) \equiv \mathrm{tt}$ $\to_{\{1\}} \mathrm{tt} \equiv \mathrm{tt} \to_\emptyset \mathrm{true}$,
- $\mathrm{nat}(0) \equiv \mathrm{tt} \to_{\{1\}} \mathrm{tt} \equiv \mathrm{tt} \to_\emptyset \mathrm{true} \to_\emptyset \mathrm{true} \to_\emptyset \mathrm{true} \to_\emptyset \mathrm{true}$,

where $s \to_{U(\triangleq\{u_1,\ldots,u_n\})} t$ denotes (multiple-step) reduction from s to t by contracting redexes at the pairwise disjoint positions $u_1, \ldots, u_n, n \geq 0$, using rewrite rules in $\mathcal{U}(\mathcal{R})$. Note that we consider OCTRSs, and hence the term t is uniquely determined.

We call these reduction derivations the traces of the i-reduction derivation. Since the i-reduction derivation of $\mathcal{R}$ consists of the reduction derivation of $\mathcal{U}(\mathcal{R})$, the notions developed for orthogonal TRSs are applicable to the i-reduction derivation. In particular, Huet and Lévy's notion of a leftmost outside-in reduction derivation for orthogonal TRSs is applicable to the i-reduction derivation. Hence, we have the following definition of leftmost outside-in i-reduction and narrowing derivations.

Definition 6.1

- An i-reduction derivation is leftmost outside-in (LOI in short) if each trace of the i-reduction derivation is LOI.
- A narrowing derivation is LOI if the corresponding i-reduction derivation is LOI.

Example 6.2 An example of LOI narrowing derivation is the following.

$$\begin{cases} \text{ones} \to \text{s}(0) : \text{ones}, \\ \text{hd}(x : xs) \to x \Leftarrow \text{nat}(x) \equiv \text{tt}. \end{cases}$$

$$\text{hd}(\text{ones}) \equiv w \rightsquigarrow_{\emptyset} \text{hd}(\text{s}(0) : \text{ones}) \equiv w \rightsquigarrow_{\emptyset} \text{s}(0) \equiv w \rightsquigarrow_{\{w \mapsto \text{s}(0)\}} \text{true}.$$

By the application of the standardization theorem of Huet and Lévy [8], we have the following lemma.

Lemma 6.1 Let $\mathcal{R}$ be an s-OCTRS and G be a goal. If there exists a successful i-reduction derivation $G \twoheadrightarrow\!\!\!\!\!\rightarrow \top$, then there exists an LOI i-reduction derivation.

Now the completeness result for 1s-CTRS is straightforward to obtain.

Theorem 6.1 LOI conditional narrowing is complete with respect to normalizable solutions for 1s-OCTRSs.

Proof: Let $\mathcal{R}$ be a 1s-OCTRS. Suppose we have a normalizable solution σ of a goal G, i.e., $\sigma G \twoheadrightarrow_{\mathcal{R}} \top$. By the confluence of $\to_{\mathcal{R}}$, $\hat{\sigma} G \twoheadrightarrow_{\mathcal{R}} \top$, where $\hat{\sigma}$ is a normalized substitution obtained by reducing all the terms in $Cod\,\sigma$. By Lemma 6.1 there exists an LOI i-reduction derivation $\hat{\sigma} G \twoheadrightarrow\!\!\!\!\!\rightarrow_{\mathcal{R}} \top$. By the lifting lemma 5.1 for 1s-CTRS there exists an LOI narrowing derivation $G \rightsquigarrow\!\!\!\!\!\rightarrow_{\tau} \top$ such that $\tau \preceq \hat{\sigma}[\mathcal{V}(G)]$. Hence $\tau \preceq_{\mathcal{R}} \sigma[\mathcal{V}(G)]$. ∎

The completeness proof of LOI conditional narrowing for 2s-OCTRSs is more involved.

Definition 6.2 Let $\mathcal{R}$ be an arbitrary CTRS, and $\mathcal{S}_n$ be associated TRSs that are inductively defined as follows:

$$\begin{array}{lcll} \mathcal{S}_0 & = & \mathcal{R}_0, & \\ \mathcal{S}_{n+1} & = & \{(\sigma l, \sigma r) & \mid \; R : l \to r \Leftarrow Q \in \mathcal{R},\ \mathcal{S}_n \vdash \sigma Q \\ & & & \quad \text{and } \sigma\restriction_{\text{Ext}(R)} \text{ is } \to_{\mathcal{S}_n}\text{-normalized}, \sigma \in \Theta\}. \end{array}$$

As in $\to_{\mathcal{R}}$, we define the reduction relation $\to_{\mathcal{S}}$ associated with TRSs S_n as $\cup_n \to_{\mathcal{S}_n}$.

A CTRS $\mathcal{R}$ is called level-normal if $\to_{\mathcal{R}_n} = \to_{\mathcal{S}_n}$ for all $n \geq 0$. In level-normal 2-CTRSs, it can be easily shown that G $\twoheadrightarrow_{\mathcal{R}}$ $\top$ iff there exists a restricted i-reduction G $\twoheadrightarrow\!\!\!\!\!\rightarrow_{\mathcal{R}}$ $\top$.

Theorem 6.2 [13] Conditional narrowing is complete with respect to sufficiently normalizable solutions for level-confluent and level-normal 2s-CTRSs.

Since 2s-OCTRSs are level-confluent [6, 1] and level-normal [11], we obtain the completeness result for 2s-OCTRSs.

Corollary 6.1 Conditional narrowing is complete with respect to sufficiently normalizable solutions for 2s-OCTRSs.

Finally, we have the following completeness result for 2s-OCTRSs. The proof is similar to the proof of Theorem 6.1.

Theorem 6.3 LOI conditional narrowing is complete with respect to sufficiently normalizable solutions for orthogonal 2s-CTRSs.

Corollary 6.1 guarantees that we can obtain all data term solutions by conditional narrowing.

7 Calculus LNC

In this section we consider the realization of LOI narrowing. In a separate paper [10] we proposed a calculus that realizes LOI narrowing for orthogonal TRSs. That calculus is extended straightforwardly to handle conditional narrowing. Note that all rewritings in conditional narrowing and i-reduction derivations are performed actually using $\mathcal{U}(\mathcal{R}_{\equiv})$. Hence except that new goals in the condition of rewrite rules are added to each goal being solved, basic computing mechanism for narrowing remains unchanged even in the conditional case.

In the following calculus, which we call **LNC**, narrowing is decomposed into more primitive operations. We give **LNC** as an inference system that operates on goals. Since each step is finer than single-step narrowing, intermediate forms of goals will appear in each inference steps. In the following calculus, a goal is a sequence of equations (not restricted to strict equations).

Definition 7.1 Let $\mathcal{R}$ be an s-OCTRS. A calculus **LNC** for $\mathcal{R}$ is a pair $(\mathcal{G},\mathcal{I})$, where

- $\mathcal{G}$ is a set of goals,
- $\mathcal{I}$ is a set of inference rules defined as follows.

  - [on] outermost narrowing

    $$\frac{f(s_1,\ldots,s_n)=t,S}{s_1=l_1,\ldots,s_n=l_n,Q,r=t,S} \qquad t\notin\mathcal{V}$$

    if there exists a new variant $f(l_1,\ldots,l_n)\rightarrow r\Leftarrow Q$ of a rewrite rule in $\mathcal{R}$.
  - [d] decomposition

    $$\frac{f(s_1,\ldots,s_n)=f(t_1,\ldots,t_n),S}{s_1=t_1,\ldots,s_n=t_n,S}$$
  - [v] variable elimination
    * [v1]

      $$\frac{t=x,S}{\sigma S,\text{ where }\sigma=\{x\mapsto t\}}$$

* [v2]

$$\frac{x = t, S}{\sigma S, \text{where } \sigma = \{x \mapsto t\}} \qquad t \notin \mathcal{V}$$

– [ons] outermost narrowing for strict equations

$$\frac{f(s_1, \ldots, s_n) \equiv t, S}{s_1 = l_1, \ldots, s_n = l_n, Q, r \equiv t, S} \quad \text{or} \quad \frac{s \equiv f(t_1, \ldots, t_n), S}{t_1 = l_1, \ldots, t_n = l_n, Q, s \equiv r, S}$$

if there exist a new variant $f(l_1, \ldots, l_n) \to r \Leftarrow Q$ of a rewrite rule in $\mathcal{R}$.

– [ds] decomposition for strict equations

$$\frac{c(s_1, \ldots, s_n) \equiv c(t_1, \ldots, t_n), S}{s_1 \equiv t_1, \ldots, s_n \equiv t_n, S}$$

where $c \in \mathcal{F}_C$.

– [ims] imitation for strict equations

$$\frac{c(s_1, \ldots, s_n) \equiv y, S}{\theta(s_1 \equiv y_1, \ldots, s_n \equiv y_n, S)} \quad \text{or} \quad \frac{y \equiv c(t_1, \ldots, t_n), S}{\theta(y_1 \equiv t_1, \ldots, y_n \equiv t_n, S)}$$

where $c \in \mathcal{F}_C$ and $\theta = \{y \mapsto c(y_1, \ldots, y_n)\}$.

– [ts] elimination of trivial strict equations

$$\frac{x \equiv y, S}{\sigma S}$$

where $\sigma = \begin{cases} \{x \mapsto y\} & \text{if } x \not\equiv y \\ \emptyset & \text{otherwise.} \end{cases}$

Note:

- There exists indeterminacy between the choice of [on] and [d], and between the choice of two rules of [ons].
- We do not need an inference rule

$$\frac{t = f(s_1, \ldots, s_n), S}{s_1 = l_1, \ldots, s_n = l_n, Q, t = r, S}$$

since a narrowable term is never generated on the right-hand side of the equations of a goal.

- It is easy to see that for a goal $s_1 = t_1, \ldots, s_i = t_i, \ldots, s_n = t_n$, [3]

$$\mathcal{V}(\{s_1, t_1, \ldots, s_{i-1}, t_{i-1}, s_i\}) \cap \mathcal{V}(t_i) = \emptyset.$$

Hence, we have $x \notin \mathcal{V}(t)$ in [v1] and [v2]. The so-called occur check, i.e., the check of $x \notin \mathcal{V}(t)$, is unnecessary in applying the inference rules [v1] and [v2].

[3] Consider strict equation $p \equiv q$ as $p \equiv q = \text{true}$.

An equation of the form $t = x$ is always processed by the inference rule [v1]. In other words, narrowing on the term t is not performed if the right-hand side of the equation is a variable. In our calculus, this is the meaning of lazy narrowing. In the implementation of **LNC**, substitutions are maintained separately, and terms are essentially represented in directed acyclic graphs (dag). The inference rule [v1] coupled with the dag representation of terms is also regarded as the essence of lazy narrowing.

In LOI narrowing, arguments of a function term is narrowed first if necessary to the extent that the function term as a whole becomes narrowable. This is realized by the inference rules [on] and [ons]. The inference rule [ims] is used to narrow the subterms of a constructor term.

We have two kinds of equations, i.e., ordinary equations and strict equations. Initially a goal consisting of strict equations is given. By the inference rule [ons], ordinary equations are generated from strict equations. Ordinary equations are used to equate actual parameters of a function term and formal parameters of a rewrite rule.

The set $\mathcal{I}$ is a union of $\mathcal{B} \triangleq \{[\text{on}], [\text{d}], [\text{v}]\}$ of the basic inference rules and $\mathcal{S} \triangleq \{[\text{ons}], [\text{ds}], [\text{ims}], [\text{ts}]\}$ of the inference rule for strict equations. To guarantee the completeness, $\mathcal{B}$ is sufficient if we use $\mathcal{R}_{\equiv}$. $\mathcal{S}$ is provided for efficiency sake, and in practice necessary to make the evaluator of the functional-logic programs viable.

We will see this in the following example.

Example 7.1

$$
\begin{array}{ll}
 & \text{append}([s(0)], w) \equiv y : [0] \\
\overset{\text{on}}{\leadsto} & \text{append}([s(0)], w) \equiv u_1 : u_2, y : [0] = v_1 : v_2, u_1 \equiv v_1, u_2 \equiv v_2, \text{true} = \text{true} \\
\overset{\text{on}}{\leadsto} & [s(0)] = x_1 : xs_1, w = ys, \text{nat}(x_1) = \text{tt}, x : \text{append}(xs_1, xs_2) = u_1 : u_2, \\
 & y : [0] = v_1 : v_2, u_1 \equiv v_1, u_2 \equiv v_2, \text{true} = \text{true} \\
\overset{\mathcal{B}}{\leadsto} & s(0) : \text{append}([], w) = u_1 : u_2, y : [0] = v_1 : v_2, u_1 \equiv v_1, u_2 \equiv v_2, \text{true} = \text{true} \\
\overset{\mathcal{B}}{\leadsto} & s(0) \equiv y, \text{append}([], w) \equiv [0], \text{true} = \text{true} \\
\overset{\mathcal{B}}{\leadsto} & \square \\
 & \{y \mapsto s(0), w \mapsto [0]\}
\end{array}
$$

$$
\begin{array}{ll}
 & \text{append}([s(0)], w) \equiv y : [0] \\
\overset{\text{ons}}{\leadsto} & [s(0)] = x : xs_1, w = ys_1, \text{nat}(x_1) \equiv \text{tt}, x_1 : \text{append}(xs_1, ys_1) \equiv y : [0] \\
\overset{\mathcal{I}}{\leadsto} & s(0) : \text{append}([], w) \equiv y : [0] \\
\overset{\mathcal{I}}{\leadsto} & \square \\
 & \{y \mapsto s(0), w \mapsto [0]\}
\end{array}
$$

We showed in [10] that **LNC** is a complete realization of LOI narrowing for orthogonal TRSs. The proof given in [10] can be extended to the LOI conditional narrowing.

The completeness theorem for **LNC** is obtained from the following proposition.

Proposition 7.1 Let $\mathcal{R}$ be an s-OCTRS and G be a goal consisting of strict equations. If there exists a narrowing derivation $G \overset{*}{\leadsto}_{\theta} \top$, then there exists an LNC-derivation $G \overset{LNC}{\leadsto}_{\sigma} \square$ such that $\sigma \preceq \theta$.

Theorem 7.1 **LNC** is complete with respect to sufficiently normalizable solutions for 2s-OCTRSs.

Example 7.2 We use Example 6.2. The following example shows that **LNC** performs the lazy evaluation.

$$
\begin{array}{ll}
 & \mathrm{hd}(\mathrm{ones}) \equiv w \\
\overset{\mathrm{ons}}{\rightsquigarrow} & \mathrm{ones} = x : xs,\ \mathrm{nat}(x) \equiv \mathrm{tt},\ x \equiv w \\
\overset{\mathrm{on}}{\rightsquigarrow} & 1 : \mathrm{ones} = x : xs,\ \mathrm{nat}(x) \equiv \mathrm{tt},\ x \equiv w \\
\overset{\mathrm{d}}{\rightsquigarrow} & 1 = x, \mathrm{ones} = xs,\ \mathrm{nat}(x) \equiv \mathrm{tt},\ x \equiv w \\
\overset{\mathrm{v}_1}{\rightsquigarrow}_{\{x \mapsto 1\}} & \mathrm{ones} = xs,\ \mathrm{nat}(1) \equiv \mathrm{tt},\ 1 \equiv w \\
\overset{\mathrm{v}_1}{\rightsquigarrow}_{\{xs \mapsto \mathrm{ones}\}} & \mathrm{nat}(1) \equiv \mathrm{tt},\ 1 \equiv w \\
\overset{\mathrm{ons}}{\rightsquigarrow} & \mathrm{s}(0) = \mathrm{s}(x),\ \mathrm{nat}(x) \equiv \mathrm{tt},\ \mathrm{s}(0) \equiv w \\
\overset{\mathrm{d}}{\rightsquigarrow} & 0 = x,\ \mathrm{nat}(x) \equiv \mathrm{tt},\ \mathrm{s}(0) \equiv w \\
\overset{\mathrm{v}}{\rightsquigarrow}_{\{x \mapsto 0\}} & \mathrm{nat}(0) \equiv \mathrm{tt},\ \mathrm{s}(0) \equiv w \\
\overset{\mathrm{ons}}{\rightsquigarrow} & 0 = 0,\ \mathrm{tt} \equiv \mathrm{tt},\ \mathrm{s}(0) \equiv w \\
\overset{\mathrm{d}}{\rightsquigarrow} & \mathrm{tt} \equiv \mathrm{tt},\ \mathrm{s}(0) \equiv w \\
\overset{\mathrm{ds}}{\rightsquigarrow} & \mathrm{s}(0) \equiv w \\
\overset{\mathrm{ims}}{\rightsquigarrow}_{\{w \mapsto \mathrm{s}(v)\}} & 0 \equiv v \\
\overset{\mathrm{ims}}{\rightsquigarrow}_{\{v \mapsto 0\}} & \square
\end{array}
$$

8 Applicative CTRS

Based on the completeness result of LOI conditional narrowing, we have designed and implemented a functional-logic programming language called Ev[15]. The language is based on an applicative 2s-OCTRS. Note that applicative 2s-OCTRSs are also 2s-OCTRSs. An applicative TRS (and CTRS) is a rewrite system that has only one defined function symbol Ap. In the applicative rewrite systems, terms are either variables or constants or applicative terms Ap(t, s), where t and s are terms. For example, Ap(Ap(map, f), x)) is an applicative term. Assuming terms are associated to left, we can omit Ap and unnecessary parentheses, as in λ-terms of the lambda calculus. For example we write the above applicative term as map f x. The rewrite rules in the applicative rewrite system that correspond to the first two rewrite rules in Example 3.1 are written as follows.

$$\mathtt{append}\ [\,]\ \mathtt{ys} \to \mathtt{y} \Leftarrow \mathtt{nats}\ \mathtt{ys} \equiv \mathtt{tt},$$

$$\mathtt{append}\ (\mathtt{x} : \mathtt{xs})\ \mathtt{ys} \to \mathtt{x} : \mathtt{append}\ \mathtt{xs}\ \mathtt{ys} \Leftarrow \mathtt{nat}\ \mathtt{x} \equiv \mathtt{tt}.$$

In the applicative CTRS we can write a rewrite rule that will take a function (regarded as constant) as an argument. Thus we can write a program dealing with higher-order functions, which one would expect from functional programming. For example, the map function is written as

$$\begin{cases} \mathtt{map}\ \mathtt{f}\ [\,] \to [\,], \\ \mathtt{map}\ \mathtt{f}\ (\mathtt{x} : \mathtt{xs}) \to \mathtt{f}\ \mathtt{x} : \mathtt{map}\ \mathtt{f}\ \mathtt{xs}. \end{cases}$$

It is easy to adapt **LNC** to deal with applicative CTRSs. In the applicative systems, we can give a goal whose solution contains higher-order terms. For example, we can obtain a solution $\{f \mapsto s\}$ for a goal map $f\ [0] \equiv [s\ 0]$. When the program is large, it is in practise infeasible to obtain a function as a solution. Therefore in Ev, we prohibit by type-checking the use of function-typed variables in conditions and goals.

9 Concluding remarks

We have presented LOI conditional narrowing for s-OCTRSs. LOI conditional narrowing enjoys the completeness. Furthermore we have shown the calculus that realizes LOI conditional narrowing. The calculus shows that LOI conditional narrowing can be realized by several inference rules that perform basic operations of narrowing. Those inference rules are easy to implement. As an application, the calculus has been used to implement a new functional-logic programming language based on applicative 2s-OCTRSs.

References

[1] J. A. Bergstra and J. W. Klop. Conditional rewrite rules: Confluence and termination. *Journal of Computer and System Sciences*, 32(3):323–362, 1986.

[2] A. Bockmayr, S. Krischer, and A. Werner. Narrowing strategies for arbitrary canonical rewrite systems. Technical report, Universität Karlsruhe, 1993.

[3] A. M. Bockmayr. *Beiträge zur Theorie des logisch-funktionalen Programmierens.* PhD thesis, Universität Karlsruhe, 1990.

[4] L. Friborg. SLOG: a logic programming language interpreter based on clausal superposition and rewriting. In *Proceddings of the 2nd IEEE Symposium on Logic Programming, Boston*, pages 172–184, 1985.

[5] 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.

[6] E. Giovannetti and C. Moiso. A completeness result for E-unification algorithms based on conditional narrowing. In *Proceedings of the Workshop on Foundations of Logic and Functional Programming, Lecture Notes in Computer Science 306*, pages 157–167, 1986.

[7] Michael Hanus. Efficient Implementation of Narrowing and Rewriting. In *Proc. Int. Workshop on Processing Declarative Knowledge*, 1991. LNAI 567.

[8] G. Huet and J. Lévy. Computations in orthogonal rewriting systems, I. In J.-L. Lassez and G. Plotkin, editors, *Computational logic: essays in honor of Alan Robinson*, pages 395–414. The MIT Press, 1991.

[9] J. Hullot. Canonical forms and unification. In *Proceedings of the 5th Conference on Automated Deduction, Lecture Notes in Computer Science 87*, pages 318–334, 1980.

[10] T. Ida and K. Nakahara. Leftmost outside-in narrowing calculi. 1994. submitted for publication.

[11] T. Ida and S. Okui. Outside-in conditional narrowing. *IEICE Transactions on Information and Systems*, 1994. to appear.

[12] H. C.R. Lock. *The Implementation of Functional Logic Programming Languages.* PhD thesis, Universität Karlsruhe, 1992. Published as GMD-Bericht Nr. 208, by R. Oldenbourg Verlag, 1993.

[13] A. Middeldorp and E. Hamoen. Completeness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computing*, 1994. To appear.

[14] J. J. Moreno-Navarro and M. Rodríguez-Artalejo. Logic programming with functions and predicates: The language BABEL. *Journal of Logic Programming*, 12:191–223, 1992.

[15] Tomoyuki Nishioka. A functional-logic language based on combinatory term rewriting system. In *10th Conference Proceedings Japan Society for Software Science and Technology*, pages 333–336, 1993. in Japanese.

[16] W. Nutt, P. Réty, and G. Smolka. Basic narrowing revisited. *Journal of Symbolic Computation*, 7:295–317, 1989.

[17] J.-H. You. Enumerating outer narrowing derivations for constructor-based term rewriting systems. *Journal of Symbolic Computation*, 7:319–341, 1989.