J. Functional Programming **7** (2): 129–161, March 1997. Printed in the United Kingdom

*Leftmost outside-in narrowing calculi**

TETSUO IDA
Institute of Information Sciences and Electronics and
Center for Tsukuba Advanced Research Alliance
University of Tsukuba, Tsukuba 305, Japan

KOICHI NAKAHARA†
Canon Inc., Shimomaruko, Ohta-ku, Tokyo 146, Japan

Abstract

We present narrowing calculi that are computation models of functional-logic programming languages. The narrowing calculi are based on the notion of the leftmost outside-in reduction of Huet and Lévy. We note the correspondence between the narrowing and reduction derivations, and define the leftmost outside-in narrowing derivation. We then give a narrowing calculus OINC that generates the leftmost outside-in narrowing derivations. It consists of several inference rules that perform the leftmost outside-in narrowing. We prove the completeness of OINC using an ordering defined over a narrowing derivation space. To use the calculus OINC as a model of computation of functional-logic programming, we extend OINC to incorporate strict equality. The extension results in a new narrowing calculus, s-OINC. We show also that s-OINC enjoys the same completeness property as OINC.

Capsule Review

Consider a term rewriting system T with reduction relation R. Solving an equation in R means that, for given terms $t(x, y), s(x, y)$ of T, one tries to solve the equation $t(x, y) = s(x, y)$, i.e. to find terms X, Y of T such that

$$t(X, Y) =_R s(X, Y).$$

Solving such problems is useful for implementing functional-logic programming languages. The method of narrowing consists of substituting for x, y terms $u(\vec{z}), v(\vec{z})$, respectively, obtaining the equation $t(u(\vec{z}), v(\vec{z})) = s(u(\vec{z}), v(\vec{z}))$ in such a way that, for example, $t(u(\vec{z}), v(\vec{z}))$ contains an R-redex; then one contracts this redex, obtaining $t'(u(\vec{z}), v(\vec{z}))$, and one tries to solve for $\vec{z}$ in the equation

$$t'(u(\vec{z}), v(\vec{z})) =_R s(u(\vec{z}), v(\vec{z})).$$

Solutions found in this way will also be solutions for the original equation, but in general, one only finds a subset of the solutions; hence the name 'narrowing'. If one particular way of narrowing finds all solutions, then it is called complete.

It is clear that this method is related to making reduction paths. The paper treats a method, related to the leftmost outside-in reduction strategy of Huet and Lévy, in which 'narrowing

* A version extended with detailed proofs of some of the propositions in this paper is available from the authors' web site, `http://www.score.is.tsukuba.ac.jp/~ida`.
† Part of this work was carried out while the second author was on the University of Tsukuba Doctoral Program of Engineering.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

paths' are systematically produced. It is proved that this method is complete for finding all solutions relevant for functional programming. A second method introduced in the paper is also complete for finding solutions relevant for functional-logic programming.

1 Introduction

Recently, narrowing has received much research interest in the declarative programming community as it was found to be an important computing mechanism of functional-logic programming languages (Antoy *et al.*, 1994; Friborg, 1985; Giovannetti *et al.*, 1991; Hanus, 1990; Lock, 1992; Moreno-Navarro and Rodríguez-Artalejo, 1992; Reddy, 1985). In this paper we present narrowing calculi that are used for implementing functional-logic programming languages. Our narrowing calculi are based on the notion of the outside-in reduction of Huet and Lévy (1991) for orthogonal term rewriting systems. Huet and Lévy showed that, for a given reduction derivation of a term s to its normal form (if it exists), there exists a leftmost outside-in reduction derivation from s to its normal form. This derivation is also called 'standard' due to this property.

The practical implications of the standard derivation lie in the following. First, for a sub-class of orthogonal term rewriting systems, called strongly sequential systems, there exists a sequential strategy (i.e. an effective means to locate a redex that should be reduced next without look-ahead) by which we generate a standard reduction derivation. This strategy is often called a 'call-by-need' strategy, since it selects a redex only when its contraction is definitely needed in each reduction. Secondly, it provides a theoretical basis of the lazy evaluation in the framework of (first-order) functional programming.

By the correspondence of reduction and narrowing, in particular by the use of a so-called lifting lemma, we can obtain a narrowing derivation that corresponds to the standard reduction derivation. This narrowing derivation, which we call a leftmost outside-in narrowing derivation, deserves special investigation, as the leftmost outside-in reduction derivation does in reduction. Let a method of narrowing that generates the leftmost outside-in narrowing derivation be called leftmost outside-in narrowing. The leftmost outside-in narrowing behaves very much like the leftmost outside-in reduction. It narrows the subterms at the same positions that are contracted by the reduction using the same rewrite rules. It performs 'lazy narrowing'. Furthermore, to process narrowable expressions in an outside-in manner is amenable to the implementation of narrowing.

There exists an important difference between reduction and narrowing derivations, however. For a given standard reduction derivation its lifted narrowing derivation is not necessarily that in which only 'needed' narrowable terms are contracted. This phenomenon was observed by several researchers, and lead them to discover new methods of narrowing. You (1989) presented an outer narrowing for constructor-based orthogonal systems and Antoy *et al.* (1994) presented a needed narrowing strategy for strongly sequential constructor-based systems. Darlington and Guo

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

(1989) noted the similarity between reduction and narrowing derivations, and developed a narrowing algorithm for constructor-based orthogonal term rewriting systems. Their algorithm is essentially the same as our leftmost outside-in narrowing restricted to constructor-based systems.

Because of Huet and Lévy's intriguing definition of the leftmost outside-in reduction derivation (and narrowing derivation thereof), it is not easy to perceive the computational characteristics of the leftmost outside-in narrowing. We present the leftmost outside-in narrowing as a computation of calculi consisting of several inference rules that together perform the leftmost outside-in narrowing. The calculi show clearly fundamental computation steps of the leftmost outside-in narrowing, and lead us to their implementations on computers.

Our narrowing calculi are defined for arbitrary orthogonal term rewriting systems. This has a practical implication in the design of a functional-logic programming language, since there is an important class of orthogonal term rewriting systems that is not constructor-based; for example, many applicative term rewriting systems, when used as defining functional (and functional-logic) programs, are orthogonal and non-constructor-based.[1]

In addition to the efficiency and ease of implementation, another central issue in designing a narrowing calculus is to secure the property of completeness. Roughly speaking, the completeness of a narrowing calculus is the ability, as an equation solver, to find all the solutions of a given equation. Unrestricted narrowing is known to be complete for terminating confluent term rewriting systems (Hullot, 1980). Besides the needed narrowing and outer narrowing, several restricted narrowing methods have been studied, and the completeness results have been obtained for certain classes of term rewriting systems and with respect to certain classes of solutions (Bockmayr *et al.*, 1992; Echahed, 1988; Hullot, 1980; Middeldorp and Hamoen, 1994). We prove that our narrowing calculi are complete for orthogonal term rewriting systems with respect to a certain (and in practice sufficiently large) class of solutions.

This paper is organized as follows. In section 2 we introduce our notations. In section 3 we explain narrowing as a computation of an inference system, **NC** (Narrowing Calculus). From **NC** we develop another narrowing calculus that performs the leftmost outside-in narrowing. In section 4 we give an inductive definition of the leftmost outside-in narrowing derivation and relate the notion of the leftmost outside-in narrowing derivation to the standard reduction derivation of Huet and Lévy. In section 5 we present a calculus called **OINC** that generates a leftmost outside-in derivation. The **OINC** calculus enjoys soundness and completeness. In section 5.2 we prove the completeness of **OINC**, and in section 6 we incorporate strict equality into **OINC**. We show that this extended calculus, to be called **s-OINC**, is a natural and efficient model of computation for functional-logic programming, and further, that **s-OINC** is complete.

[1] See, for example, the program examples in Huet and Lévy (1991).

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

2 Preliminaries

Let $\mathscr{F}$ be a set of function symbols, and $\mathscr{V}$ a set of variables, satisfying $\mathscr{F} \cap \mathscr{V} = \emptyset$. Terms are defined as usual over the alphabet $\mathscr{F} \cup \mathscr{V}$. The set of terms is denoted by $\mathscr{T}(\mathscr{F}, \mathscr{V})$, or simply by $\mathscr{T}$. A term t is called linear when no variable occurs in t more than once. The set $\mathscr{F}$ is divided into disjoint sets $\mathscr{F}_{\mathscr{C}}$ and $\mathscr{F}_{\mathscr{D}}$; $\mathscr{F}_{\mathscr{C}}$ is a set of constructor symbols and $\mathscr{F}_{\mathscr{D}}$ is a set of defined function symbols. A term in $\mathscr{T}(\mathscr{F}_{\mathscr{C}}, \mathscr{V})$ is called a constructor term[2], and a term in $\mathscr{T}(\mathscr{F}, \mathscr{V})$ whose root symbol is a constructor symbol is called a head-constructor term. We assume that a distinguished symbol 'true' is in the set $\mathscr{T}(\mathscr{F}_{\mathscr{C}}, \emptyset)$

$\mathscr{V}(\mathscr{A})$ denotes a set of variables occurring in a syntactic object $\mathscr{A}$. The set $\mathscr{O}(t)$ of positions of a term t is a set of sequences of positive integers that address subterms of t. An empty sequence is denoted by ε. A position u in $\mathscr{O}(t)$ addresses a subterm $t_{|u}$, where $t_{|u}$ is defined as follows. Let $t \triangleq f(t_1, \ldots, t_n)$.[3] Then, $t_{|u} \triangleq t$ if $u = \varepsilon$ and $t_{|u} \triangleq t_{i|_{u'}}$ if $u = i.u'$. A subterm $t_{|u}$ is called proper if $u \neq \varepsilon$. A position u in $\mathscr{O}(t)$ is called a non-variable position if $t_{|u}$ is not a variable. The set of non-variable positions of t is denoted by $\overline{\mathscr{O}}(t)$. A term obtained from t by replacing $t_{|u}$, where $u \in \mathscr{O}(t)$, by a term s is denoted by $t[s]_u$. An equation $s = t$, where $s = t \in \mathscr{T}$, is a special term whose root symbol is $=$ (used as an infix operator, and allowed only at the root position).

Partial order $\preceq$ over positions is defined as follows. For $u, v \in \mathscr{O}(t)$, $v \preceq u$ if v is a prefix of u, i.e. $\exists w$ such that $vw = u$. $v \prec u$ if $v \preceq u$ and $v \neq u$. Positions u and v are called disjoint, written as $u \mid v$, if $\neg(u \preceq v)$ and $\neg(v \preceq u)$. u is called to be to the left of v if u is written as $w_1 i w_2$ and v is written as $v = w_1 j w_3$ for $i < j$. $u \in O(\subseteq \mathscr{O}(t))$ is the leftmost in O if for any $v \in O$ $\neg(v$ is to the left of $u)$.

A substitution is a mapping from $\mathscr{V}$ to $\mathscr{T}$ whose domain $\mathscr{D}\theta$ defined as $\mathscr{D}\theta = \{x \mid \theta x \not\equiv x, x \in \mathscr{V}\}$ is finite. The codomain of θ is defined as $\mathscr{C}od\ \theta = \{\theta x \mid x \in \mathscr{D}\theta\}$. We identify a substitution θ with the set $\{x \mapsto \theta x \mid x \in \mathscr{D}\theta\}$. An empty substitution is defined as the empty set $\emptyset$. A substitution is extended to an endomorphism over $\mathscr{T}$ as usual. For $M \subseteq \mathscr{T}$, θM is defined as $\theta M = \{\theta t \mid t \in M\}$. Let V be a finite subset of $\mathscr{V}$. A variable-renaming substitution is a mapping that maps V to V bijectively. The restriction of θ to V is denoted by $\theta \restriction_V$. We write $\theta_1 = \theta_2[V]$ when $\theta_1 \restriction_V = \theta_2 \restriction_V$. 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 $\theta_1 \preceq \theta_2$.[4] When $\sigma\theta_1 = \theta_2[V]$ holds for some substitution σ, we write $\theta_1 \preceq \theta_2[V]$. Two substitutions θ_1 and θ_2 are equivalent, written as $\theta_1 \sim \theta_2$, if $\theta_1 \preceq \theta_2$ and $\theta_2 \preceq \theta_1$. $\theta_1 \prec \theta_2$ if $\theta_1 \preceq \theta_2$ and θ_1 is not equivalent to θ_2. $\mathscr{V}(\theta)$ is defined as $\mathscr{D}\theta \cup \mathscr{V}(Cod\ \theta)$.

Let $\mathscr{A}$ be any syntactic object and $\rho : V \to V$ be a variable-renaming substitution, where $V \supseteq \mathscr{V}(\mathscr{A})$. $\rho\mathscr{A}$ is called a variant of $\mathscr{A}$.

[2] In Giovannetti *et al.* (1991) and Moreno-Navarro and Rodríguez-Artalejo (1992), a term in $\mathscr{T}(\mathscr{F}_{\mathscr{C}}, \mathscr{V})$ is called a 'data' term.

[3] We use the symbol $\triangleq$ to denote the definitional equality on syntactic objects.

[4] We use the same symbol for order relations over positions and substitutions.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Let R be a binary relation. R^* denotes the reflexive and transitive closure of R, and R^+ is the transitive closure of R.

Let $\mathscr{F}$ and $\mathscr{V}$ be given, and $l, r \in \mathscr{T}(\mathscr{F}, \mathscr{V})$. A rewrite rule is a pair $l \to r$ of terms which satisfies the conditions $l \notin \mathscr{V}$ and $\mathscr{V}(r) \subseteq \mathscr{V}(l)$. A term rewriting system $\mathscr{R}$ is a set of rewrite rules. A term rewriting system $\mathscr{R}$ is called constructor-based if, for every rewrite rule $f(l_1, \ldots, l_n) \to r \in \mathscr{R}$, $f \in \mathscr{F}_{\mathscr{D}}$ and $l_1, \ldots, l_n$ are constructor terms. Let s and t be terms. The rewrite relation $\to_{\mathscr{R}}$ induced from $\mathscr{R}$ is defined as follows: $s \to_{\mathscr{R}} t$ if $s_{|u} \equiv \sigma l$, and $t \equiv s[\sigma r]_u$ for some position $u \in \mathscr{O}(s)$, some substitution σ, and some rewrite rule $l \to r \in \mathscr{R}$. The reflexive and transitive closure $\to^*_{\mathscr{R}}$ of $\to_{\mathscr{R}}$ is also denoted as $\twoheadrightarrow_{\mathscr{R}}$. The relation $=_{\mathscr{R}}$ is defined as the reflexive, transitive, and symmetric closure of $\to_{\mathscr{R}}$.

We further assume the readers' familiarity with basic concepts of term rewriting, as are expounded in Dershowitz and Jouannaud (1990) or Klop (1992).

3 Narrowing calculus

Narrowing is a method for solving an equation for a given term rewriting system (abbreviated as TRS). Operationally, narrowing is a combination of instantiating an equation and a rewrite rule by applying a most general substitution, and of rewriting the instantiated equation by the instantiated rewrite rule to form a new equation (Fay, 1979; Slagle, 1974; Hullot, 1980). Narrowing is successively applied to obtain an equation both sides of which are unifiable. The whole process of successive rewriting of equations is also called 'narrowing'.

3.1 Calculus NC

For our purpose, narrowing is best presented in the form of a calculus $(\mathscr{G}, \mathscr{L})$, where $\mathscr{G}$ is a set of objects manipulated in this calculus, and $\mathscr{L}$ is a set of inference rules that operate on $\mathscr{G}$. Calculus **NC** that will be given in the following operates on goals. A (possibly empty) sequence of equations and true's is called a 'goal'. In this paper, a non-empty goal is usually denoted by E, e, E', where e is an equation and E and E' are goals. An empty goal is denoted by $\Box$.

In this paper we consider narrowing with respect to a particular class of TRSs, called Orthogonal Term Rewriting Systems (OTRS) that are defined as follows:

Definition 3.1 (OTRS)
A TRS $\mathscr{R}$ is called orthogonal if $\mathscr{R}$ satisfies the following conditions:

- [left-linearity] For any rewrite rule $l \to r \in \mathscr{R}$, l is linear.
- [non-ambiguity] For any two variants $l \to r$ and $l' \to r'$ of rewrite rules in $\mathscr{R}$, there is no unifier of l and $l'_{|u}$ for all $u \in \overline{\mathscr{O}}(l')$, except in the case that $l \to r$ and $l' \to r'$ are variants of the same rewrite rule and $u = \varepsilon$.

We treat OTRSs for the following reasons:

- We are primarily interested in the narrowing that is used as a computing mechanism of functional-logic programs. OTRSs model many functional-logic programs.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

- OTRSs have been studied extensively in terms of their behavior in reductions, and established theories of OTRSs can be applied to the study of narrowing.

Our definition of **NC** and some properties pertaining to it, however, are given for arbitrary TRSs. We restrict ourselves to OTRSs in section 4.2 and thereafter, in which we discuss the leftmost-outside-in narrowing.

*Definition 3.2 (***NC** *over goals)*
Let $\mathscr{R}$ be a TRS.[5] A calculus **NC** for $\mathscr{R}$ is a pair $(\mathscr{G}, \mathrm{NC})$, where $\mathscr{G}$ and NC are the following.

- $\mathscr{G}$ is a set of goals.
- NC is a set of inference rules defined as follows:
  - [n] narrowing
    $$\frac{E, e, E'}{\theta E, \theta e[r]_u, \theta E'}$$
    if there exist
    - a new variant $l \to r$ of a rewrite rule in $\mathscr{R}$,
    - $u \in \overline{\mathscr{C}}(e)$,[6] and
    - a most general unifier θ such that $\theta e_{|u} \equiv \theta l$.
  - [f] reflection
    $$\frac{E, s = t, E'}{\theta E, \mathrm{true}, \theta E'}$$
    if s and t are unifiable with a most general unifier θ.

With reference to the inference rule [n], we say that the equation e is narrowed at the position u. A term $e_{|u}$ is called a narrowable expression (narex for short). A term is 'narrowable' if it contains a narex as a subterm. A term that contains no narex is called 'non-narrowable'. Furthermore, a substitution θ is called non-narrowable if $\forall t \in \mathscr{C}od\ \theta$, t is non-narrowable.

Let G and G' be goals. We write $G \overset{\mathrm{n}}{\leadsto}_\theta G'$ if G' is obtained from G by a single application of the inference rule [n], where θ is a most general unifier formed in the application, and likewise for $G \overset{\mathrm{f}}{\leadsto}_\theta G'$. We also write $G \overset{\mathrm{n}}{\leadsto\!\!\!\!\leadsto}_\theta G'$ if G' is obtained from G by zero or more applications of the inference rule [n], and likewise for $G \overset{\mathrm{f}}{\leadsto\!\!\!\!\leadsto}_\theta G'$. The substitution θ is a composition of the substitutions formed in each step. Similarly, we write $G \overset{\mathrm{NC}}{\leadsto}_\theta G'$ when G' is obtained from G by a single application of an inference rule in NC, and likewise for $G \overset{\mathrm{NC}}{\leadsto\!\!\!\!\leadsto}_\theta G'$. A sequence $G_0 \overset{\mathrm{NC}}{\leadsto}_{\theta_1} G_1 \overset{\mathrm{NC}}{\leadsto}_{\theta_2} \cdots \overset{\mathrm{NC}}{\leadsto}_{\theta_k} G_k$ is called an NC-derivation. Let the symbol $\top$ denote generically a sequence of zero or more true's. The NC-derivation that ends with $\top$ is called a successful NC-derivation. When we have a successful NC-derivation starting form G_0, we say that the goal G_0 is solved. A successful NC-derivation $G_0 \overset{\mathrm{NC}}{\leadsto}_{\theta_1} G_1 \overset{\mathrm{NC}}{\leadsto}_{\theta_2} \cdots \overset{\mathrm{NC}}{\leadsto}_{\theta_k} \top$ yields a solution $(\theta_k \cdots \theta_1)\restriction_{\mathscr{V}(G_0)}$ of the goal G_0.

[5] In this paper, $\mathscr{R}$ is assumed not to contain a rewrite rule either side of which is an equation.
[6] Note $u \neq \varepsilon$.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Let A be an NC-derivation $A : G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} G'$. The goal G is called an initial goal of the NC-derivation A. When $k = 0$, the above NC-derivation is empty. The empty NC-derivation is denoted by 0.

Example 3.1
Let $\mathscr{R}$ be a TRS given by:

$$\mathscr{R} = \begin{cases} f(z) \to z, \\ g(1) \to 1. \end{cases}$$

Given an equation $f(g(x)) = f(y)$, there are 13 successful NC-derivations. Among them we give below the following two successful NC-derivations:

$$f(g(x)) = f(y) \overset{\mathrm{f}}{\rightsquigarrow}_{\{y \mapsto g(x)\}} \mathrm{true}, \tag{1}$$

$$\begin{aligned} f(g(x)) = f(y) \quad & \overset{\mathrm{n}}{\rightsquigarrow}_{\{z_1 \mapsto g(x)\}} g(x) = f(y) \overset{\mathrm{n}}{\rightsquigarrow}_{\{z_2 \mapsto y\}} g(x) = y \\ & \overset{\mathrm{n}}{\rightsquigarrow}_{\{x \mapsto 1\}} 1 = y \overset{\mathrm{f}}{\rightsquigarrow}_{\{y \mapsto 1\}} \mathrm{true}. \end{aligned} \tag{2}$$

NC-derivations (1) and (2) yield the solutions $\{y \mapsto g(x)\}$ and $\{x \mapsto 1, y \mapsto 1\}$, respectively. The former solution contains a term that is still narrowable, whereas the latter solution contains only normal forms. The latter is also obtainable from the equation $y = g(x)$ that can be formed from the former solution $\{y \mapsto g(x)\}$. From the viewpoint of equation solving, the former solution would be satisfactory, but from the viewpoint of functional-logic programming, the latter solution $\{x \mapsto 1, y \mapsto 1\}$ is desirable. We will take the viewpoint of programming languages, and consider the latter solution as our intended solution. To guarantee that solutions are normal forms, later we introduce certain syntactic restrictions on goals and rewrite rules.

3.2 Correspondence between narrowing and reduction derivations

By the construction of narrowing, we see a correspondence between the NC-derivation

$$e_0 \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_1} e_1 \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_2} \cdots \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_k} e_k$$

and the reduction derivation

$$\mu_0 e_0 \to_{\mathscr{R}} \mu_1 e_1 \to_{\mathscr{R}} \cdots \to_{\mathscr{R}} \mu_k e_k$$

where $\mu_i = \theta_k \cdots \theta_{i+1}$ for $i = 0, \ldots, k$,

in which the same rewrite rule is employed at the same position in each step of both derivations. Whenever we say an NC-derivation corresponds to a reduction derivation (and *vice versa*), we implicitly assume that the same rewrite rule is employed at the same position in each step of both derivations.

The last step $\overset{\mathrm{f}}{\rightsquigarrow}$ of a successful NC-derivation corresponds to the reduction in which a rewrite rule $x = x \to \mathrm{true}$ is used. Let $\mathscr{R}_+$ denote a TRS extended with the rewrite rule $x = x \to \mathrm{true}$. Then, the whole successful NC-derivation $e_0 \overset{\mathrm{n}}{\rightsquigarrow}_{\theta} \mathrm{true}$ can be made to correspond to the reduction derivation $\theta e_0 \twoheadrightarrow_{\mathscr{R}_+} \mathrm{true}$.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

The correspondence between narrowing and reduction derivations can be extended in an obvious way to the derivations of goals consisting of multiple equations. We should note, however, that the induced reduction relation $\rightarrow$ is with respect to $\mathscr{R}_+$ not to $\mathscr{R}$.

4 Leftmost outside-in narrowing

The calculus **NC** is too general as a computation model in that it does not incorporate a method by which we can locate a narex. Selection of a narex could be specified by a computation rule which is often called a strategy. A computation rule could specify, for example, that the outermost-leftmost narexes be processed first among the narexes. In this paper we are aiming at a calculus in which a computation rule is built in. Towards that goal, we define a special class of successful NC-derivations called leftmost outside-in NC-derivations.

4.1 Well-founded order on a set of successful NC*-derivations*

Huet and Lévy defined the leftmost outside-in reduction derivation inductively on the length of derivations using a set of external positions.[7] In this paper we give an alternative (but equivalent) definition of the leftmost outside-in narrowing derivations inductively on the complexity of derivations. The complexities of successful NC-derivations are compared by an ordering defined over a set of successful NC-derivations.

Let $(D, \trianglerighteq)$ be an ordered set, where D is a set of successful NC-derivations and $\trianglerighteq$ is an ordering over the set D defined below.

Definition 4.1

- The size, size(t), of a term t is the number of variables and function symbols occurring in t.
- The degree $\mathrm{d}(\sigma)$ of a substitution σ is defined as follows:

$$\mathrm{d}(\sigma) = (\sum_{v \in V} \mathrm{size}(\sigma v)) - |\mathscr{V}(\sigma V)|,$$

where $|X|$ for any finite set X is the cardinality of X, and $V = \mathscr{V}(\sigma)$.

The notion of degree was given by Eder (1985). The degree is shown to preserve the orderings $\prec$ and $\sim$. The following proposition is due to Eder (1985):

Proposition 4.1
Let σ and θ be substitutions.

- If $\sigma \sim \theta$ then $\mathrm{d}(\sigma) = \mathrm{d}(\theta)$.
- If $\sigma \prec \theta$ then $\mathrm{d}(\sigma) < \mathrm{d}(\theta)$.

It is easy to see that $\mathrm{d}(\theta)$ is a non-negative integer. As a corollary of Proposition 4.1, we have: $\prec$ is a well-founded ordering.

[7] The definition of an external position is given in section 4.3.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Definition 4.2
Let G be a goal and $A : G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top \in D$. The complexity $\|A\|$ of A is defined as a triple $\langle \mathrm{n}(A), \mathrm{d}(\theta), \#G \rangle$, where

- $\mathrm{n}(A)$ is the number of applications of the inference rule [n] in the derivation A,
- $\mathrm{d}(\theta)$ is the degree of the substitution θ, and
- $\#G$ is the number of variables and function symbols occurring in G excluding the symbols $=$ and true.

Definition 4.3
- The ordering $\gg$ on the complexities of a successful NC-derivation is the lexicographic extension of the ordering $>$ on natural numbers.
- The ordering $\rhd$ on D is defined as follows:
  Let $A, A' \in D$. $A \rhd A'$ if $\|A\| \gg \|A'\|$.
- $\trianglerighteq \triangleq \rhd \cup =$.

Note that the ordering $\rhd$ is well-founded, and hence $(D, \trianglerighteq)$ is a well-founded set. The least (in this ordering) successful NC-derivation is a zero-step NC-derivation $\top \overset{\mathrm{NC}}{\rightsquigarrow}_{\emptyset} \top$. We denote this NC-derivation by $0_{\top}$. We have $\|0_{\top}\| = \langle 0, 0, 0 \rangle$.

Intuitively, a successful NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ describes the process of solving a goal G. The ordering $\rhd$ over NC-derivations specifies which solving processes are simpler. We show an easy example to illustrate this view of NC-derivations. Suppose a successful NC-derivation

$$A : G_0 \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_1} G_1 \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$$

is given. The NC-derivation $A' : G_1 \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ is simpler than A, since A' is included in A. Indeed, in our ordering, we have $A \rhd A'$, since $\mathrm{n}(A') = \mathrm{n}(A) - 1$.

This is not the only case that the solving process is simplified. If we can transform A to another successful NC-derivation $A'' : G'_0 \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta'} \top$ in such a way that

- the goal G_0 is transformed to G'_0 such that $\#G'_0 < \#G_0$,
- θ' is equivalent to θ, and
- $\mathrm{n}(A'') = \mathrm{n}(A)$,

then we simplify the goal and its solving process. Since $(D, \trianglerighteq)$ is a well-founded set, the simplification of successful NC-derivations eventually terminates with the NC-derivation $0_{\top}$. This also suggests an effective means of solving a goal.

There are special pairs of successful NC-derivations related by the subset of the relation $\rhd$. This subset will be denoted by $\overset{\mathrm{OI}}{\rhd}$. The relation $\overset{\mathrm{OI}}{\rhd}$, although found *a posteriori*, gives a hint on the design of a new narrowing calculus that embodies the leftmost outside-in narrowing. The following lemmas (Lemmas 4.2, 4.3, 4.6 and 4.7) enable us to enumerate those pairs of successful NC-derivations. The leftmost outside-in NC-derivation will be defined inductively with respect to the relation $\overset{\mathrm{OI}}{\rhd}$.

Let us now observe that the complexity of an NC-derivation is invariant by 'α-conversion'.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Lemma 4.1
Let $\mathscr{R}$ be an arbitrary TRS, G be a goal, and α be a variable-renaming substitution. If there exists an NC-derivation $A : G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ then there exists an NC-derivation $A' : \alpha G \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma} \top$ such that $\sigma\alpha \sim \theta$. In both NC-derivations the same rewrite rule is employed at the same position of the corresponding equation in each step.

Proof
By the induction on the length of the NC-derivation. □

Clearly, in the above lemma we have $||A|| = ||A'||$.

Lemma 4.2
Let $A : \top, s = x, E \overset{\mathrm{f}}{\rightsquigarrow}_{\eta(=\{x \mapsto s\})} \top, \eta E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ be an NC-derivation and $A' : \eta E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ be the NC-derivation taken from the sub-derivation $\top, \eta E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$. Then $A \rhd A'$.

Proof
It is straightforward to show that $\theta \preceq \theta\eta$. If $\theta \sim \theta\eta$ then $\mathrm{d}(\theta) = \mathrm{d}(\theta\eta)$ by Proposition 4.1. But in this case, we have $s \in \mathscr{V}$, and hence $\#(\top, s = x, E) > \#(\eta E)$. Otherwise, $\mathrm{d}(\theta) < \mathrm{d}(\theta\eta)$. Thus we obtain $A \rhd A'$. □

We write $A \overset{\mathrm{v1}}{\rhd} A'$, if A and A' are the NC-derivations of the forms given above.

Lemma 4.3
Let $A : \top, x = t, E \overset{\mathrm{f}}{\rightsquigarrow}_{\eta(=\{x \mapsto t\})} \top, \eta E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$, where $t \notin \mathscr{V}$ be an NC-derivation and $A' : \eta E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ be the NC-derivation taken from the sub-derivation $\top, \eta E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$. Then $A \rhd A'$.

Proof
Similar to the proof of Lemma 4.2. □

We write $A \overset{\mathrm{v2}}{\rhd} A'$, if A and A' are the NC-derivations of the forms given above.

To prove Lemmas 4.6 and 4.7, we need the following lemmas.

Notation 4.1
Let G be a goal consisting of $n(\geqslant 0)$ equations, and $i_1, \ldots, i_n$ be a permutation of $1, \ldots, n$. $G[i_1, \ldots, i_j], j \leqslant n$ denotes a goal obtained from G by replacing the i_1-th, $\ldots$, i_j-th equations in the goal G by true's.

Lemma 4.4 (Decomposition lemma)
If there exists an NC-derivation $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n) \overset{\mathrm{f}}{\rightsquigarrow}_{\theta} \mathrm{true}$, then there exists an NC-derivation $s_1 = t_1, \ldots, s_n = t_n \overset{\mathrm{f}}{\rightsquigarrow}_{\sigma} \top$, such that $\sigma \sim \theta$.

Proof
Let $G \triangleq s_1 = t_1, \ldots, s_n = t_n$. It is easy to show that for $j = 1, \ldots, n$, there exists an NC-derivation

$$\sigma_{i_{j-1}} \cdots \sigma_{i_1} G[i_1, \ldots, i_{j-1}] \overset{\mathrm{f}}{\rightsquigarrow}_{\sigma_{i_j}\sigma_{i_j}\sigma_{i_{j-1}} \cdots \sigma_{i_1}} G[i_1, \ldots, i_j],$$

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

where $\sigma_{i_{j-1}} \cdots \sigma_{i_1} \preceq \theta$ and $\sigma_{i_j}\sigma_{i_{j-1}} \cdots \sigma_{i_1} \preceq \theta$. By combining the above n NC-derivations and letting $\sigma = \sigma_{i_n} \cdots \sigma_{i_1}$, we obtain an NC-derivation

$$s_1 = t_1, \ldots, s_n = t_n \overset{\text{f}}{\rightsquigarrow}_{\sigma} \top,$$

such that $\sigma \preceq \theta$. Since σ and θ are unifiers of $f(s_1, \ldots, s_n)$ and $f(t_1, \ldots, t_n)$, and θ is most general, we also have $\theta \preceq \sigma$. Hence $\sigma \sim \theta$. □

Lemma 4.5 ($\overset{\text{f}}{\rightsquigarrow}$-promotion lemma)
Let $\mathscr{R}$ be an arbitrary TRS and G be a goal consisting of $n(\geqslant 0)$ equations. If there exists a successful NC-derivation

$$A : G \overset{\text{n}}{\rightsquigarrow}_{\theta'} G' \overset{\text{f}}{\rightsquigarrow}_{\theta_1} \theta_1 G'[1] \overset{\text{f}}{\rightsquigarrow}_{\theta_2} \cdots \overset{\text{f}}{\rightsquigarrow}_{\theta_{n-1}} \theta_{n-1} \cdots \theta_2\theta_1 G'[1, \ldots, n-1]$$

$$\overset{\text{f}}{\rightsquigarrow}_{\theta_n} \top,$$

that yields a substitution $\theta = \theta_n \ldots \theta_2\theta_1\theta'$,

then there exists a successful NC-derivation

$$A' : G \overset{\text{n}}{\rightsquigarrow} \overset{\text{f}}{\rightsquigarrow}_{\sigma_1} \sigma_1 G[i_1] \overset{\text{n}}{\rightsquigarrow} \overset{\text{f}}{\rightsquigarrow}_{\sigma_2} \cdots \overset{\text{n}}{\rightsquigarrow} \overset{\text{f}}{\rightsquigarrow}_{\sigma_{n-1}} \sigma_{n-1} \ldots \sigma_1 G[i_1, \ldots, i_{n-1}]$$

$$\overset{\text{n}}{\rightsquigarrow} \overset{\text{f}}{\rightsquigarrow}_{\sigma_n} \top, \text{ where } i_1, \ldots, i_n \text{ is a permutation of } 1, \ldots, n,$$

that yields a substitution $\sigma = \sigma_n \ldots \sigma_2\sigma_1$,

such that

- all the $\overset{\text{n}}{\rightsquigarrow}$-steps in A are also taken in A' in the same order as in A,
- in each corresponding $\overset{\text{n}}{\rightsquigarrow}$-steps the same rewrite rule is employed at the same position,
- in the last step of $\overset{\text{n}}{\rightsquigarrow}$ -steps (if any) of each sub-derivation

  $$\sigma_{j-1} \ldots \sigma_1 G[i_1, \ldots, i_{j-1}] \overset{\text{n}}{\rightsquigarrow} \overset{\text{f}}{\rightsquigarrow}_{\sigma_j} \sigma_j \ldots \sigma_1 G[i_1, \ldots, i_j], \text{ for } j = 1, \ldots, n,$$

  narrowing contraction is performed on the i_j-th equation of the goal G, and
- $\theta \sim \sigma$.

Proof
By repeated applications of the switching lemma (Lemma A1) given in the Appendix. We switch adjacent $\overset{\text{n}}{\rightsquigarrow}$- and $\overset{\text{f}}{\rightsquigarrow}$-steps, and adjacent $\overset{\text{f}}{\rightsquigarrow}$- and $\overset{\text{f}}{\rightsquigarrow}$-steps. □

We call the NC-derivation A' in the $\overset{\text{f}}{\rightsquigarrow}$-promotion lemma the $\overset{\text{f}}{\rightsquigarrow}$-promoted NC-derivation of A. The idea of Lemma 4.5 is as follows. Whenever one equation in the goal is ready to be solved (i.e. the rule [f] is applicable), after zero or more applications of the rule [n] on that equation in the same order as in A, we apply the rule [f] immediately.

The following simple example explains a $\overset{\text{f}}{\rightsquigarrow}$-promoted NC-derivation clearly.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Example 4.1
Suppose a TRS

$$\mathscr{R} = \begin{cases} a \to b, \\ b \to \mathrm{c}, \end{cases}$$

is given. We have a successful NC-derivation

$$A : a = x, a = \mathrm{c} \overset{\mathrm{n}}{\rightsquigarrow}_{\emptyset} a = x, b = \mathrm{c} \overset{\mathrm{n}}{\rightsquigarrow}_{\emptyset} a = x, \mathrm{c} = \mathrm{c} \overset{\mathrm{n}}{\rightsquigarrow}_{\emptyset} b = x, \mathrm{c} = \mathrm{c}$$
$$\overset{\mathrm{f}}{\rightsquigarrow}_{\{x \mapsto b\}} \mathrm{true}, \mathrm{c} = \mathrm{c} \overset{\mathrm{f}}{\rightsquigarrow}_{\emptyset} \mathrm{true}, \mathrm{true}.$$

The $\overset{\mathrm{f}}{\rightsquigarrow}$-promoted NC-derivation A' of A is the following.

$$A' : a = x, a = \mathrm{c} \overset{\mathrm{n}}{\rightsquigarrow}_{\emptyset} a = x, b = \mathrm{c} \overset{\mathrm{n}}{\rightsquigarrow}_{\emptyset} a = x, \mathrm{c} = \mathrm{c} \overset{\mathrm{f}}{\rightsquigarrow}_{\emptyset} a = x, \mathrm{true}$$
$$\overset{\mathrm{n}}{\rightsquigarrow}_{\emptyset} b = x, \mathrm{true} \overset{\mathrm{f}}{\rightsquigarrow}_{\{x \mapsto b\}} \mathrm{true}, \mathrm{true}$$

A' is obtained from A by a single $\overset{\mathrm{f}}{\rightsquigarrow}$- and $\overset{\mathrm{f}}{\rightsquigarrow}$-switching, and by a single $\overset{\mathrm{n}}{\rightsquigarrow}$- and $\overset{\mathrm{f}}{\rightsquigarrow}$-switching. In A' the order of $\overset{\mathrm{n}}{\rightsquigarrow}$-steps is the same as in A. The difference is that when $\mathrm{c} = \mathrm{c}$ is ready to be solved after the second step, the third step is the $\overset{\mathrm{f}}{\rightsquigarrow}$-step applied to $\mathrm{c} = \mathrm{c}$ rather than the $\overset{\mathrm{n}}{\rightsquigarrow}$-step on $a = x$, i.e. $a = x \overset{\mathrm{n}}{\rightsquigarrow}_{\emptyset} b = x$.

The following notion of descendant is used in Lemmas 4.6 and 4.7.

Definition 4.4
Let $G(\triangleq e_1, \ldots, e_n) \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} G'(\triangleq e'_1, \ldots, e'_n)$ be an NC-derivation.

- The term $e'_i, i \in \{1, \ldots, n\}$ is called an immediate descendant of e_i, written as $e_i \hookrightarrow e'_i$.
- A term e' is a descendant of e if $e \hookrightarrow^* e'$, where $\hookrightarrow^*$ is the reflexive and transitive closure of $\hookrightarrow$.

Lemma 4.6
Let $\mathscr{R}$ be an arbitrary TRS. If there exists an NC-derivation

$$A : \top, f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E$$
$$\overset{\mathrm{n}}{\rightsquigarrow}_{\theta_1} \top, f(s'_1, \ldots, s'_n) = f(t'_1, \ldots, t'_n), \theta_1 E \overset{\mathrm{f}}{\rightsquigarrow}_{\theta_2} \top, \theta_2\theta_1 E$$
$$\overset{\mathrm{NC}}{\rightsquigarrow}_{\theta'} \top, \tag{3}$$

where the descendants of $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$ are not narrowed at the position 1 or 2, then there exists an NC-derivation

$$A' : s_1 = t_1, \ldots, s_n = t_n, E \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma_1} \top, \sigma_1 E \tag{4}$$
$$\overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma'} \top, \tag{5}$$

such that

- $\sigma'\sigma_1 \sim \theta'\theta_2\theta_1$,
- the derivation $s_1 = t_1, \ldots, s_n = t_n \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma_1} \top$ extracted from the sub-derivation (4) is the $\overset{\mathrm{f}}{\rightsquigarrow}$-promoted derivation of $s_1 = t_1, \ldots, s_n = t_n \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_1} s'_1 = t'_1, \ldots, s'_n = t'_n \overset{\mathrm{f}}{\rightsquigarrow}_{\theta'_2} \top$, where $\theta'_2 \sim \theta_2$, and

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

- in each step of the sub-derivations (3) and (5), the same rewrite rule is employed at the same position of the corresponding equation in each goal.

Furthermore, we have $A \rhd A'$.

Proof
From the NC-derivation A, using Lemma 4.4, we can construct an NC-derivation

$$B : s_1 = t_1, \ldots, s_n = t_n, E \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_1} s'_1 = t'_1, \ldots, s'_n = t'_n, \theta_1 E \overset{\mathrm{f}}{\rightsquigarrow}_{\theta'_2} \top, \theta'_2\theta_1 E,$$

such that $\theta'_2 \sim \theta_2$. By Lemma 4.5, we have the f-promoted derivation B' of B

$$B' : s_1 = t_1, \ldots, s_n = t_n, E \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma_1} \top, \sigma_1 E.$$

with $\sigma_1 \sim \theta'_2\theta_1 \sim \theta_2\theta_1$. We have $\sigma_1 = \rho\theta_2\theta_1$ for some variable-renaming substitution ρ. Applying Lemma 4.1 to the derivation $\theta_2\theta_1 E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta'} \top$, we have an NC-derivation

$$B'' : \sigma_1 E \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma'} \top,$$

such that $\sigma'\rho \sim \theta'$. Concatenating B' and B'', we obtain the desired derivation A' with $\theta'\theta_2\theta_1 \sim \sigma'\rho\theta_2\theta_1 = \sigma'\sigma_1$. We have $A \rhd A'$ since $\mathrm{n}(A) = \mathrm{n}(A')$, $\mathrm{d}(\theta'\theta_2\theta_1) = \mathrm{d}(\sigma'\sigma_1)$ and $\#(\top, f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E) > \#(s_1 = t_1, \ldots, s_n = t_n, E)$. □

We write $A \overset{\mathrm{d}}{\rhd} A'$, if A and A' are the NC-derivations of the forms given above.

In the following Lemma 4.7 we consider the case that as for the first leftmost equation narrowing is applied to the left-hand sides of the equations in the goals. This is because Lemma 4.7 will be used in a restricted context, as we will see in section 4.2.

Lemma 4.7
Let $\mathscr{R}$ be an arbitrary TRS. If there exists an NC-derivation

$$A : \top, f(s_1, \ldots, s_n) = t, E \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_1} \top, f(s'_1, \ldots, s'_n) = \theta_1 t, \theta_1 E \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_2} \top, \theta_2 r = \theta_2\theta_1 t, \theta_2\theta_1 E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta'} \top, \quad (6)$$

where $t \notin \mathscr{V}$,

in which the descendant of $f(s_1, \ldots, s_n) = t$ is narrowed for the first time at the position 1, in some step in A using a new variant $f(l_1, \ldots, l_n) \to r$ of a rewrite rule in $\mathscr{R}$, then there exists an NC-derivation

$$A' : s_1 = l_1, \ldots, s_n = l_n, r = t, E \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma_1} \top, \sigma_1 r = \sigma_1 t, \sigma_1 E \quad (7)$$

$$\overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma'} \top, \quad (8)$$

such that

- $\sigma'\sigma_1 \sim \theta'\theta_2\theta_1$,
- the derivation $s_1 = l_1, \ldots, s_n = l_n \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma_1} \top$ extracted from the sub-derivation (7) is the $\overset{\mathrm{f}}{\rightsquigarrow}$-promoted derivation of $s_1 = l_1, \ldots, s_n = l_n \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_1} s'_1 = l_1, \ldots, s'_n = l_n \overset{\mathrm{f}}{\rightsquigarrow}_{\theta'_2} \top$, where $\theta'_2 \sim \theta_2$, and

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

- in each step of the sub-derivations (6) and (8), the same rewrite rule is employed at the same position of the same equation in each goal.

Furthermore, we have $A \rhd A'$.

Proof
Similar to the proof of the previous lemma. Since $\mathrm{n}(A') = \mathrm{n}(A) - 1$, we have $A \rhd A'$. □

We write $A \overset{\mathrm{on}}{\rhd} A'$, if A and A' are the NC-derivations of the forms given above.

Applying the above lemmas (Lemmas 4.2, 4.3, 4.6 and 4.7) repeatedly, we can transform a successful NC-derivation to a simpler one.

The transformation of successful NC-derivations can be more abstractly treated by the ordering $\overset{\mathrm{OI}}{\rhd}$ mentioned at the beginning of this subsection. Thus, we have the following definition of $\overset{\mathrm{OI}}{\rhd}$:

Definition 4.5
An ordering $\overset{\mathrm{OI}}{\rhd}$ over a set D of successful NC-derivations is defined as

$$\overset{\mathrm{OI}}{\rhd} = \overset{\mathrm{v1}}{\rhd} \cup \overset{\mathrm{v2}}{\rhd} \cup \overset{\mathrm{d}}{\rhd} \cup \overset{\mathrm{on}}{\rhd}.$$

When $A \overset{\mathrm{OI}}{\rhd} A'$, A' is said to be simpler than A. The ordering $\overset{\mathrm{OI}}{\rhd}$ is crucial in defining a leftmost outside-in NC-derivation in section 4.2.

Note in passing the following:

Fact 4.1
If $A : t = s \overset{\mathrm{f}}{\rightsquigarrow}_{\theta} \top$ then $A \ (\overset{\mathrm{d}}{\rhd} \cup \overset{\mathrm{v1}}{\rhd} \cup \overset{\mathrm{v2}}{\rhd})^{+} \ 0_{\top}$.

4.2 *Leftmost outside-in* NC*-derivations*

We next discuss what class of successful NC-derivations we will transform with the relation $\overset{\mathrm{OI}}{\rhd}$. *A priori*, at this point we shall examine a class of NC-derivations, to be called initial NC-derivations. This restriction is justified in view of our interest in narrowing methods that can be used in functional-logic programs. The reason will be clearer when we discuss the strict equality in section 6.

We first give necessary definitions.

Definition 4.6
A goal G is called right-normal if for every equation in G its right-hand side is a ground normal form.

The restriction of right-normality on goals is slightly more general than what we actually need, since we are interested in solving a strict equation $s \equiv t = \mathrm{true}$.

The choice of right-normal initial goals as our objects of narrowing leads us to a class of goals called proper goals.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Definition 4.7

- A successful NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_\theta \top$ is called initial[8] if G is right-normal.
- Let $D_0 (\subseteq D)$ be a set of initial successful NC-derivations, and

$$\mathscr{G}_0 = \{G \mid G \text{ is an initial goal of } A' \text{ such that } A \overset{\mathrm{OI}*}{\rhd} A', A \in D_0\}.$$

  An element of $\mathscr{G}_0$ is called a proper goal.

We start with an initial NC-derivation A and simplify it successively to A' using $\overset{\mathrm{OI}}{\rhd}$. In the course of this transformation, initial goals that are not right-normal are introduced. Those initial goals, together with the original right-normal goals, are collectively called 'proper' goals.

A proper goal has the following properties.

Definition 4.8

Let G be a goal $e_1, \ldots, e_i, \ldots, e_n$, and $\mathrm{Vleft}(G, i) = \mathscr{V}(e_1) \cup \ldots \cup \mathscr{V}(e_{i-1}) \cup \mathscr{V}(e_{i|_1})$.

- The i-th equation e_i of G is called left-independent if $\mathrm{Vleft}(G, i) \cap \mathscr{V}(e_{i|_2}) = \emptyset$.
- A goal G is called left-independent if all the equations in G are left-independent.

By the definition of proper goals, we can easily see that the following proposition holds if we restrict ourselves to OTRSs.

Proposition 4.2

Let $\mathscr{R}$ be an OTRS. A proper goal G has the following properties:

(G1) G is left-independent.
(G2) The right-hand side of every equation in G is linear and non-narrowable.

Note that the left-linearity of $\mathscr{R}$ is necessary for the properties (G1) and (G2), the non-ambiguity of $\mathscr{R}$ for the property (G2). Because of the condition (G2), we only have to consider narrowing on the left-hand side of equations.

We are now ready to define a leftmost outside-in NC-derivation starting from a proper goal for OTRSs.

Definition 4.9

Let $\mathscr{R}$ be an OTRS, and G be a proper goal. A Leftmost Outside-In (LOI) NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_\theta \top$ is defined inductively (with respect to $\overset{\mathrm{OI}}{\rhd}$) as follows.

- An empty successful NC-derivation $0_\top$ is LOI.
- $A : G \overset{\mathrm{NC}}{\rightsquigarrow}_\theta \top$ is LOI if there exists an NC-derivation A' such that $A \overset{\mathrm{OI}}{\rhd} A'$ and A' is LOI.

Note that we define the notion of LOI on successful NC-derivations. We will see in the next section that the notion of LOI NC-derivation is closely related to that of the standard reduction derivation of Huet and Lévy.

[8] This should not be confused with the notion of an initial goal.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

4.3 Standard reduction derivation

Huet and Lévy's definition of a standard reduction derivation is given via the notion of an external redex position. To define an external redex position, we first give the definition of a residual. Let $s \to_U t$ be an elementary reduction, contracting the redexes $s_{|u_1}, \ldots, s_{|u_n}$, where $U = \{u_1, \ldots, u_n\} \subseteq \mathcal{O}(s)$, and u_i and $u_j (i \neq j)$ are pairwise disjoint. Suppose a reduction derivation[9]

$$A : s_0 \to_{U_1} s_1 \to_{U_2} \cdots \to_{U_k} s_k \text{ is given.}$$

Let $A[i, j]$ denote a reduction derivation $s_i \to_{U_{i+1}} s_{i+1} \to \cdots \to_{U_j} s_j (i \leqslant j)$, and $A[i]$ denote $A[i, k]$. A reduction derivation may be written as juxtaposition of sub-derivations.

Let Redex(s) denote the set of all the redex positions in a term s.

Definition 4.10

Let $A : s \to_{\{u\}} t$ be an elementary reduction derivation and $v \in \text{Redex}(s)$. Suppose $l \to r$ is used to contract the redex $s_{|u}$. The set $v \backslash A$ of residuals of v by A is defined as follows:

$$v \backslash A = \begin{cases} \{v\} & \text{if } v \mid u \text{ or } v \prec u, \\ \{uw_1v_1 \mid r_{|w_1} \equiv l_{|w}\} & \\ & \text{if } v = uwv_1 \text{ and } w \in \mathcal{O}(l) - \overline{\mathcal{O}}(l), \\ \emptyset & \text{otherwise.} \end{cases}$$

The notion of residual is extended to non-elementary reduction derivations in the following way:

$$v \backslash 0 = \{v\},$$
$$v \backslash (AB) = \bigcup_{w \in v \backslash A} w \backslash B.$$

A set Red(A) of initial redex positions contributing to A is defined as follows:

$$\text{Red}(A) = \{u \in \text{Redex}(s_0) \mid \exists i \leqslant k \; U_i \cap (u \backslash A[1, i-1]) \neq \emptyset\}.$$

The reduction derivation A preserves $u \in \mathcal{O}(s_0)$ if A does not contract a redex above u, i.e. $\forall i \leqslant k \; \neg(\exists v \in U_i, v \prec u)$. An external position is then defined as follows:

Definition 4.11

Let A be a reduction derivation starting from s. A position u in s is external for A if

[9] To be precise, the notion of a reduction derivation

$$s_0 \to_{\mathscr{R}} s_1 \to_{\mathscr{R}} \cdots \to_{\mathscr{R}} s_k, \text{ given in section 3.2,}$$

and the notion of the reduction derivation

$$s_0 \to_{U_1} s_1 \to_{U_2} \cdots \to_{U_k} s_k, \text{ that is being defined here,}$$

are different. The former reduction derivation is made into the latter by taking $U_i, i = 1, \ldots, k$ to be a singleton set consisting of a redex position where the reduction occurs in $s_{i-1} \to_{\mathscr{R}} s_i$. Having this distinction in mind, we use the former notation for the reduction derivation defined here, as well.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

- A preserves u, or
- $A = A_1A_2A_3$ and there exists $v \prec u$ such that
  - — A_1 preserves u,
  - — $A_2 : t \rightarrow_V t'$, with $v \in V$ and $u \in \mathrm{Pattern}(t, v)$, where
    - – $\mathrm{Pattern}(t, v) = \{vw \in \mathscr{O}(t) \mid w \in \overline{\mathscr{O}}(l)\}$, and
    - – $l \rightarrow r$ is a rewrite rule used to contract the redex $t_{|v}$,

    and
  - — v is external for A_3.

The set of external positions for A is denoted by $\mathscr{X}(A)$. The set $\mathscr{E}(A)$ of external redex positions for A is defined as $\mathscr{E}(A) = \mathscr{X}(A) \cap \mathrm{Red}(A)$. Finally, we have the definition of standard reduction derivation.

Definition 4.12
A reduction derivation A is called standard if either $A = 0$ or $A = A_1A_2$, where A_1 is the elementary reduction derivation contracting the leftmost redex position in $\mathscr{E}(A)$ and A_2 is standard.

The standard reduction derivation starting from a term s has the following important properties:

(i) It represents the class of the reduction derivations starting from s.
(ii) The standard reduction derivation from s to its normal form is the one that contracts only the redexes that are needed to get to the normal form.

Clause (i) accounts for the use of the terminology 'standard'. It is formally stated as follows:

Theorem 4.1 (Standardization theorem (Huet and Lévy, 1991))
Let $\mathscr{R}$ be an OTRS. Every reduction derivation class contains a unique standard reduction derivation.

We next consider to apply the above theorem to reduction derivations starting from goals. For a non-empty goal G the reduction derivation $G \twoheadrightarrow_{\mathscr{R}_+} \top$ contains a reduction derivation

$$e_0 \rightarrow e_1 \rightarrow \cdots \rightarrow e_k \rightarrow \mathrm{true}, \text{ where } e_0 \text{ is in } G,$$

the last step of which is the reduction by the rewrite rule $x = x \rightarrow \mathrm{true}$, and the rest of which are reductions by the rewrite rule in OTRS $\mathscr{R}$. Since we defined the notion of standard reduction for OTRSs, we have to extend the definition of standard reduction derivation to the reduction derivation starting from an equation and ending with true.

Definition 4.13
Let $\mathscr{R}$ be an OTRS and e an equation. A reduction derivation

$$e_0 \rightarrow_{\mathscr{R}} e_1 \rightarrow_{\mathscr{R}} \cdots \rightarrow_{\mathscr{R}} e_k \rightarrow_{\mathscr{R}_+} \mathrm{true}, k \geqslant 0$$

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

is standard if

$$e_0 \to_{U_1} e_1 \to_{U_2} \cdots \to_{U_k} e_k,$$

where $U_i = \{u_i\}$ and $e_{|u_i}$ is a redex contracted at the reduction $e_{i-1} \to_{\mathscr{R}} e_i$ for $i = 1, \ldots, k$ is standard.

Then, we have the following definition of standard reduction derivation starting from a goal:

Definition 4.14
Let $\mathscr{R}$ be an OTRS, and G be a goal. A standard reduction A from a goal G to $\top$ is inductively defined as follows:

- $A : \Box \to_{\emptyset} \Box$ is standard.
- $A : e, E \to\!\!\!\!\to_{\mathscr{R}_+} \text{true}, E \to\!\!\!\!\to_{\mathscr{R}_+} \top$ is standard if
  - — $e \to\!\!\!\!\to_{\mathscr{R}_+} \text{true}$ is standard, and
  - — $E \to\!\!\!\!\to_{\mathscr{R}_+} \top$ is standard.

By this definition of standard reduction derivation starting from a goal we obtain the following proposition as a corollary of Theorem 4.1, which we will use later.

Corollary 4.1
Let $\mathscr{R}$ be an OTRS, and G be an arbitrary goal. If there exists a reduction derivation $G \to\!\!\!\!\to_{\mathscr{R}_+} \top$ then there exists a standard reduction derivation $G \to\!\!\!\!\to_{\mathscr{R}_+} \top$.

4.4 Standard NC*-derivation*

We next apply the notion of 'standard' of the reduction derivation to NC-derivations using the correspondence between narrowing and reduction derivations that we saw in section 3.2. Some precaution is necessary. We cannot simply define a standard NC-derivation as the one whose corresponding reduction derivation is standard for the following reasons:

- We deal with NC-derivations that are generated by the transformation, using the relation $\overset{\text{OI}}{\rhd}$, of initial NC-derivations.
- The relation $\overset{\text{OI}}{\rhd}$ does not preserve the 'standard' property of a reduction derivation.

To see the points above, let us take an example.

Example 4.2
Let $\mathscr{R}$ be given as follows:

$$\mathscr{R} = \begin{cases} f(\mathrm{c}(x)) \to x, \\ a \to \mathrm{b}. \end{cases}$$

We want an NC-derivation $f(\mathrm{c}(a)) = \mathrm{b} \overset{\text{n}}{\rightsquigarrow}_{\{x \mapsto a\}} a = \mathrm{b} \overset{\text{n}}{\rightsquigarrow}_{\emptyset} \mathrm{b} = \mathrm{b} \overset{\text{f}}{\rightsquigarrow}_{\emptyset} \text{true}$ to be standard, since the reduction derivation $f(\mathrm{c}(a)) = \mathrm{b} \to a = \mathrm{b} \to \mathrm{b} = \mathrm{b} \to \text{true}$ is standard.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

To solve a goal $f(\mathrm{c}(a)) = \mathrm{b}$, we may opt for solving a goal $\mathrm{c}(a) = \mathrm{c}(x), x = \mathrm{b}$. The goal $\mathrm{c}(a) = \mathrm{c}(x)$ is solved by an NC-derivation $\mathrm{c}(a) = \mathrm{c}(x) \overset{\mathrm{f}}{\rightsquigarrow}_{\{x \mapsto a\}}$ true. This NC-derivation should be standard, since the corresponding reduction $\mathrm{c}(a) = \mathrm{c}(a) \to$ true is standard by definition. The goal $\mathrm{c}(a) = \mathrm{c}(x)$ may be solved by another NC-derivation $\mathrm{c}(a) = \mathrm{c}(x) \overset{\mathrm{n}}{\rightsquigarrow}_{\emptyset} \mathrm{c}(\mathrm{b}) = \mathrm{c}(x) \overset{\mathrm{f}}{\rightsquigarrow}_{\{x \mapsto \mathrm{b}\}}$ true. Although the reduction derivation $\mathrm{c}(a) = \mathrm{c}(\mathrm{b}) \to \mathrm{c}(\mathrm{b}) = \mathrm{c}(\mathrm{b}) \to$ true is standard, we do not want this NC-derivation to be standard since intuitively the narrowing step $\mathrm{c}(a) = \mathrm{c}(x) \overset{\mathrm{n}}{\rightsquigarrow}_{\emptyset} \mathrm{c}(\mathrm{b}) = \mathrm{c}(x)$ is superfluous.

Based on the above observation, we define a standard NC-derivation as follows:

Definition 4.15
Let $\mathscr{R}$ be an OTRS.

- Let e_0 be a proper goal. An NC-derivation $e_0 \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_1} e_1 \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_2} \cdots \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_k} e_k (\triangleq s_k = t_k) \overset{\mathrm{f}}{\rightsquigarrow}_{\theta_{k+1}}$ true, $k \geqslant 0$, is standard[10] if

(S1) the corresponding reduction derivation

$$\mu_0 e_0 \to_{\{1.u_1\}} \mu_1 e_1 \to \cdots \to_{\{1.u_k\}} \mu_k e_k \to \text{true},$$

where $\mu_i = \theta_{k+1} \cdots \theta_{i+1}$ for $i = 0, \ldots, k$, is standard, and

(S2) $\min(\{u_1, \ldots, u_k\}) \subseteq \overline{\mathscr{O}}(t_k)$, where $\min(U) = \{v \in U \mid \neg(\exists u \in U, u \prec v)\}$, and $\{u_1, \ldots, u_k\}$ is a multi-set consisting of positions $u_1, \ldots, u_k$.

- Let G be a proper goal. A standard NC-derivation for a successful NC-derivation $A : G \overset{\mathrm{NC}}{\rightsquigarrow}_{\eta} \top$ is inductively defined as follows.
  - An empty successful NC-derivation $0_{\top}$ is standard.
  - $A : \top, e, E \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma} \top, \sigma E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ (the leftmost equation is always solved first) is standard if
    - $e \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma}$ true extracted from the above NC-derivation is standard, and
    - $\sigma E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ extracted from the above NC-derivation is standard.

For the NC-derivation $s = t \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta}$true where t is a ground term, the condition (S2) is always satisfied. If t is a variable, only $s = t \overset{\mathrm{f}}{\rightsquigarrow}_{\{t \mapsto s\}}$true is a standard NC-derivation although s may be narrowable. The condition (S2) is imposed to prevent such an s from being narrowed. Initial goals such as $s = t$ above may in general be generated for solving a goal that contains s as a subterm.

For example, to solve $f(s_1, \ldots, s_n) = t$, we will solve the equations $s_1 = l_1, \ldots, s_n = l_n, r = t$, where $f(l_1, \ldots, l_n) \to r$ is employed at some step of the NC-derivation $f(s_1, \ldots, s_n) = t \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta}$true. Clearly, if the NC-derivation $f(s_1, \ldots, s_n) = t \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta}$true satisfies the condition (S1), then the NC-derivations solving each (descendant of) equation of the goal $s_1 = l_1, \ldots, s_n = l_n, r = t$ satisfy the condition (S1). However, this decomposition of the equation solving does not preserve the 'standard' property

[10] In Huet and Lévy (1991), a standard reduction derivation is also called a leftmost outside-in reduction derivation. Since we want to distinguish the notion introduced by Huet and Lévy and the notion of our LOI derivation, we always call Huet and Lévy's leftmost outside-in reduction 'standard reduction'.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

in the converse direction. With the condition (S2) the converse holds. Suppose $e_0 \triangleq s_i = l_i$ in clause (S1) of Definition 4.15. Condition (S2) guarantees that $i.u_1, \ldots, i.u_k \in \mathrm{Pattern}(f(s'_1, \ldots, s'_n) = t, 1)$ in B_2 of the reduction derivation $B = B_1B_2B_3$ reconstructed from the NC-derivation $s_1 = l_1, \ldots, s_n = t_n, r = t \overset{\mathrm{NC}}{\rightsquigarrow}_\theta \top$, where

- $B_1 : f(s_1, \ldots, s_n) = t \twoheadrightarrow f(s'_1, \ldots, s'_n) = t$,
- $B_2 : f(s'_1, \ldots, s'_n) = t \rightarrow \sigma r = t$, and
- $B_3 : \sigma r = t \twoheadrightarrow t = t$ in Definition 4.11.

Hence, we have $u_j \in \mathscr{X}(B)$ for $j = 1, \ldots, k$.

The following lemma is an immediate consequence of the definition of standard NC-derivation:

Lemma 4.8
Let A be a successful NC-derivation starting from a proper goal. A is standard $\Leftrightarrow$ a successful NC-derivation B such that $A \overset{\mathrm{OI}}{\triangleright} B$ is standard.

By Lemma 4.8 it is easy to see the equivalence of standard and LOI NC-derivations.

Theorem 4.2
Let A be a successful NC-derivation starting from a proper goal. A is standard $\Leftrightarrow$ A is LOI.

4.5 *Standardization of* NC*-derivations*

We can obtain a standardization theorem for narrowing derivations using the following lifting lemma (Hullot, 1980; Middeldorp and Hamoen, 1994).

Lemma 4.9 (Lifting lemma for NC*-derivations)*
Let $\mathscr{R}$ be an arbitrary TRS. Suppose G is an arbitrary goal, θ is a normalized substitution and V is a set of variables such that $\mathscr{V}(G) \cup \mathscr{D}\theta \subseteq V$. If there exists a reduction derivation

$$\theta G \twoheadrightarrow_{\mathscr{R}_+} G'$$

then there exist a goal G'' and substitutions θ' and σ such that

- $G \overset{\mathrm{NC}}{\rightsquigarrow}_\sigma G''$,
- $\theta' G'' \equiv G'$,
- $\theta'\sigma = \theta[V]$, and
- θ' is normalized.

In both derivations the same rewrite rule is employed at the same position of the corresponding equation in each step.

Proposition 4.3, which states the property of the solutions of initial NC-derivations, and Lemma 4.10 will be used in the proof of the standardization theorem.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Proposition 4.3
Let $\mathscr{R}$ be an OTRS, and G be a right-normal goal. If there exists a successful NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$, then the solution $\theta\restriction_{\mathscr{V}(G)}$ is non-narrowable.

Lemma 4.10
Let $\mathscr{R}$ be an OTRS, θ be a normalized substitution, and G be a goal. Suppose we have reduction derivations $A : \theta G \twoheadrightarrow_{\mathscr{R}_+} \top$ and $A' : \sigma G \twoheadrightarrow_{\mathscr{R}_+} \top$ such that $\xi\sigma = \theta$ with ξ normalized, and in each corresponding reduction step of both derivations the same rewrite rule is employed at the same position of the same equation. If A is standard then A' is standard.

Proof
By the induction on the length of the derivation. We only have to note that no new redex is created by lifting from A to A'. □

Theorem 4.3 (Standardization theorem for NC*-derivations)*
Let $\mathscr{R}$ be an OTRS and G be a right-normal goal. If there exists a successful NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$, then there exists an LOI NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma} \top$ such that $\sigma \preceq \theta[\mathscr{V}(G)]$.

Proof
By Proposition 4.3 the substitution $\eta = \theta\restriction_{\mathscr{V}(G)}$ is non-narrowable. By the correspondence of narrowing and reduction derivations, there exists a reduction derivation $\eta G \twoheadrightarrow_{\mathscr{R}_+} \top$. By Theorem 4.1 there exists a standard reduction derivation $A : \eta G \twoheadrightarrow_{\mathscr{R}_+} \top$. By Lifting Lemma 4.9, we obtain a successful NC-derivation $B : G \overset{\mathrm{NC}}{\rightsquigarrow}_{\sigma} \top$ such that $\xi\sigma = \eta[\mathscr{V}(G)]$ for some normalized substitution ξ. Let B' be the reduction derivation $\sigma G \twoheadrightarrow_{\mathscr{R}_+} \top$ that corresponds to B. Applying Lemma 4.10 to the reduction derivations A and B', we can conclude that B' is standard, and hence B is standard by definition. By Theorem 4.2, the above NC-derivation is LOI. □

Example 4.3
Let $\mathscr{R}$ be an OTRS

$$\begin{cases} f(\mathrm{g}(\mathrm{d}), z) \to \mathrm{a}, \\ \mathrm{g}(\mathrm{c}) \to \mathrm{g}(\mathrm{d}). \end{cases}$$

The NC-derivation

$$f(\mathrm{g}(x), \mathrm{g}(y)) = \mathrm{a} \overset{\mathrm{n}}{\rightsquigarrow}_{\{x \mapsto \mathrm{c}\}} f(\mathrm{g}(\mathrm{d}), \mathrm{g}(y)) = \mathrm{a} \overset{\mathrm{n}}{\rightsquigarrow}_{\{y \mapsto \mathrm{c}\}} f(\mathrm{g}(\mathrm{d}), \mathrm{g}(\mathrm{d})) = \mathrm{a}$$

$$\overset{\mathrm{n}}{\rightsquigarrow}_{\{z \mapsto \mathrm{g}(\mathrm{d})\}} \mathrm{a} = \mathrm{a} \overset{\mathrm{f}}{\rightsquigarrow}_{\emptyset} \mathrm{true}$$

yields a solution $\theta = \{x \mapsto \mathrm{c}, y \mapsto \mathrm{c}\}$. This NC-derivation is not LOI, since the NC-derivation

$$\mathrm{true}, \mathrm{g}(y) = z, \mathrm{a} = \mathrm{a} \overset{\mathrm{n}}{\rightsquigarrow}_{\{y \mapsto \mathrm{c}\}} \mathrm{true}, \mathrm{g}(\mathrm{d}) = z, \mathrm{a} = \mathrm{a} \overset{\mathrm{f}}{\rightsquigarrow}_{\{z \mapsto \mathrm{g}(\mathrm{d})\}} \top$$

is not LOI. Of course, the reduction derivation

$$f(\mathrm{g}(\mathrm{c}), \mathrm{g}(\mathrm{c})) = \mathrm{a} \to f(\mathrm{g}(\mathrm{d}), \mathrm{g}(\mathrm{c})) = \mathrm{a} \to f(\mathrm{g}(\mathrm{d}), \mathrm{g}(\mathrm{d})) = \mathrm{a} \to \mathrm{a} = \mathrm{a} \to \mathrm{true}$$

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

is not standard, since the position 1.2 in $f(g(\mathrm{d}), g(\mathrm{c})) = \mathrm{a}$ for the reduction derivation starting from this equation is not external.

The LOI NC-derivation that yields a solution $\sigma = \{x \mapsto \mathrm{c}\} \preceq \theta$ is the following:

$$f(g(x), g(y)) = \mathrm{a} \overset{\mathrm{n}}{\rightsquigarrow}_{\{x \mapsto \mathrm{c}\}} f(g(\mathrm{d}), g(y)) = \mathrm{a} \overset{\mathrm{n}}{\rightsquigarrow}_{\{z \mapsto g(y)\}} \mathrm{a} = \mathrm{a} \overset{\mathrm{f}}{\rightsquigarrow}_{\emptyset} \mathrm{true}.$$

The above NC-derivation is not the only LOI NC-derivation. The following NC-derivation

$$f(g(x), g(y)) = \mathrm{a} \overset{\mathrm{n}}{\rightsquigarrow}_{\{x \mapsto \mathrm{d}, z \mapsto g(y)\}} \mathrm{a} = \mathrm{a} \overset{\mathrm{f}}{\rightsquigarrow}_{\emptyset} \mathrm{true}$$

that yields a solution $\{x \mapsto \mathrm{d}\}$ is also LOI.

To obtain the solution $\{x \mapsto \mathrm{c}\}$, narrowing of $g(x)$ is needed. However, to obtain a solution $\{x \mapsto \mathrm{d}\}$, narrowing of $g(x)$ is not needed.

So needed-ness of a narrowing step depends on the rewrite rule to be applied, and hence on the solution to be computed. Antoy *et al.* (1994) observed this, and defined a notion of needed narrowing with respect to the computed solution via Huet and Lévy's notion of needed redex.

4.6 *Completeness of* LOI **NC**

We now show that LOI narrowing is complete with respect to normalizable solutions for OTRSs.

The completeness of narrowing is formally stated as follows:

Definition 4.16
Let $\mathscr{R}$ be an arbitrary TRS and G a goal.

- A substitution σ is called a correct answer substitution of G if there exists a reduction derivation $\sigma G \twoheadrightarrow_{\mathscr{R}^+} \top$.
- **NC** is said to be complete with respect to a certain class of correct answer substitutions of a goal G for a TRS $\mathscr{R}$ if for every correct answer substitution σ of G in that class, there exists a successful NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ such that $\theta \preceq_{\mathscr{R}} \sigma[\mathscr{V}(G)]$. Here, $\preceq_{\mathscr{R}}$ is defined as follows: let θ_1 and θ_2 be substitutions. $\theta_1 =_{\mathscr{R}} \theta_2$ if $\theta_1 x =_{\mathscr{R}} \theta_2 x$ for all $x \in \mathscr{V}$, $\theta_1 \preceq_{\mathscr{R}} \theta_2$ if $\rho\theta_1 =_{\mathscr{R}} \theta_2$ for some substitution ρ, and $\theta_1 \preceq_{\mathscr{R}} \theta_2[V]$ if $\rho\theta_1 =_{\mathscr{R}} \theta_2[V]$ for some substitution ρ.
- In particular, let $\mathscr{R}$ be an OTRS and G be a proper goal. LOI **NC** is said to be complete for a certain class of correct answer substitutions if for every correct answer substitution σ of G in that class, there exists an LOI NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ such that $\theta \preceq_{\mathscr{R}} \sigma[\mathscr{V}(G)]$.

Using the following well-known completeness theorem of narrowing for confluent TRSs (Hullot, 1980; Middeldorp and Hamoen, 1994)[11], we can obtain the completeness result of LOI narrowing.

[11] Theorem 4.4 is given in Middeldorp and Hamoen (1994). Hullot first proved the completeness of NC for complete (confluent and strongly normalizing) TRSs. It is straightforward to extend the result for confluent TRSs, since Hullot's proof relies on the lifting lemma.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Theorem 4.4 (Completeness of **NC***)*
NC is complete with respect to normalizable correct answer substitutions of an arbitrary goal for a confluent TRS.

Theorem 4.4 can be proved easily using lifting lemma 4.9.

Theorem 4.5 (Completeness of LOI **NC***)*
LOI **NC** is complete with respect to normalizable correct answer substitutions of right-normal goals for OTRSs.

Proof
For every normalizable correct answer substitution σ of a right-normal goal G, there exists an NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ such that $\theta \preceq_{\mathscr{R}} \sigma[\mathscr{V}(G)]$ by Theorem 4.4. By Theorem 4.3 there exists an LOI NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta'} \top$ such that $\theta' \preceq \theta[\mathscr{V}(G)]$. It is easy to see that $\theta' \preceq_{\mathscr{R}} \sigma[\mathscr{V}(G)]$. □

5 Calculus OINC

In this section we present a calculus **OINC** that generates LOI NC-derivations. In the calculus **OINC** the inference rules [n] and [f] of **NC** are decomposed into several more primitive inference rules. Furthermore, a computation rule by which to locate narexes is built-in in **OINC**. Standardization Theorem 4.3 for NC-derivations allows us to deal only with LOI NC-derivations for the completeness of **OINC**. Hence in **OINC**, all the goals are proper and the equations in the goals are solved from left to right.

5.1 Definition of OINC

As in **NC**, we give the calculus **OINC** in the form of an inference system.

*Definition 5.1 (***OINC***)*
Let $\mathscr{R}$ be an OTRS. A calculus **OINC** for $\mathscr{R}$ is a pair $(\mathscr{G}, \mathrm{OINC})$, where

- $\mathscr{G}$ is a set of proper goals, and
- OINC is a set of the following inference rules.

— [on] outermost narrowing

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

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

— [d] decomposition

$$\frac{f(s_1,\ldots,s_n) = f(t_1,\ldots,t_n), E}{s_1 = t_1,\ldots,s_n = t_n, E}$$

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

— [v] variable elimination

– [v1]

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

– [v2]

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

Note:

- There exists indeterminacy between the choice of [on] and [d]. We will see an example of this in Example 5.1.
- We do not need an inference rule

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

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

since a narrowable term is never generated on the right-hand side of the equations in a goal by Proposition 4.2.

- The term true is never generated in the inference steps of **OINC**. Hence, unless true's are in an initial goal, true's do not appear in the goals. Since true's are superfluous in the calculus **OINC**, we remove true's in initial goals and assume that all goals (including initial goals) of **OINC** do not contain true's.
- By Proposition 4.2 we have $x \notin \mathscr{V}(t)$ in [v1] and [v2]. Hence, the so-called occur check (i.e. the check of $x \notin \mathscr{V}(t)$) is unnecessary in applying the inference rules [v1] and [v2].

An equation of the form $t = x$ is always processed by the inference rule [v1]. In our context, this is the meaning of lazy narrowing. In some implementation of **OINC**, 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 may also be regarded the essence of the lazy narrowing.

Relations over goals $\overset{\text{on}}{\leadsto}$, $\overset{\text{d}}{\leadsto}$, $\overset{\text{v1}}{\leadsto}$, $\overset{\text{v2}}{\leadsto}$, $\overset{\text{OI}}{\leadsto}$ and their reflexive and transitive closures are defined as in the calculus **NC**. The relations $\overset{\text{v1}}{\leadsto}$, $\overset{\text{v2}}{\leadsto}$, $\overset{\text{OI}}{\leadsto}$ and their reflexive and transitive closures may be subscripted by the substitutions that are formed in the inference steps.

We should note the correspondence between $\overset{\text{on}}{\leadsto}$, $\overset{\text{d}}{\leadsto}$, $\overset{\text{v1}}{\leadsto}$, $\overset{\text{v2}}{\leadsto}$ and $\overset{\text{OI}}{\leadsto}$, and $\overset{\text{on}}{\rhd}$, $\overset{\text{d}}{\rhd}$, $\overset{\text{v1}}{\rhd}$, $\overset{\text{v2}}{\rhd}$, and $\overset{\text{OI}}{\rhd}$, respectively. By the definition of the relation $\overset{\text{OI}}{\leadsto}$, it is clear that if G is proper and $G \overset{\text{OI}}{\leadsto} G'$, then G' is proper. An OINC-derivation $G_0 \overset{\text{OI}}{\leadsto}_{\theta_1} G_1 \overset{\text{OI}}{\leadsto}_{\theta_2} \ldots \overset{\text{OI}}{\leadsto}_{\theta_k} \square$ is defined as in **NC**. The OINC-derivation $G_0 \overset{\text{OI}}{\leadsto}_{\theta_1} G_1 \overset{\text{OI}}{\leadsto}_{\theta_2} \ldots \overset{\text{OI}}{\leadsto}_{\theta_k} \square$ yields a solution $(\theta_k \cdots \theta_2 \theta_1) \restriction_{\mathscr{V}(G_0)}$ of G_0.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Example 5.1
We use the OTRS in Example 4.3. The following are OINC-derivations that yield solutions $\{x \mapsto \mathrm{c}\}$ and $\{x \mapsto \mathrm{d}\}$, respectively:

$$
\begin{array}{lll}
f(g(x), g(y)) = \mathrm{a} & \overset{\mathrm{on}}{\rightsquigarrow} & g(x) = g(\mathrm{d}), g(y) = z, \mathrm{a} = \mathrm{a} \\
& \overset{\mathrm{on}}{\rightsquigarrow} & x = \mathrm{c}, g(\mathrm{d}) = g(\mathrm{d}), g(y) = z, \mathrm{a} = \mathrm{a} \\
& \overset{\mathrm{v2}}{\rightsquigarrow}_{\{x \mapsto \mathrm{c}\}} & g(\mathrm{d}) = g(\mathrm{d}), g(y) = z, \mathrm{a} = \mathrm{a} \\
& \overset{\mathrm{d}}{\rightsquigarrow} & \mathrm{d} = \mathrm{d}, g(y) = z, \mathrm{a} = \mathrm{a} \\
& \overset{\mathrm{d}}{\rightsquigarrow} & g(y) = z, \mathrm{a} = \mathrm{a} \\
& \overset{\mathrm{v1}}{\rightsquigarrow}_{\{z \mapsto g(y)\}} & \mathrm{a} = \mathrm{a} \\
& \overset{\mathrm{d}}{\rightsquigarrow} & \square. \\
f(g(x), g(y)) = \mathrm{a} & \overset{\mathrm{on}}{\rightsquigarrow} & g(x) = g(\mathrm{d}), g(y) = z, \mathrm{a} = \mathrm{a} \\
& \overset{\mathrm{d}}{\rightsquigarrow} & x = \mathrm{d}, g(y) = z, \mathrm{a} = \mathrm{a} \\
& \overset{\mathrm{v2}}{\rightsquigarrow}_{\{x \mapsto \mathrm{d}\}} & g(y) = z, \mathrm{a} = \mathrm{a} \\
& \overset{\mathrm{v1}}{\rightsquigarrow}_{\{z \mapsto g(y)\}} & \mathrm{a} = \mathrm{a} \\
& \overset{\mathrm{d}}{\rightsquigarrow} & \square.
\end{array}
$$

Before we proceed further with **OINC**, we make sure that **OINC** computes a correct solution.

Proposition 5.1 (Soundness of OINC*)*
Let $\mathscr{R}$ be an OTRS, and G be a proper goal. A solution $\theta\restriction_{\mathscr{V}(G)}$ of G yielded by an OINC-derivation $G \overset{\mathrm{OI}}{\rightsquigarrow}_{\theta} \square$ is a correct answer substitution of G.

Proof
By the induction on the length of the OINC-derivation. □

5.2 *Completeness of* OINC

The calculus **OINC** is a complete implementation of **NC**. The completeness of **OINC** states that given a right-normal goal G, for any successful NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ there exists an OINC-derivation

$$G \overset{\mathrm{OI}}{\rightsquigarrow}_{\sigma} \square, \text{ such that } \sigma \preceq \theta[\mathscr{V}(G)].$$

The solution $\theta\restriction_{\mathscr{V}(G)}$ is non-narrowable (hence normalized) by Proposition 4.3.

The proof of the completeness of **OINC** proceeds as follows:

1. We have already seen that for any successful NC-derivation there exists an LOI NC-derivation.
2. We transform an LOI NC-derivation A to another LOI NC-derivation A' that is simpler than A.
3. We connect A and A' by an inference step of **OINC**.
4. By an inductive argument, we show that A is replaced by an OINC-derivation.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

This proof method was employed by Hölldobler in proving the completeness of an inference system TRANS (Hölldobler, 1989).

The key to the completeness proof is the following lemma:

Lemma 5.1

Let $\mathscr{R}$ be an OTRS, and G be a proper goal. If there exists an LOI NC-derivation $G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$, then there exists an OINC-derivation $\overline{G} \overset{\mathrm{OI}}{\rightsquigarrow}_{\sigma} \Box$, such that $\sigma \sim \theta$, where $\overline{G}$ is a goal obtained from G by removing true's in G.

Proof

By the induction on $\overset{\mathrm{OI}}{\rhd}$. Obviously, the result holds for the base case. Let G be a proper goal $\top, s = t, E$ and $A : G \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top$ be an LOI NC-derivation. Assume that the result holds for any NC-derivation B such that $A \overset{\mathrm{OI}}{\rhd} B$. We distinguish the following four cases:

(1) t is a variable x.
A is written as

$$A : \top, s = x, E \overset{\mathrm{f}}{\rightsquigarrow}_{\theta_1(=\{x \mapsto s\})} \top, \theta_1 E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta'} \top.$$

Note that the term s does not contain a variable x by the property (G1). There exists an LOI NC-derivation $B : \theta_1 E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta'} \top$ such that $A \overset{\mathrm{OI}}{\rhd} B$. By the induction hypothesis, there exists an OINC-derivation

$$\theta_1 \overline{E} \overset{\mathrm{OI}}{\rightsquigarrow}_{\sigma'} \Box,$$

such that $\sigma' \sim \theta'$. By the inference rule [v1] of **OINC**, we have an OINC-derivation

$$s = x, \overline{E} \overset{\mathrm{v1}}{\rightsquigarrow}_{\theta_1} \theta_1 \overline{E}.$$

Therefore, we have an OINC-derivation

$$s = x, \overline{E} \overset{\mathrm{v1}}{\rightsquigarrow}_{\theta_1} \theta_1 \overline{E} \overset{\mathrm{OI}}{\rightsquigarrow}_{\sigma'} \Box.$$

Here, we have $\sigma = \sigma' \theta_1 \sim \theta' \theta_1 = \theta$.

(2) t is a non-variable term.

(2-1) s is a variable x.
A is written as

$$A : \top, x = t, E \overset{\mathrm{f}}{\rightsquigarrow}_{\theta_1(=\{x \mapsto t\})} \top, \theta_1 E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta'} \top,$$

since t is not narrowable by the property (G2), and t does not contain x by the property (G1). The rest of the proof is the same as that of the case (1).

(2-2) s is a term $f(s_1, \ldots, s_n)$.
We further distinguish the following two cases:

(2-2a) A is written as

$$A : \top, f(s_1, \ldots, s_n) = t, E \overset{\mathrm{NC}}{\rightsquigarrow}_{\theta} \top,$$

where a descendant of the equation $f(s_1, \ldots, s_n) = t$ is narrowed at the position 1 with a new variant $f(l_1, \ldots, l_n) \to r$ of a rewrite rule in $\mathscr{R}$.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

By Lemma 4.7, there exists an LOI NC-derivation

$$B : s_1 = l_1, \ldots, s_n = l_n, r = t, E \overset{\text{NC}}{\rightsquigarrow}_{\theta'} \top,$$

such that $\theta' \sim \theta$ and $A \overset{\text{OI}}{\rhd} B$. By the induction hypothesis and the inference rule [on], there exists an OINC-derivation

$$f(s_1, \ldots, s_n) = t, \overline{E} \overset{\text{on}}{\rightsquigarrow} s_1 = l_1, \ldots, s_n = l_n, r = t, \overline{E} \overset{\text{OI}}{\rightsquigarrow}_{\sigma'} \Box,$$

such that $\sigma' \sim \theta'$.

(2-2b) A is written as

$$A : \top, f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E \overset{\text{NC}}{\rightsquigarrow} \top.$$

where the descendants of the equation $f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n)$ are not narrowed at the position 1 or 2.

By Lemma 4.6, there exists an LOI NC-derivation

$$B : s_1 = t_1, \ldots, s_n = t_n, E \overset{\text{NC}}{\rightsquigarrow}_{\theta'} \top,$$

such that $\theta' \sim \theta$ and $A \overset{\text{OI}}{\rhd} B$. Hence, we have an OINC-derivation

$$f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), \overline{E} \overset{\text{d}}{\rightsquigarrow} s_1 = t_1, \ldots, s_n = t_n, \overline{E} \overset{\text{OI}}{\rightsquigarrow}_{\sigma'} \Box$$

such that $\sigma' \sim \theta'$ by the induction hypothesis.

□

Theorem 5.1 (Completeness of **OINC***)*
OINC is complete with respect to normalizable correct answer substitutions of right-normal goals for OTRSs.

Proof
Similar to the proof of Theorem 4.5. Use Theorems 4.3 and 4.4 and Lemma 5.1. □

Moreover, from Proposition 4.3, Theorem 4.3 and Theorem 5.1, we can see that the solutions obtained by **OINC** is non-narrowable.

6 Calculus s-OINC

In section 4 we restricted ourselves to solving right-normal goals, and in section 5.2 we have obtained the completeness result of **OINC** for right-normal goals. Furthermore, we can obtain all the normalized solutions of a right-normal goal.

Readers might wonder whether right-normal goals are too restrictive. This restriction is justified from the programming language point of view. In functional programming we are interested in computing normal forms, in particular constructor terms, since they are the values we want to obtain as the result of computation. For example, by giving a functional program 1+2+3, we want to compute a value 6, not 3+2+1 nor 3+3. In functional-logic programming, the computation is equation solving rather than reduction. Nevertheless, we are still interested in obtaining values (i.e. constructor terms) which we can get in the form of constructor term substitutions.[12] One way to achieve this is to restrict the goals as sequences of strict

[12] A substitution is called a constructor term substitution if its codomain contains only constructor terms.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

equations. As Proposition 4.3 shows, the solutions of right-normal goals by **OINC** are normalized for OTRSs. Furthermore, if the OTRSs are constructor-based, the solutions of strict equations are constructor term substitutions.

6.1 Strict equations

We will define a strict equality as a function whose meaning is given by rewrite rules.

Definition 6.1

- A strict equality $\equiv$ is a function[13] defined as follows:

$$s \equiv t = \begin{cases} \text{true} & \text{if } s,t \in \mathcal{T}(\mathcal{F}_{\mathscr{C}},\emptyset), \text{ and } s \text{ and } t \text{ are the same terms,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

- A strict equation is an equation of the form $s \equiv t = \text{true}$, where s and t are arbitrary terms.

A strict equation is obviously right-normal.

To cope with the strict equations, we extend a TRS $\mathscr{R}$ with rewrite rules that define the strict equality for each constructor symbol $c \in \mathcal{F}_{\mathscr{C}}$,

$$\mathscr{R}_c = \begin{cases} \{c \equiv c \to \text{true}\} \\ \qquad \text{if the arity of } c \text{ is } 0, \\ \{c(x_1,\ldots,x_n) \equiv c(y_1,\ldots,y_n) \to x_1 \equiv y_1 \wedge \ldots \wedge x_n \equiv y_n\} \\ \qquad \text{if the arity of } c \text{ is } n > 0, \end{cases}$$

and with the rewrite rule: $\text{true} \wedge x \to x$, where $\wedge$ is a right-associative infix function symbol. Let $\mathscr{R}^{\equiv} = \mathscr{R} \cup \bigcup_{c \in \mathcal{F}_{\mathscr{C}}} \mathscr{R}_c \cup \{\text{true} \wedge x \to x\}$. The calculus **OINC** and its completeness result remain valid for $\mathscr{R}^{\equiv}$ since $\mathscr{R}^{\equiv}$ is an OTRS.

Furthermore, abbreviating an equation $s \equiv t = \text{true}$ in a goal as $s \equiv t$, we can treat $s \equiv t$ as if it were an ordinary (non-strict) equation. Solving a strict equation $s \equiv t$ with **OINC**, we can obtain a normalized substitution θ such that θs and θt have a common reduct that is a constructor term.

The idea of using a strict equation in functional-logic programming languages originates in a logic plus functional language K-LEAF (Giovannetti *et al.*, 1991) and has been exploited by several researchers (Antoy *et al.*, 1994; Loogen *et al.*, 1993; Narain, 1986).

6.2 Extension of OINC

To handle strict equations efficiently, we next extend **OINC**. To motivate the extension let us take an example.

[13] We use $\equiv$ in the infix form to denote the strict equality. The symbol $\equiv$ is used in two ways in this paper; to denote the syntactic equality over terms and the strict equality. The distinction should be clear from the context.

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Example 6.1
Let $\mathscr{R}$ be a TRS given in Example 3.1. With respect to $\mathscr{R}^{\equiv}$, we solve a goal $f(g(x)) \equiv f(y)$ in **OINC**. Let $\mathscr{R}$ be $\{f(w) \to w, g(1) \to 1\}$. We solve the following goal, with respect to $\mathscr{R}^{\equiv}$.

$$A_1 : f(g(x)) \equiv f(y) \overset{\text{on}}{\rightsquigarrow} f(g(x)) = 1, f(y) = 1, \text{true} = \text{true}$$
$$\overset{\text{on}}{\rightsquigarrow} g(x) = z_1, z_1 = 1, f(y) = 1, \text{true} = \text{true}$$
$$\overset{\text{v1}}{\rightsquigarrow}_{\{z_1 \mapsto g(x)\}} g(x) = 1, f(y) = 1, \text{true} = \text{true} \overset{\text{OI}}{\rightsquigarrow\!\!\!\rightarrow}_{\{x \mapsto 1, z_2 \mapsto 1, y \mapsto 1\}} \square.$$

This OINC-derivation appears to be very redundant. With some insight we can think of an inference step that can bypass some of the above steps. Let us try to apply a kind of $\overset{\text{on}}{\rightsquigarrow}$-step on the strict equation directly.

$$f(g(x)) \equiv f(y) \rightsquigarrow g(x) = z, z \equiv f(y).$$

We denote this step by $\overset{\text{ons}}{\rightsquigarrow}$. Then we can obtain a new derivation:

$$A_2 : f(g(x)) \equiv f(y) \overset{\text{ons}}{\rightsquigarrow} g(x) = z_1, z_1 \equiv f(y) \overset{\text{v1}}{\rightsquigarrow}_{\{z_1 \mapsto g(x)\}} g(x) \equiv f(y)$$
$$\overset{\text{OI}}{\rightsquigarrow\!\!\!\rightarrow}_{\{x \mapsto 1, \ldots\}} 1 \equiv f(y) \overset{\text{ons}}{\rightsquigarrow} y = z_2, 1 \equiv z_2 \rightsquigarrow \cdots \rightsquigarrow \square.$$

We see that in the derivation A_2 the equation $g(x) = z_1$ is generated in one step, whereas in the derivation A_1 it is generated in two $\overset{\text{on}}{\rightsquigarrow}$-steps.

The problem with **OINC** in handling the strict equations is not only the number of redundant steps, but the difficulty of choosing an adequate rewrite rule for strict equations when there are many constructor symbols $c \in \mathscr{F}_{\mathscr{C}}$. In the above example, in the derivation A_1 we select the rewrite rule $1 \equiv 1 \to \text{true}$ immediately in the first step of the derivation. In practice, this is impossible without trials-and-backtracks.

We will circumvent these difficulties in the following way. A basic idea for taking the shortcut that we saw above is to narrow the left- and right-hands of the strict equations independently. Suppose that a goal $s \equiv t$ is given. We narrow s and t independently until s and t become constructor terms, say $c(s')$ and $c(t')$, respectively (if t becomes $c'(t')$ where $c \not\equiv c'$, this derivation will never become successful). Then we repeat this process with s' and t'.

We are now ready to give the inference rules for strict equations.

6.3 Inference-rules for strict equations

- [ons] outermost narrowing for strict equations

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

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

- [ds] decomposition for strict equations

$$\frac{c(s_1, \ldots, s_n) \equiv c(t_1, \ldots, t_n), E}{s_1 \equiv t_1, \ldots, s_n \equiv t_n, E} \quad c \in \mathscr{F}_{\mathscr{C}}$$

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

- [ims] imitation for strict equations

$$\frac{c(s_1,\ldots,s_n) \equiv y, E}{\theta(s_1 \equiv y_1,\ldots,s_n \equiv y_n, E)} \quad c \in \mathscr{F}_{\mathscr{C}} \quad \text{and}$$

$$\frac{y \equiv c(t_1,\ldots,t_n), E}{\theta(y_1 \equiv t_1,\ldots,y_n \equiv t_n, E)} \quad c \in \mathscr{F}_{\mathscr{C}}$$

where $\theta = \{y \mapsto c(y_1,\ldots,y_n)\}$.

- [ts] elimination of trivial strict equations

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

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

Since the strict equations are symmetrical in narrowing, we need two rules in the inference rules [ons] and [ims]. Note also that thanks to the inference rules for the strict equations we no longer need the set of rewrite rules for the strict equations.

Relations $\overset{\text{ons}}{\rightsquigarrow}$, $\overset{\text{ds}}{\rightsquigarrow}$, $\overset{\text{ims}}{\rightsquigarrow}$, and $\overset{\text{ts}}{\rightsquigarrow}$ are defined as in the calculus **OINC**. Let $\overset{\text{s-OI}}{\rightsquigarrow} = \overset{\text{OI}}{\rightsquigarrow} \cup \overset{\text{ons}}{\rightsquigarrow} \cup \overset{\text{ds}}{\rightsquigarrow} \cup \overset{\text{ims}}{\rightsquigarrow} \cup \overset{\text{ts}}{\rightsquigarrow}$. Except for the inference rule [ims], the new inference rules are easy to understand. The inference rule [ims] is used to narrow the subterms of a head-constructor term when the other side of the strict equation is a variable. Let us take an example.

Example 6.2
Let $\mathscr{F}_{\mathscr{C}} = \{1, \mathrm{c}_1\}$, where c_1 is a constructor symbol of arity one. We use the TRS $\mathscr{R}$ of Example 3.1, and solve a goal $G \triangleq \mathrm{c}_1(f(x)) \equiv y$. A successful derivation is as follows.

$$G \overset{\text{ims}}{\rightsquigarrow}_{\{y \mapsto \mathrm{c}_1(y_1)\}} f(x) \equiv y_1 \overset{\text{ons}}{\rightsquigarrow} x = z, z \equiv y_1 \overset{\text{v1}}{\rightsquigarrow}_{\{z \mapsto x\}} x \equiv y_1 \overset{\text{ts}}{\rightsquigarrow}_{\{x \mapsto y_1\}} \Box.$$

The solution obtained in this derivation is $\{y \mapsto \mathrm{c}_1(y_1), x \mapsto y_1\}$.

Let s-OINC = {[ons], [ds], [ims], [ts]}$\cup$ OINC. We define a new calculus **s-OINC** = $(\mathscr{G}, \text{s-OINC})$, where $\mathscr{G}$ is a set of proper goals as in **OINC**. The initial goal is a sequence of a right-normal equations, some of which may be a strict equation.

6.4 *Soundness and completeness of* s-OINC

The soundness of **s-OINC** restricted to constructor ground instances is easy to prove. The completeness theorem for **s-OINC** is obtained from the following proposition:

Proposition 6.1
Let $\mathscr{R}$ be an OTRS, and G be a proper goal. If there exists an OINC-derivation with respect to $\mathscr{R}^{\equiv}$

$$G \overset{\text{OI}}{\rightsquigarrow}_{\theta} \Box$$

then there exists an s-OINC-derivation with respect to $\mathscr{R}$

$$G \overset{\text{s-OI}}{\rightsquigarrow}_{\sigma} \Box, \text{ such that } \sigma \preceq \theta[\mathscr{V}(G)].$$

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

Proof
By the induction on the length of the OINC-derivation. The structure of the proof is similar to the proof of Lemma 5.1 used to prove the completeness of **OINC**. □

Since the completeness of **OINC** is already established, the completeness of **s-OINC** follows immediately from Proposition 6.1.

Theorem 6.1 (Completeness of s-OINC*)*
s-OINC is complete with respect to normalizable correct answer substitutions of right-normal goals for OTRSs.

7 Conclusion

We have presented leftmost outside-in narrowing and the narrowing calculus **OINC** based on the leftmost outside-in narrowing. The calculus **OINC** realizes lazy evaluation in narrowing in that it delays narrowing on narrowable terms that are to be bound to variables. Furthermore, it enjoys the property of completeness for orthogonal TRSs with respect to normalizable answers.

To use the calculus **OINC** as a model of computation for functional-logic programming we extended **OINC** to incorporate strict equality. The extension results in a new narrowing calculus **s-OINC**. It has been also shown that the calculus **s-OINC** enjoys the same completeness property as **OINC**. The calculus **s-OINC** was used for the design and implementation of a higher-order functional-logic programming language based on applicative term rewriting systems (Hamana *et al.*, 1995).

Our completeness results are restricted to orthogonal TRSs. The orthogonality is used critically in the definition of proper goals and in the standardization theorem of Huet and Lévy. Recently, more abstract treatment of the standardization theorem was reported, and the standardization theorem has been extended to some class of ambiguous TRSs (Gonthier *et al.*, 1992). One possible extension of our results would be to a class of TRSs for which the standardization theorem holds.

A Switching lemma

The switching lemma presented without a proof in this appendix is used in the proof of Lemma 4.5.

Lemma A1 [Switching lemma]
Let $\mathscr{R}$ be an arbitrary TRS, $G \triangleq E_1, e_1, E_2, e_2, E_3$ be a goal, and u_1 and u_2 are positions such that $u_1 \in \overline{\mathscr{O}}(e_1)$ and $u_2 \in \overline{\mathscr{O}}(e_2)$. If there exists an NC-derivation

$$\begin{array}{lll} A : G & \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_1} & (G_1 \triangleq)\ \theta_1(E_1, e_1[r_1]_{u_1}, E_2, e_2, E_3) \\ & \overset{\mathrm{n}}{\rightsquigarrow}_{\theta_2} & (G_2 \triangleq)\ \theta_2\theta_1(E_1, e_1[r_1]_{u_1}, E_2, e_2[r_2]_{u_2}, E_3) \\ & \overset{\mathrm{NC}}{\rightsquigarrow\!\!\!\!\rightarrow}_{\theta'} & \top, \end{array}$$

where a new variant $l_1 \to r_1$ of a rewrite rule in $\mathscr{R}$ and a new variant $l_2 \to r_2$ of a rewrite rule in $\mathscr{R}$ are employed in the first and second steps, respectively,

https://doi.org/10.1017/S0956796897002645 Published online by Cambridge University Press

then there exists an NC-derivation

$$\begin{array}{lll} A' : G & \overset{\mathrm{n}}{\rightsquigarrow}_{\sigma_1} & (G'_1 \triangleq)\ \sigma_1(E_1, e_1, E_2, e_2[r_2]_{u_2}, E_3) \\ & \overset{\mathrm{n}}{\rightsquigarrow}_{\sigma_2} & (G'_2 \triangleq)\ \sigma_2\sigma_1(E_1, e_1[r_1]_{u_1}, E_2, e_2[r_2]_{u_2}, E_3) \\ & \overset{\mathrm{NC}}{\rightsquigarrow\!\!\!\rightsquigarrow}_{\sigma'} & \top, \end{array}$$

such that $\theta'\theta_2\theta_1 \sim \sigma'\sigma_2\sigma_1$,

where the rewrite rules $l_2 \to r_2$ and $l_1 \to r_1$ are employed in the first and second steps, respectively.

This result is applicable to a TRS that contains a rewrite rule $x = x \to \text{true}$. Hence, we can consider $\mathscr{R}_+$, in which the rewrite rule $x = x \to \text{true}$ is applied to the root position of an equation. Therefore, this switching lemma also implies the following:

- Adjacent $\overset{\mathrm{n}}{\rightsquigarrow}$-step and $\overset{\mathrm{f}}{\rightsquigarrow}$-step (in this order) contracting different equations can be switched. Note, however, that adjacent $\overset{\mathrm{f}}{\rightsquigarrow}$-step and $\overset{\mathrm{n}}{\rightsquigarrow}$-step (in this order) cannot be switched in general.
- Adjacent $\overset{\mathrm{f}}{\rightsquigarrow}$-steps can be switched.

Acknowledegments

We thank Aart Middeldorp, Toshiyuki Yamada and the referees for many valuable suggestions of improvements.

This research was supported in part by Grant-in-Aid for Scientific Research (C) 06680300 of the Ministry of Education, Science, Sports and Culture of Japan.

References

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

Bockmayr, A., Krischer, S. and Werner, A. (1992) An optimal narrowing strategy for general canonical systems. *Proceedings 3rd International Workshop on Conditional Term Rewriting Systems*, pp. 483–497. *Lecture Notes in Computer Science 656*. Springer-Verlag.

Darlington, J. and Guo, Y.-K. (1989) Narrowing and unification in functional programming – an evaluation mechanism for absolute set abstraction. *Proceedings Rewriting Techniques and Applications*, pp. 92–108. *Lecture Notes in Computer Science 355*. Springer-Verlag.

Dershowitz, N. and Jouannaud, J.-P. (1990) Rewrite systems. In: van Leeuwen, J. (ed), *Handbook of Theoretical Computer Science*, vol. B, pp. 243–320. MIT Press/Elsevier.

Echahed, R. (1988) On completeness of narrowing strategies. *Proceedings 13th Colloquium on Trees in Algebra and Programming*, pp. 89–101. *Lecture Notes in Computer Science 299*. Springer-Verlag.

Eder, E. (1985) Properties of substitutions and unifications. *Journal of Symbolic Computation*, **1**, 31–46.

Fay, M. (1979) First-order unification in equational theories. *Proceedings 4th International Workshop on Automated Deduction, Austin*, pp. 161–167.

Friborg, L. (1985) SLOG: a logic programming language interpreter based on clausal superposition and rewriting. *Proceedings 2nd IEEE Symposium on Logic Programming, Boston*, pp. 172–184.

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

Gonthier, G., Lévy, J.-J. and Melliès, P.-A. (1992) An abstract standardisation theorem. *Proceedings of the 7th Annual Symposium on Logic in Computer Science*, pp. 72–81.

Hamana, M., Nishioka, T., Nakahara, K., Middeldorp, A. and Ida, T. (1995) A design and implementation of a functional-logic language based on applicative term rewriting systems (in Japanese). *Transactions of Information Processing Society of Japan*, **30**(8), pp. 1897–1995.

Hanus, M. (1990) Compiling logic programs with equality. *Proceedings 2nd International Symposium on Programming Language Implementation and Logic Programming*, pp. 387–401. *Lecture Notes in Computer Science 456*. Springer-Verlag.

Hölldobler, S. (1989) Foundations of equational logic programming. *Lecture Notes in Artificial Intelligence*, **353**.

Huet, G. and Lévy, J.-J. (1991) Computations in orthogonal rewriting systems, I. Lassez, J.-L. and Plotkin, G. (eds), *Computational Logic: Essays in honor of Alan Robinson.* MIT Press, pp. 395–414.

Hullot, J. (1980) Canonical forms and unification. *Proceedings 5th Conference on Automated Deduction*, pp. 318–334. *Lecture Notes in Computer Science 87*. Springer-Verlag.

Klop, J. W. (1992) Term rewriting systems. Abramsky, S., Gabbay, D. and Maibaum, T. (eds), *Handbook of Logic in Computer Science*, vol. 2. Oxford University Press, pp. 1–116.

Lock, H. C. R. (1992) *The implementation of functional logic programming languages.* PhD thesis, Universität Karlsruhe. Published as GMD-Bericht Nr. 208, by R. Oldenbourg Verlag, 1993.

Loogen, R., Fraguas, F. L. and Rodríguez-Artalejo, M. (1993) A demand driven computation strategy for lazy narrowing. *Proceedings 5th International Symposium on Programming Language Implementation and Logic Programming*, pp. 184–200. *Lecture Notes in Computer Science 714*. Springer-Verlag.

Middeldorp, A. and Hamoen, E. (1994) Completeness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computing*, **5**(3/4), 213–253.

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

Narain, S. (1986) A technique for doing lazy evaluation in logic. *Journal of Logic Programming*, **3**, 259–276.

Reddy, U. S. (1985) Narrowing as the operational semantics of functional languages. *Proceedings of IEEE International Symposium on Logic Programming*, pp. 138–151.

Slagle, J. R. (1974) Automatic theorem proving in theories with simplifiers, commutativity and associativity. *Journal of the ACM*, **21**, 622–642.

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