ISE

ISE-TR-92-97

Lazy Narrowing Calculi

by

Satoshi Okui and Tetsuo Ida

February 1992

INSTITUTE
OF
INFORMATION SCIENCES AND ELECTRONICS
UNIVERSITY OF TSUKUBA

Lazy Narrowing Calculi

Satoshi Okui and Tetsuo Ida
Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba 305, Japan

Abstract

Lazy narrowing calculi are presented and examined from a viewpoint of a computation model for lazy functional-logic programming languages. First, we present a lazy narrowing calculus LNC_0, which Hölldobler originally gave for paramodulation. We recast Giovannetti and Moiso's completeness results of conditional narrowing in the presence of the extra-variables, and prove the completeness of LNC_0 with respect to the conditional narrowing. Next, we develop LNC_0 into a new lazy narrowing calculus to be called LNC_1, which can be used as a computation model for a lazy functional-logic programming language. LNC_1 is derived by imposing restrictions on a form of equations that are used in goals and equation definitions. We show, under certain conditions, the completeness of LNC_1 by relating it to the completeness of LNC_0. LNC_1 has only four inference rules, the choice of which in a refutation is determinate, hence is efficiently implemented.

Research supported in part by Grants-in-Aid for Scientific Research No.03680022 and No.03235201, the Ministry of Education, Science and Culture.
e-mail address: {okui,ida}@softlab.is.tsukuba.ac.jp

1 Introduction

Narrowing was originally proposed in the 70's to automate equational reasoning within the framework of term rewriting systems [24]. Renewed interest in narrowing came from the researches of integrating functional and logic programming languages. Several researchers noticed that narrowing can be used as a basic computing mechanism of functional-logic programs [10, 23, 12, 26, 20, 2, 17, 7, 18, 5, 13]. Basic ideas are to use equations to define a function, and then to use narrowing both for reduction and for equation solving. Unrestricted narrowing originally proposed in equation solving is not so efficient as to be a practical computation model for functional-logic programming languages.

Researches on integrating functional and logic programming languages in this direction, therefore, amount to the speed-up of narrowing. The problem can be addressed in two ways; namely, to define appropriate expressions to be used for narrowing, and to design efficient narrowing calculi on those expressions. Since we are interested in the kind of narrowing calculi that are used for computation models for programming languages, we naturally assume a certain class of expressions as programs and we also require the efficiency of the calculi.

In this conjunction Hullot's basic narrowing [16] and the technique developed by Bosco et al. of simulating the basic narrowing by the SLD-resolution [4] can be regarded as our starting point. Bosco et al. developed a language K-LEAF based on the flattening and the "outermost" SLD-resolution [2] and presented an efficient implementation of an outermost SLD-resolution by extending WAM [3]. The former makes possible significant reduction of the search space which narrowing calculi have to explore by restricting the expressions that narrowing can be applied to, and the latter obtained further reduction of the search space by restricting the equations to the class of equations that can be viewed as defining functions. In a more general theoretical framework Hölldobler developed calculi for equational and logic programming [14].

In this paper we start from a narrowing calculus to be called LNC_0 together with a language $\mathcal{L}_0$ for the LNC_0. LNC_0 consists of a set of the inference rules which is a subset of the inference rules of Hölldobler's calculus TRANS. LNC_0 is used to solve a set of conditional equations. While Hölldobler's interest is in general equational and logic programming in [14], we restrict our interest to equations defining functions. We therefore step forward to increased efficiency sacrificing the generality as equation solving. We then define another calculus LNC_1 together with a language $\mathcal{L}_1$ so that calculus LNC_1 can be used for a model of a functional-logic programming language.

We next show that narrowing calculus LNC_1 is a sound and complete calculus for equation solving, and further show that it can be used as a computation model for a functional-logic programming languages. The proofs of the soundness and completeness are based solely on transformations of expressions by inference rules without resorting to new notions such as produced variables and ordering relations of equations that are used in discussing the operational semantics of K-LEAF.

The paper is organized as follows. In section 2 we give our notations of term rewriting and logic programming. In section 3 we present the language $\mathcal{L}_0$ for the narrowing calculus LNC_0. In section 4 we present LNC_0 together with the completeness proof. The soundness of LNC_0 relies on Hölldobler's result. In section 5 we give the language $\mathcal{L}_1$ which is a restriction of the language $\mathcal{L}_0$, but is tuned for a programming language. In section 6 we give the calculus LNC_1 which is derived from LNC_0. The soundness and completeness results of LNC_1 are then given.

2 Preliminaries

We assume the reader's familiarity with term rewriting [15] and logic programming [19]. The set of variables is denoted by $V \ni x, y, z, \ldots$. The set of (first-order) *terms* (notation $\mathcal{T} \ni t, s, \ldots$) is defined as usual. $Var(o)$ is a set of variables appearing in a syntactic object o. For example, o is a term, an equation or a sequence of equations. The set of *occurrences* of t (notation $Occ(t) \ni u, v, \ldots$) is defined as usual. The *length* of an occurrence is defined by the length as the sequence of natural numbers. When the length of u is shorter than the length of v, we write $u \prec v$. An occurrence of length 0 is denoted by Λ. The *subterm* of t at an occurrence $u(\in Occ(t))$ is denoted by t/u. When t/u is not a variable, we say u is a non-variable occurrence. The set of non-variable occurrences of t is denoted by $\overline{Occ}(t)$. The *replacement* of s in t at $u(\in Occ(t))$ is denoted by $t[u \leftarrow s]$. A term t is called *linear* when no variable occurs in t more than once. Let $\rightarrow$ be any compatible relations on terms. Reflexive transitive closure of $\rightarrow$ is denoted by $\twoheadrightarrow$. Symmetric closure of $\twoheadrightarrow$ is denoted by $\leftrightarrow\!\!\!\!\twoheadrightarrow$. A *substitution* is denoted by $\theta, \sigma, \rho, \gamma \ldots$. The *empty substitution* is denoted by ε. The *domain*, *codomain* and variables in the codomain of a substitution θ are denoted by $Dom(\theta)$, $Cod(\theta)$ and $Vcod(\theta)$ respectively. The *restriction* of θ to V is denoted by $\theta{\restriction}V$. The *composition* of θ_2 and θ_1, (first apply θ_1, then θ_2) is denoted by $\theta_2\theta_1$. We write $\theta_1 \twoheadrightarrow \theta_2$ iff $\theta_1 x \twoheadrightarrow \theta_2 x$ for any variable x. θ is a *normalized* substitution (w.r.t. $\rightarrow$) iff for any $x \in Dom(\theta)$ θx is a normal form (w.r.t. $\rightarrow$). When $\theta \twoheadrightarrow \theta'$ for some normalized substitution θ', we call θ a *normalizable* substitution and call θ' a *normal form* of θ. A normal form of θ is denoted by $\theta{\downarrow}$. When $\sigma\theta_1 = \theta_2$ for some substitution σ, we write $\theta_1 \geq \theta_2$. $\geq$ is quasi-ordering. When $\theta_1 \geq \theta_2$ and $\theta_2 \geq \theta_1$ we write $\theta_1 \sim \theta_2$. A *most general unifier* (mgu) is defined as usual.

3 Language $\mathcal{L}_0$

We first present the language $\mathcal{L}_0$ for the narrowing calculus LNC_0. $\mathcal{L}_0$ is a language of first-order Horn clause logic equipped with only equality predicate symbol. The equality symbol (=) is used as an infix predicate symbol. The syntax of $\mathcal{L}_0$ is as follows:

$$
\begin{array}{rcl}
\langle \text{conditional equation} \rangle & ::= & \langle \text{head} \rangle \langle \text{body} \rangle \mid \langle \text{head} \rangle \\
\langle \text{head} \rangle & ::= & \langle \text{equation} \rangle \\
\langle \text{body} \rangle & ::= & \Leftarrow \langle \text{equation} \rangle_1, \ldots, \langle \text{equation} \rangle_n \\
 & & \text{where } n \geq 1 \\
\langle \text{goal} \rangle & ::= & \Leftarrow \langle \text{equation} \rangle_1, \ldots, \langle \text{equation} \rangle_n \\
 & & \text{where } n \geq 0 \\
\langle \text{equation} \rangle & ::= & \langle \text{term} \rangle = \langle \text{term} \rangle
\end{array}
$$

Note:

- $\langle \text{term} \rangle$ is a syntactic category for $\mathcal{T}$ (a set of terms).
- $\Leftarrow$ is a logical connective *implication* and , (comma) is a logical connective *and*.
- The syntax of variables and function symbols are left unspecified. However, as a convention in this paper, we use x, y, z and w to denote variables and f, g and h to denote function symbols.

We specify the syntax of $\mathcal{L}_1$ in a general manner, here. We will later impose syntactic restrictions on the equations in the head and body as extra syntactic rules. At this point, we naturally impose following conditions on the occurrences of variables.

An equation $t = s$ of the head satisfies the following conditions.

- t is a non-variable term.
- $Var(t) \supseteq Var(s)$.

These conditions are usually imposed when we consider $t = s$ as a rewrite rule scheme $t \rightarrow s$. More syntactic restrictions will be introduced later to enable us to regard a set of conditional equations as a certain type of conditional term rewriting systems that we desire.

A set of conditional equations is called *conditional equational system* (coined by Dershowitz and Okada [8]).

Example 1. The following two conditional equations define a function *append* which appends two lists.

$$append(nil, x) = x$$
$$append(cons(x, y), z) = cons(x, append(y, z))$$

Suppose we require the inputs to *append* be lists of integers. We may define *ints_append* as follows:

$$int\_list(nil) = true$$
$$int\_list(cons(x, y)) = true \Leftarrow int(x) = true, int\_list(y) = true$$
$$int\_append(x, y) = append(x, y) \Leftarrow int\_list(x) = true, int\_list(y) = true$$

int_list and *int* are intended to be truth-value functions. We may simply write $int\_list(\ldots)$ and $int(\ldots)$ instead of $int\_list(\ldots) = true$ and $int(\ldots) = true$ respectively.

Hereafter, t, s, l and r denote terms, G denotes a goal, E and F denote the (possibly empty) conjunction of equations.

4 Lazy narrowing calculus LNC_0

Given a conditional equational system R, the lazy narrowing calculus LNC_0 w.r.t. R is defined as a formal system consisting of the following inference rules over a set of goals.

4.1 Inference rules

The inference rules of LNC_0 are as follows.

1. Outermost narrowing [on]

$$\frac{\Leftarrow E, f(s_1, \ldots, s_n) \doteq s, E'}{\Leftarrow E, s_1 = t_1, \ldots, s_n = t_n, F, t = s, E'} \quad f(t_1, \ldots, t_n) = t \Leftarrow F$$

2. Imitation [im]

$$\frac{\Leftarrow E, f(s_1, \ldots, s_n) \doteq x, E'}{\Leftarrow \theta(E, s_1 = x_1, \ldots, s_n = x_n, E') \text{ where } \theta = \{f(x_1, \ldots, x_n)/x\}}$$

3. Variable elimination [v]

$$\frac{\Leftarrow E, x \doteq t, E'}{\Leftarrow \theta(E, E') \text{ where } \theta = \{t/x\}} \quad x \notin Var(t)$$

4. Removal of trivial equations [t]

$$\frac{\Leftarrow E, t = t, E'}{\Leftarrow E, E'}$$

5. Decomposition [d]

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

Note:

- $s \doteq t$ denotes either $s = t$ or $t = s$. Hence, inference rules [on], [im] and [v] respectively denote two rules.
- The outermost narrowing rule [on] is applicable to an equation of a goal if there exists a new variant $f(t_1, \ldots, t_n) = t \Leftarrow F$ of a conditional equation in R,
- The variable elimination rule [v] is applicable to an equation of a goal if $x \notin Var(t)$.
- LNC_0 is a subset of TRANS (complete set of transformation rules) of Hölldobler [14].

Abusing the notation, we use LNC_0 to denote the set of inference rules of LNC_0. Let G and G' be the premise and the conclusion of the rule $[\alpha] \in LNC_0$ respectively and θ be the substitution formed in the inference step. We write $G \overset{\theta}{\rightarrow}_{[\alpha]} G'$. When $[\alpha] \in \{[\text{on}], [\text{t}], [\text{d}]\}$, θ is the empty substitution ε. θ is sometimes omitted. An n-step *derivation* of G_0 and R w.r.t. LNC_0 is a sequence

$$G_0 \overset{\theta_1}{\rightarrow}_{[\alpha_1]} G_1 \overset{\theta_2}{\rightarrow}_{[\alpha_2]} \cdots \overset{\theta_{n-1}}{\rightarrow}_{[\alpha_{n-1}]} G_{n-1} \overset{\theta_n}{\rightarrow}_{[\alpha_n]} G_n$$

where $[\alpha_1], \ldots, [\alpha_n] \in LNC_0$ and is written as $G_0 \overset{\theta}{\twoheadrightarrow}_{LNC_0} G_n$ where $\theta = \theta_n \ldots \theta_1$. The length of an n-step derivation is defined to be n. The length of a derivation may be 0 or infinite. In

a derivation we mark the selected equations in the bold face as shown in Example 2 below. A derivation of G and R w.r.t. LNC_0 which ends with the *empty clause* (denoted by $\Box$) is called a *refutation* of G and R w.r.t. LNC_0. Logically, it is a refutation of a set of clauses $\{G\} \cup R$. When $G \twoheadrightarrow^{\theta}_{LNC_0} \Box$, $\theta{\uparrow}Var(G)$ is called a *computed answer substitution* of G and R w.r.t. LNC_0.

The subset $\{[v], [t], [d]\}$ of LNC_0 is referred to as UC (standing for "unification calculus"). A derivation, a refutation and a computed answer substitution w.r.t. UC (and other calculi given later) are defined as in LNC_0.

Example 2. Let R be a conditional equational system $\{f(x) = h, g = g\}$.
and G be a goal $\Leftarrow f(g) = z$. A refutation of G and R w.r.t. LNC_0 is

$$\Leftarrow f(g) = z \ \rightarrow_{[on]} \Leftarrow \mathbf{g} = \mathbf{x}, h = z \overset{\{g/x\}}{\rightarrow}_{[v]} \Leftarrow h = z \overset{\{h/z\}}{\rightarrow}_{[v]} \Box.$$

This example shows how LNC_0 realizes so-called lazy evaluation by binding a variable to reducible term g. An infinite derivation starting from $\Leftarrow f(g) = z$ can be generated if we choose the outermost narrowing rule [on] to apply to the goal $\Leftarrow g = x$ instead of the variable elimination rule [v].

4.2 Soundness

We next consider the soundness of LNC_0 as a calculus for defining a refutation proof procedure for a given equational specification (given in the form of a conditional equational system). Let G be a goal $\Leftarrow E$ and R be a conditional equational system. A substitution $\theta{\uparrow}Var(G)$ is called a *correct answer substitution* of G and R if θE is a logical consequence of R.

Since LNC_0 is a subset of TRANS, the soundness of LNC_0 w.r.t. correct answer substitutions is an immediate consequence of Hölldobler's Theorem 7.2.3 (soundness of TRANS w.r.t. correct answer substitutions) [14, p. 186].

Theorem 1. (soundness of LNC_0) Let R be a conditional equational system and G be a goal. The computed answer substitutions of G and R w.r.t. LNC_0 are the correct answer substitutions of G and R.

4.3 Completeness

For a *conditional rewriting system* which satisfies the properties of *level-confluence* [9] and *level-normality* (italicized notions are defined shortly), LNC_0 gives a complete refutation procedure. The proof of the completeness of LNC_0 proceeds in the following way:

1. We introduce another narrowing calculus called NC. NC is a calculus more akin to the original idea of narrowing of Slagel [24].
2. We recast Giovannetti and Moiso's completeness result of conditional narrowing in the presence of extra-variables (i.e. variables occurring in its body without occurring in the LHS of the head of a conditional equation) [9] in the framework of NC.
3. We prove the completeness of LNC_0 in the presence of extra-variables by relating it to the completeness of NC.

We begin by introducing the narrowing calculus NC w.r.t. a conditional equational system R. The formulas of NC are the goals of $\mathcal{L}_0$, and the inference rules of NC are given below.

1. Narrowing [n]

$$\frac{\Leftarrow E, s \doteq t, E'}{\Leftarrow \theta(E, F, s[u \leftarrow r] = t, E') \text{ where } \theta \text{ is an mgu of } s/u \text{ and } l, \text{ and } u \in \overline{Occ}(s)} \; l = r \Leftarrow F$$

2. Reflection [f]

$$\frac{\Leftarrow E, s = t, E'}{\Leftarrow \theta(E, E') \text{ where } \theta \text{ is an mgu of } s \text{ and } t}$$

The narrowing rule [n] is applicable to an equation of a goal if there exists a new variant $l = r \Leftarrow F$ of a conditional equation in R such that $\theta(s/u) = \theta l$ for some mgu θ.

The following Theorem 2 shows that NC is a sound calculus. Theorem 2 is an immediate consequence of Hölldobler's Theorem 6.2.1 (soundness of paramodulation and reflection)[14, p.96].

Theorem 2. (soundness of NC) Let R be a conditional equational system and G be a goal. The computed answer substitutions of G and R w.r.t. NC are correct answer substitutions of G and R.

The completeness of NC depends on the confluence property of the rewrite relations induced by the conditional equational system. Given a conditional equational system R, depending on the interpretation of the equality (=) in the body of the conditional equations, we can define several conditional rewriting systems as R being its underlying conditional equational system and then can define the rewrite relations induced by them. Without extra-variables, it is sufficient to consider a confluent conditional rewriting system whose rule scheme is $t \rightarrow s \Leftarrow s_1 \downarrow t_1, \ldots, s_m \downarrow t_m$. [1] With the presence of extra-variables, Giovannetti and Moiso showed that the confluence of the rewrite relation induced by the above rule scheme is no longer sufficient for the completeness that we are interested in, and proposed a notion of level-confluence to guarantee the completeness. Below we will give the definition of the level-confluence according to Giovannetti and Moiso.

Given an equational system R we define a conditional rewriting system R^* by a set of conditional rewrite rules $t \rightarrow s \Leftarrow s_1 \downarrow t_1, \ldots, s_m \downarrow t_m$ for each conditional equation $t = s \Leftarrow s_1 = t_1, \ldots, s_m = t_m$ in R. Here, the equality (=) in the body of the conditional equations are interpreted as the predicate over the existence of a common reduct. That is, for a rewrite relation $\rightarrow_R$, $s_i \downarrow t_i$ means that there exists a term q_i such that $s_i \rightarrow\!\!\!\!\!\rightarrow_R q_i \leftarrow\!\!\!\!\!\leftarrow_R t_i$. The seeming circularity of the definition of rewrite relation $\rightarrow_R$ is to be resolved by imposing a hierarchy of the rewrite relations.

An *n-level rewrite relation* $\overset{n}{\rightarrow}_R$ $(n \geq 0)$ is inductively defined as follows:

[1] It is called a *standard* conditional rewriting system in Dershowitz and Okada [8].

1. $\overset{0}{\rightarrow}_R$ is the empty relation.

2. $t \overset{n+1}{\rightarrow}_R s$ holds iff there exists a new variant $l \rightarrow r \Leftarrow t_1 \downarrow s_1, \ldots, t_m \downarrow s_m$ of a conditional rewrite rule in R^*, an occurrence $u \in \overline{Occ}(t)$ and a substitution θ such that $s = t[u \leftarrow \theta r]$, $t/u = \theta l$ and for all $i = 1, \ldots, m$ there exists q_i such that $\theta t_i \overset{n}{\twoheadrightarrow}_R q_i \overset{n}{\twoheadleftarrow}_R \theta s_i$.

A conditional rewriting system R^* is called *level-confluent* iff, for all $i \geq 0$, the rewrite relation $\overset{i}{\rightarrow}_R$ is separately confluent.

Rewrite relation $\rightarrow_R$ is defined as $\bigcup_{i \geq 0} \overset{i}{\rightarrow}_R$. Note that level-confluence implies confluence, but not vice versa.

Similarly (beware of the slight difference), an *n-level normalized rewrite relation* $\overset{n}{\rightarrow}_R{}'$ $(n \geq 0)$ is inductively defined as follows:

1. $\overset{0}{\rightarrow}_R{}'$ is the empty relation.

2. $t \overset{n+1}{\rightarrow}_R{}' s$ holds iff there exists a new variant $l \rightarrow r \Leftarrow t_1 \downarrow s_1, \ldots, t_m \downarrow s_m$ of a conditional rewrite rule in R^*, an occurrence $u \in \overline{Occ}(t)$ and a substitution θ such that $s = t[u \leftarrow \theta r]$, $t/u = \theta l$, for all $i = 1, \ldots, m$ there exists q_i such that $\theta t_i \overset{n}{\twoheadrightarrow}_R q_i \overset{n}{\twoheadleftarrow}_R \theta s_i$ and for all extra-variables x in $t_1 = s_1, \ldots, t_m = s_m$ θx is in n-normal form (i.e. normal form w.r.t. $\overset{n}{\rightarrow}_R{}'$).

A conditional rewriting system R^* is called *level-normal* iff $\overset{i}{\rightarrow}_R = \overset{i}{\rightarrow}_R{}'$ for all $i \geq 0$. The properties of level-confluence and level-normality of a conditional rewriting system R^* are applied also to the underlying conditional equational system R. There are syntactic sufficient conditions for a conditional equational system to be level-confluent and level-normal. In the section 5, we will see that programs of the language $\mathcal{L}_1$ are level-confluent and level-normal.

Now we have the following theorem of Giovannetti and Moiso.

Theorem 3. (completeness of NC) Let R be a conditional equational system which is level-confluent and level-normal, and G be a goal. For any normalizable correct answer substitution θ of G and R, there exists a computed answer substitution θ' of G and R w.r.t. NC such that $\theta' \geq \theta\downarrow$.

Remark 1. Giovannetti and Moiso presented the theorem for terminating conditional term rewriting systems [9]. The terminating assumption was needed at two places in their proof. It was required firstly to guarantee R to be level-normal, and secondly to guarantee the given substitution θ to be normalizable. The first requirement can be lifted when we assume R to be level-normal. The second is lifted when we consider only normalizable correct answer substitutions.

In order to prove the completeness of LNC_0 we need the unification theorem of Martelli and Montanari [21] and lifting and switching lemmas for LNC_0.

The following Lemma 1 is a restatement of Martelli and Montanari's Theorem 2.3 [21, p. 262] in our framework.

Lemma 1.(unification theorem of Martelli and Montanari) Let G be a goal. If there exists a refutation $G \overset{\theta}{\rightarrow}_{[f]} \square$ then there exists a refutation $G \overset{\theta}{\twoheadrightarrow}_{UC} \square$.

Martelli and Montanari showed that UC gives an mgu θ' such that $\theta' \sim \theta$. We easily see that we can obtain θ' which is the same as θ.

Lemma 2.(lifting lemma for LNC_0) Let R be a conditional equational system, G be a goal and γ be a substitution. If there exists a refutation $\gamma G \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square$, then there exists a refutation $G \overset{\theta'}{\twoheadrightarrow}_{LNC_0} \square$ such that $\theta' \geq \theta\gamma$, and whose length is the same as the length of the refutation $\gamma G \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square$.

Lemma 3.(switching lemma for LNC_0) Let R be a conditional equational system. If there exists a refutation $\Pi : \Leftarrow E, \mathbf{s} = \mathbf{t}, s' = t', E' \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square$ where $s = t$ is selected in the first step and $s' = t'$ in the second step of the refutation, then there exists a refutation $\Leftarrow E, s = t, \mathbf{s}' = \mathbf{t}', E' \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square$ which is the same as Π except that the first two selections of the equations are switched and whose length is the same as the length of the refutation Π.

Lemmas 2 and 3 can be proved by the induction on the length of the refutation w.r.t. LNC_0. Detailed proofs are given in appendix A.1 and A.2 respectively.

Lemma 4. (completeness of LNC_0 w.r.t. NC) Let R be a conditional equational system and G be a goal. If there exists a refutation $G \overset{\theta}{\twoheadrightarrow}_{NC} \square$ then there exists a refutation $G \overset{\theta'}{\twoheadrightarrow}_{LNC_0} \square$ such that $\theta' \geq \theta$.

Lemma 4 is proved by the induction on the length of the refutation $G \overset{\theta}{\twoheadrightarrow}_{NC} \square$ using Lemmas 1, 2 and 3. The proof is given in appendix A.3.

By Theorem 3 and Lemma 4 we obtain the completeness of LNC_0.

Theorem 4.(completeness of LNC_0) Let G be a goal and R be a conditional equational system which is level-confluent and level-normal. For any normalizable correct answer substitution θ of G and R, there exists a computed answer substitution θ' of G and R w.r.t. LNC_0 such that $\theta' \geq \theta\downarrow$.

Remark 2. LNC_0 is a subset of Hölldobler's TRANS (complete set of transformation rules). TRANS contains [pv] (paramodulation at a variable position) and [tc] (application of a trivial clause) in addition to the rules of LNC_0. Since we are interested in a conditional equational system which can be interpreted as a conditional term rewriting system rather than general conditional equations, we omit from the beginning of our discussion the [pv] and [tc] of TRANS.

Although LNC_0 enjoys the soundness and completeness as a calculus for giving an equation solving procedure it is not qualified, in our view, as a computation model for a programming language because of its inefficiency. For a given equation, applicable inference rules are not unique. This will create a large search space which is untolerable in practice. LNC_0 is theoretically important, however, since it is the basis of the calculus LNC_1 which we explain next.

5 Language $\mathcal{L}_1$

We first present the language $\mathcal{L}_1$ for the narrowing calculus LNC_1. $\mathcal{L}_1$ is a more restricted language than $\mathcal{L}_0$. The reasons for imposing further restrictions on $\mathcal{L}_0$ are as follows.

- We are interested in the integration of functional and logic programming, rather than adding equational reasoning capabilities to logic programming. Hence we view conditional equations as defining functions. In practice, conditional equations are used in more restricted form than in $\mathcal{L}_0$ when they define functions. For example, in ML, equational expressions are of the form

  $$f(d_1, \ldots, d_n) = t$$

  where $d_1, \ldots, d_n$ are restricted to expressions called atomic patterns. The following expression

  $$f(x, f(y, z)) = f(f(x, y), z) \tag{1}$$

  which is regarded as an assertion about a property of the function f (associativity) are excluded from ML programs since $f(y, z)$ in (1) is not an atomic pattern. To exclude the (conditional) equations like (1), we distinguish, in $\mathcal{L}_1$, constructor symbols from function symbols, following the design philosophy of modern functional programming languages. Constructor symbols are used to construct data structures. For example, `cons` in Example 1 is a constructor symbol which is used to construct a list.

- The completeness of LNC_0 requires a given conditional equational system to be level-confluent and level-normal. These properties, however, are known to be undecidable in general. Therefore, we must guarantee that a conditional equational system possesses these properties by some easily realizable measures, for example, by syntactic restrictions.

5.1 Terms and equations in $\mathcal{L}_1$

Terms in $\mathcal{L}_1$ are classified into three categories, i.e. function terms, constructor terms and variables, depending on the leftmost symbol of the terms. A *function term* (respectively *constructor term*) is a term whose leftmost symbol is a function (respectively constructor) symbol. A *data term* is either a variable or a constructor term whose proper subterms are data terms. Hereafter, we use d and e to denote data terms.

We require the following conditions on a conditional equational system R.

C1. Each conditional equation in R is of the form:

$$f(d_1, \ldots, d_n) = t \Leftarrow t_1 = e_1, \ldots, t_m = e_m$$

where $d_1, \ldots, d_n$ are data terms and $e_1, \ldots, e_m$ are ground data terms.

C2. R satisfies the following conditions:

C2-1. (weak non-ambiguity) For any conditional equations $s_1 = t_1 \Leftarrow F_1$ and $s_2 = t_2 \Leftarrow F_2$ in R, if there exists a substitution θ such that $\theta(s_1/u) = \theta s_2$ for some $u \in \overline{Occ}(s_1)$ then $\theta(s_1[u \leftarrow t_2]) = \theta t_1$.

C2-2. (left linearity) For any conditional equations $s = t \Leftarrow F$ in R, s is a linear term.

Note that when the condition C1 is assumed in the condition C2-1, only possible case of $\theta(s_1/u) = \theta s_2$ is for $u = \Lambda$.

For a conditional rewriting system R^*, we define an induced unconditional rewriting system $U(R^*)$ as $\{l \rightarrow r \mid l \rightarrow r \Leftarrow s_1 \downarrow t_1, \ldots, s_n \downarrow t_n \in R^*\}$, and define (unconditional) rewrite relation $\rightarrow_{U(R^*)}$ as usual. A conditional rewriting system R^* is $\mathbb{III}n$ if R satisfies the condition C2 and the RHSs of the equations in the body of conditional equations of R are in ground normal forms w.r.t. $\rightarrow_{U(R^*)}$ [1]. R is called $\mathbb{III}n$ if R^* is $\mathbb{III}n$. We see that $\mathbb{III}n$ conditional equational systems are level-confluent (Theorem 3.5 [1]), and level-normal (pointed out in [11]).

For a goal we require the following conditions:

C3. A goal $\Leftarrow E, t = d, E'$ satisfies the following conditions:

C3-1. $Var(t) \cap Var(d) = \emptyset$.

C3-2. $Var(E) \cap Var(d) = \emptyset$.

C3-3. d is a linear data term.

Note:

- The conditions C3 (C3-1 $\sim$ C3-3) look awkward at first sight. They are, however, necessary to ensure the soundness and completeness of LNC_1. We will show the examples later.
- The conditions C3 are imposed on an initial goal. We will show later in Proposition 1 (1) and (2) that goals created by the applications of the inference rules of LNC_0 or LNC_1 also satisfy the conditions C3.

A conditional equational system which satisfies the conditions C1 and C2 is referred to as an $\mathcal{L}_1$ *program* and a conditional equation in an $\mathcal{L}_1$ program is referred to as an $\mathcal{L}_1$ conditional equation. We easily see that $\mathcal{L}_1$ programs are $\mathbb{III}n$, and therefore that they are level-confluent and level-normal. A goal which satisfies the conditions C3 is referred to as an $\mathcal{L}_1$ *goal.* Note that since an $\mathcal{L}_1$ program and an $\mathcal{L}_1$ goal are also a conditional equational system of $\mathcal{L}_0$ and a goal of $\mathcal{L}_0$ respectively, LNC_0 and NC are well defined for an $\mathcal{L}_1$ program, and an $\mathcal{L}_1$ goal is a legitimate formula of LNC_0 and NC.

The disparity of the restrictions on the bodies and the goals is accounted for by the observations where we ask an oracle to work. The level-normality demands that extra-variables are to be instantiated to normal forms when the equations containing them are introduced to a goal by an inference step [In] [2], whereas in an initial goal extra-variables (synonymous to free variables in this context) are not requested to be instantiated to normal forms. As we will see later, for LNC_1 we only insist on the completeness for normalizable answer substitutions. Since an answer substitution is an "answer," it is user's (or human's) responsibility to decide whether terms obtained in answer substitutions are normalizable. Of course it is undecidable to know terms are normalizable for a non-terminating term rewriting system. Thus a user has to run a risk of non-terminating computation when he wants to ascertain the normalizability of the answer, whereas in the goals the calculus itself has to check whether extra-variables are instantiated to a normalizable term or not. This is undecidable again. However, to continue computation without losing the level-normality the calculus has to know it. A simple method to check that extra-variables are always instantiated to normal forms is to make the conditional term rewriting systems $\mathbb{III}n$.

[2] to be given in section 6.1

- Φ_1: transformation of an $\mathcal{L}_1$ conditional equation to homogeneous form

$$\Phi_1[\![f(\ldots,d,\ldots)=s \Leftarrow E]\!] = \Phi_1[\![f(\ldots,x,\ldots)=s \Leftarrow x=d,E]\!]$$

  where d is a non-variable data term and x is a fresh variable.

- Φ_2: shallowing for the bodies of an $\mathcal{L}_1$ conditional equation:

$$A \Leftarrow \Phi_2[\![\ldots,s=c(\ldots,d,\ldots),\ldots]\!] = A \Leftarrow \Phi_2[\![\ldots,s=c(\ldots,x,\ldots),x=d,\ldots]\!]$$

  where d is a non-variable data term, x is a fresh variable and A is an equation.

- Φ_3: shallowing for an $\mathcal{L}_1$ goal:

$$\Leftarrow \Phi_3[\![\ldots,s=c(\ldots,d,\ldots),\ldots]\!] = \Leftarrow \Phi_3[\![\ldots,s=c(\ldots,x,\ldots),x=d,\ldots]\!]$$

  where d is a non-variable data term or a variable occurring in the initial goal, and x is a fresh variable.

Figure 1: transformation rules

If the computed answer substitution is used for further computation it is sensible (and safe) to ask initial goals to satisfy the condition that we request to bodies. Since we want to obtain a completeness result that is as general as possible we impose the conditions C3 on initial goals, which are less strict than the conditions of $\mathbb{III}n$.

5.2 Transformation to the basic forms

$\mathcal{L}_1$ programs are sufficiently expressive as functional-logic programs. It is still structurely very complex, however. In order for LNC_1 to be simple and efficient as a calculus, we transform $\mathcal{L}_1$ conditional equations and goals to structurely simpler forms called basic $\mathcal{L}_1$ conditional equations and basic $\mathcal{L}_1$ goals respectively. The transformation consists of three transformation rules: transformation to a homogeneous form and shallowing on the bodies of $\mathcal{L}_1$ conditional equations and on initial $\mathcal{L}_1$ goals. The transformation rules are given in Figure 1.

An $\mathcal{L}_1$ conditional equation of the form $f(x_1,\ldots,x_n)=t \Leftarrow F$ where $x_1,\ldots,x_n$ are distinct variables is called *homogeneous* [25]. The transformation rule Φ_1 shown in Figure 1 transforms an $\mathcal{L}_1$ conditional equation to a homogeneous $\mathcal{L}_1$ conditional equation.

A data term of the form $c(x_1,\ldots,x_n)$ or a variable is called *shallow*. The transformation rule Φ_2 transforms an $\mathcal{L}_1$ conditional equation to an $\mathcal{L}_1$ conditional equation such that the RHSs of equations in the body are linear shallow data terms, and Φ_3 transforms an $\mathcal{L}_1$ goal to an $\mathcal{L}_1$ goal such that the RHSs of equations in the goal are linear shallow data terms. The notion of shallowness is due to Cheong [6].

There is a subtle difference between Φ_2 and Φ_3. Φ_3 is applicable to a sequence of equations of the form $\ldots,s=c(\ldots,d,\ldots),\ldots$ even when d is a variable if it occurs also in the initial goal,

whereas Φ_2 is not. We will see in section 6.3 that this additional shallowing by Φ_3 is essential to obtain a class of substitutions called constructor term substitutions (to be defined in section 6.3) which we want to regard as answers of the computation.

To an $\mathcal{L}_1$ conditional equation, Φ_1 is first applied and then Φ_2 is applied. The resulting conditional equation is homogeneous and all the RHSs of the equations in the body are shallow. It is called a *basic* conditional equation. For an $\mathcal{L}_1$ conditional equational system R, $Basic(R)$ is defined as a set consisting of basic $\mathcal{L}_1$ conditional equations. To be precise, $Basic(R) = \{A' \Leftarrow \Phi_2[\![F']\!] \mid \text{let } A' \Leftarrow F' \text{ be } \Phi_1[\![A \Leftarrow F]\!], A \Leftarrow F \in R\}$. Similarly, a *basic* goal, written as $Basic(G)$, is defined for an $\mathcal{L}_1$ goal G. That is, $Basic(G)$ is $\Leftarrow \Phi_3[\![E]\!]$, where G is $\Leftarrow E$.

6 Lazy narrowing calculus LNC_1

In this section we present the main result of the paper. It consists in the lazy narrowing calculus LNC_1 and its soundness and completeness. We begin by introducing LNC_1 w.r.t. a basic $\mathcal{L}_1$ program $Basic(R)$. The formulas of LNC_1 are basic $\mathcal{L}_1$ goals.

6.1 Inference rules

1. Lazy narrowing [ln]

$$\frac{\Leftarrow f(s_1, \ldots, s_n) = d, E}{\Leftarrow \theta(F, t = d, E) \text{ where } \theta = \{s_1/x_1, \ldots, s_n/x_n\}} \quad f(x_1, \ldots, x_n) = t \Leftarrow F$$

2. Variable elimination of data terms [vd]

$$\frac{\Leftarrow x = d, E}{\Leftarrow \theta E \text{ where } \theta = \{d/x\}}$$

3. Variable elimination of constructor terms [vc]

$$\frac{\Leftarrow c(s_1, \ldots, s_n) = x, E}{\Leftarrow \theta E \text{ where } \theta = \{c(s_1, \ldots, s_n)/x\}}$$

4. Unification of constructor terms [u]

$$\frac{\Leftarrow c(s_1, \ldots, s_n) = c(x_1, \ldots, x_n), E}{\Leftarrow \theta E \text{ where } \theta = \{s_1/x_1, \ldots, s_n/x_n\}}$$

Note:

- The lazy narrowing rule [ln] is applicable to a goal if there exists a new variant $f(x_1, \ldots, x_n) = t \Leftarrow F$ of a basic $\mathcal{L}_1$ conditional equation in $Basic(R)$.
- The variable elimination rule [vc] and the unification rule [u] are applied to an equation of the goals whose LHS is a constructor term.
- By Proposition 1 (2) which we give shortly, $Var(t) \cap Var(d) = \emptyset$ for any equation $t = d$ of the goals in a derivation w.r.t. LNC_1. Hence the occur check is unnecessary when a binding is formed in the application of the rules of LNC_1.

Let R be an $\mathcal{L}_1$ program and G be an $\mathcal{L}_1$ goal. When there exists a refutation $Basic(G) \overset{\theta}{\twoheadrightarrow}_{LNC_1} \square$ of $Basic(G)$ and $Basic(R)$ w.r.t. LNC_1, we call $\theta\!\uparrow Var(G)$ a *computed answer substitution* of G and R w.r.t. LNC_1.

From the efficiency point of view, LNC_1 has following advantages over LNC_0.

- In LNC_1, at most one rule is applied to a selected equation in a goal, whereas in LNC_0 more than one rule may be applied. The computation in LNC_1, therefore, proceeds in a more determinate way, and the search space in the refutations w.r.t. LNC_1 is greatly reduced compared with LNC_0.
- In LNC_1, the occur check is unnecessary.

Example 3. Let R be an $\mathcal{L}_1$ program $\{f(x) = c, g = g\}$ and G be an $\mathcal{L}_1$ goal $\Leftarrow f(g) = z$. The refutation of $Basic(G)$ $(= G)$ and $Basic(R)$ $(= R)$ w.r.t. LNC_1 is as follows:

$$\Leftarrow f(g) = z \overset{\{g/x\}}{\underset{[\mathrm{ln}]}{\rightarrow}} \Leftarrow c = z \overset{\{c/z\}}{\underset{[\mathrm{vc}]}{\rightarrow}} \square$$

with the computed answer substitution $\{c/z\}$.

We give below some properties concerning syntactical structures of goals. The following Proposition 1 guarantees that the goals appearing in a derivation w.r.t. LNC_0 (respectively LNC_1) are $\mathcal{L}_1$ goals (respectively basic $\mathcal{L}_1$ goals). The proof is straightforward by the case analysis.

Proposition 1. Let R be an $\mathcal{L}_1$ program and G be an $\mathcal{L}_1$ goal.

(1) All goals in a derivation of G and R w.r.t. LNC_0 are $\mathcal{L}_1$ goals.

(2) All goals in a derivation of $Basic(G)$ and $Basic(R)$ w.r.t. LNC_1 are basic $\mathcal{L}_1$ goals.

The following technical lemma is used in the proof of Lemmas 6 and 10.

Proposition 2. Let R be an $\mathcal{L}_1$ program and $\Leftarrow x = d$ be an $\mathcal{L}_1$ goal. There is a unique refutation $\Leftarrow \Phi_3[\![x = d]\!] \overset{\theta}{\twoheadrightarrow}_{LNC_1} \square$ such that $\theta\!\uparrow\!\{x\} = \{d/x\}$ and $Dom(\theta) - \{x\} = Var(\Phi_3[\![x = d]\!]) - Var(x = d)$.

(Proof) By the induction on the structure of d.

∎

Furthermore, the following Proposition 3 shows that the variables in the initial goal occur only in certain goals of the derivation. The proof of Proposition 3 is straightforward by the induction on the length of the derivation.

Proposition 3. Let R be an $\mathcal{L}_1$ program and G be an $\mathcal{L}_1$ goal. In the goals of a derivation of $Basic(G)$ and $Basic(R)$ w.r.t. LNC_1,

(1) for all equations of the form $t = x$,
$x \in Var(G)$, and

(2) for all equations of the form $t = c(x_1, \ldots, x_n)$,
$x_1, \ldots, x_n \notin Var(G)$.

As we saw in Example 2, in LNC_0 the lazy evaluation is realized by applying the variable elimination [v] to an equation of the form $t = x$, where t is a reducible term. Since LNC_1 provides [vc], the lazy evaluation is realized also in LNC_1 when t is reducible to a constructor term or t is already a constructor term. This was shown in Example 3. However, since LNC_1 does not provide the variable elimination of function terms, it fails to find a correct answer substitution whose codomain contains function terms. For example, the goal $\Leftarrow f(g) = z$ in Example 3 has $\{f(g)/z\}$ as a correct answer substitution, but LNC_1 can not find it. Furthermore, when t is not reducible to a constructor term, an infinite derivation is caused. Consider a goal $\Leftarrow g = x$. It causes an infinite derivation. One might think that these are drawbacks of LNC_1, if one consider all correct answer substitutions as desired answers. However, these are not so actually. We discuss on this issue in the next section.

6.2 Consideration on answer substitutions

In logic programming [3], correct answer substitutions are considered as desired answers. This idea does not entirely capture the notion of the answer of lazy functional-logic programs. In logic programming, function symbols are used to construct data structures, and terms are not evaluated to other terms. It is natural to consider terms bound to variables in an initial goal as the answers of computation. On the other hand, in our functional-logic programming, constructor symbols are distinguished from function symbols. Constructor symbols are used to construct data structures as in logic programming, but function symbols are used to name user-defined functions which are used to form a function term that is to be evaluated. Function terms are no longer regarded as answers of computation. As a desired answer of functional-logic programs, we have to choose either data terms or constructor terms. We have chosen constructor terms as desired answers, because we consider lazy evaluation. We guarantee LNC_1 to be complete for these desired answer substitutions. A detailed discussion on this decision is given in [22].

Hereafter, we consider correct answer substitutions whose codomain consists of constructor terms or variables as our desired answers. The discussions of soundness and completeness in the following sections observe this decision.

6.3 Soundness

The following Lemma guarantees that computed answer substitutions w.r.t. LNC_1 are constructor term substitutions. A substitution θ is called a *constructor term substitution* iff terms t in

[3] We are assuming here logic programming in Prolog.

$Cod(\theta)$ implies either t is a constructor term or a variable. Note that the empty substitution ε is a constructor term substitution.

Lemma 5. Let R be an $\mathcal{L}_1$ program and G be an $\mathcal{L}_1$ goal. Let Π be a derivation

$$G_0(=Basic(G)) \rightarrow^{\theta_1}_{[\alpha_1]} G_1 \rightarrow^{\theta_2}_{[\alpha_2]} \cdots \rightarrow^{\theta_{i-1}}_{[\alpha_{i-1}]} G_{i-1} \rightarrow^{\theta_i}_{[\alpha_i]} G_i \rightarrow^{\theta_{i+1}}_{[\alpha_{i+1}]} \cdots$$

where $[\alpha_1], [\alpha_2], \ldots \in LNC_1$. Furthermore, let σ_i, $i \geq 0$ be substitutions such that

$$\begin{cases} \sigma_0 & = \varepsilon \\ \sigma_i & = \theta_i \sigma_{i-1} \ (i \geq 1). \end{cases}$$

Then, for all $i \geq 0$, $\sigma_i \upharpoonright Var(G)$ is a constructor term substitution.

(Proof) We will show, by the induction on i, that (i) $\sigma_i \upharpoonright Var(G)$ is a constructor term substitution and that (ii) the variables occurring in the RHSs of the equations of the goal G_i are not in $Dom(\sigma_i \upharpoonright Var(G)) \cup Vcod(\sigma_i \upharpoonright Var(G))$ for all $i \geq 0$.

When $i = 0$, (i) and (ii) obviously hold. For the induction step, assume that (i) and (ii) hold when $i = k - 1$. We will show (i) and (ii) hold when $i = k$. We distinguish the following four cases according to the cases of $[\alpha_k]$.

When $[\alpha_k]$ is the unification rule [u], letting G_{k-1} be $\Leftarrow c(s_1, \ldots, s_n) = c(x_1, \ldots, x_n), E$, we have a derivation:

$$\Leftarrow c(s_1, \ldots, s_n) = c(x_1, \ldots, x_n), E \rightarrow^{\theta_k}_{[u]} \Leftarrow \theta_k E (= G_k)$$

where $\theta_k = \{s_1/x_1, \ldots, s_n/x_n\}$. By the induction hypothesis, $x_1, \ldots, x_n \notin Dom(\sigma_{k-1} \upharpoonright Var(G)) \cup Vcod(\sigma_{k-1} \upharpoonright Var(G))$. Hence we have

$$\begin{aligned} \sigma_k \upharpoonright Var(G) &= \theta_k \upharpoonright Var(G) \cup \sigma_{k-1} \upharpoonright Var(G) \\ &= \sigma_{k-1} \upharpoonright Var(G) \qquad \text{by Proposition 3 (2).} \end{aligned}$$

Hence (i) holds when $i = k$. By the induction hypothesis, variables occurring in the RHSs of the equations of E are not in $Dom(\sigma_{k-1} \upharpoonright Var(G)) \cup Vcod(\sigma_{k-1} \upharpoonright Var(G))$. Furthermore, since θ_k does not affect the RHS of the equations of E by Proposition 1 (2), $\theta_k E = E$. Hence (ii) holds when $i = k$.

The case of $[\alpha_i]$ being [ln] is similar to the case of [u]. The cases of [vd] and [vc] are easy since only a constructor term substitution is formed in the application of the inference rules.

∎

Remark 3. It is essential for the soundness (and the completeness) of LNC_1 that the transformation Φ_3 is applied to an initial goal $\Leftarrow \ldots, s = c(\ldots, d, \ldots), \ldots$ even if d is a variable occurring in the initial goal. For example, let R be an $\mathcal{L}_1$ program $\{a = d\}$ and G be an $\mathcal{L}_1$ goal $\Leftarrow c(a) = c(x)$, where c and d are constructor symbols. $Basic(G)$ is $\Leftarrow c(a) = c(z), z = x$. We obtain the refutation of $Basic(G)$ and $Basic(R)(= R)$ w.r.t. LNC_1:

$$\begin{aligned} \Leftarrow c(a) = c(z), z = x \quad &\rightarrow^{\{a/z\}}_{[u]} \quad \Leftarrow a = x \\ &\rightarrow^{\varepsilon}_{[ln]} \quad \Leftarrow d = x \\ &\rightarrow^{\{d/x\}}_{[vc]} \quad \square \end{aligned}$$

and a computed answer substitution $\{d/x\}$. If we use $\Leftarrow c(a) = c(x)$ instead of $Basic(G)$ however, the refutation would be as follows;

$$\Leftarrow c(a) = c(x) \overset{\{a/x\}}{\rightarrow}_{[u]} \Box.$$

We would obtain a non-constructor term substitution instead of the desired answer $\{d/x\}$. Hence the soundness (in the sense of Theorem 5 below) of LNC_1 would not hold.

Lemma 6.(soundness of LNC_1 w.r.t. LNC_0) Let R be an $\mathcal{L}_1$ program and G be an $\mathcal{L}_1$ goal. If there exists a refutation

$$Basic(G) \twoheadrightarrow^{\theta}_{LNC_1} \Box$$

then there exists a refutation

$$G \twoheadrightarrow^{\theta'}_{LNC_0} \Box,$$

such that $\theta' \uparrow Var(G) = \theta \uparrow Var(G)$.

(Proof) We use the following sublemma.

Sublemma: Let G_0 be an $\mathcal{L}_1$ goal. If there exists a refutation Π : $Basic(G_0) \twoheadrightarrow^{\theta_0}_{LNC_1} \Box$, then there exists a refutation $G_0 \twoheadrightarrow^{\theta'_0}_{LNC_0} \Box$, such that $\theta'_0 \uparrow Var(G) = \theta_0 \uparrow Var(G)$.

In particular, when Π is $Basic(G) \twoheadrightarrow^{\theta}_{LNC_1} \Box$, the sublemma gives Lemma 6.

Proof of the sublemma: The proof is by the induction on the length k of the refutation Π. If $k = 0$ the result immediately holds. Otherwise, we distinguish the following four cases according to the first step of the refutation Π.

[1] (when the first step is [vd]) In this case, G_0 is of the form $\Leftarrow x = c(d_1, \ldots, d_n), E$. $Basic(G_0)$ is $\Leftarrow x = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!]$. Hence, Π is

$$\begin{array}{ll}
 & Basic(G_0) \\
\rightarrow^{\sigma_1}_{[vd]} & \Leftarrow \sigma_1(\Phi_3[\![z_1 = d_1, \ldots, z_n = d_n]\!], \Phi_3[\![E]\!]) \text{ where } \sigma_1 = \{c(z_1, \ldots, z_n)/x\} \\
 & \text{(which is the same as } \Leftarrow \Phi_3[\![z_1 = d_1]\!], \ldots, \Phi_3[\![z_n = d_n]\!], \Phi_3[\![\sigma_1 E]\!] \\
 & \text{since } x \text{ does not appear in } d_1, \ldots, d_n \text{ and in the RHSs of the equations in } E) \\
\twoheadrightarrow^{\sigma_2}_{LNC_1} & \Leftarrow \sigma_2 \Phi_3[\![\sigma_1 E]\!] \qquad \text{by Proposition 2} \\
\twoheadrightarrow^{\theta_1}_{LNC_1} & \Box,
\end{array}$$

where σ_2 is a substitution such that $\sigma_2 \uparrow \{z_1, \ldots, z_n\} = \{d_1/z_1, \ldots, d_n/z_n\}$ and $(Dom(\sigma_2) - \{z_1, \ldots, z_n\})$ consists of fresh variables introduced by Φ_3, and θ_1 is a substitution such that $\theta_1 \sigma_2 \sigma_1 = \theta_0$. Since σ_2 does not affect the RHSs of the equations of E, the goal $\Leftarrow \sigma_2 \Phi_3[\![\sigma_1 E]\!]$ is the same as $\Leftarrow \Phi_3[\![\sigma_2 \sigma_1 E]\!]$. Let $\sigma = \{c(d_1, \ldots, d_n)/x\}$. Then $\sigma_2 \sigma_1 E$ is the same as σE. Hence, we have a refutation $\Leftarrow \Phi_3[\![\sigma E]\!] \twoheadrightarrow^{\theta_1}_{LNC_1} \Box$. By the induction hypothesis, we obtain a refutation $\Leftarrow \sigma E \twoheadrightarrow^{\theta'_1}_{LNC_0} \Box$ such that $\theta'_1 \uparrow Var(G) = \theta_1 \uparrow Var(G)$. Since $\Leftarrow \mathbf{x} = \mathbf{c(d_1, \ldots, d_n)}, E \rightarrow^{\sigma}_{[v]} \Leftarrow \sigma E$, letting $\theta'_0 = \theta'_1 \sigma$, we obtain the desired refutation $G_0 \twoheadrightarrow^{\theta'_0}_{LNC_0} \Box$ such that $\theta'_0 \uparrow Var(G) = \theta_0 \uparrow Var(G)$.

[2] (when the first step is [ln]) In this case, G_0 is of the form $\Leftarrow f(s_1, \ldots, s_n) = c(d_1, \ldots, d_n), E$. $Basic(G_0)$ is $\Leftarrow f(s_1, \ldots, s_n) = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!]$. Let $f(y_1, \ldots, e_j, \ldots, y_n) = t \Leftarrow \Phi_2[\![y_1 = e_1, \ldots, y_{j-1} = e_{j-1}, y_{j+1} = e_{j+1}, \ldots, y_n = e_n, F]\!]$ be a

new variant of a basic $\mathcal{L}_1$ conditional equation used in the first step of Π. For simplicity we consider only the case that e_j is a variable and $e_1, \ldots, e_{j-1}, e_{j+1}, \ldots, e_n$ are non-variable terms. We write e_j as x_j. Π is

$$
\begin{array}{ll}
 & Basic(G_0) \\
\rightarrow^{\sigma}_{[\mathrm{ln}]} & \Leftarrow \sigma(\Phi_2[\![y_1 = e_1, \ldots, y_{j-1} = e_{j-1}, y_{j+1} = e_{j+1}, \ldots, y_n = e_n, F]\!], \\
 & \qquad t = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!]) \\
 & \text{where } \sigma = \{s_1/y_1, \ldots, s_j/x_j, \ldots, s_n/y_n\} \\
\rightarrow\!\!\!\rightarrow^{\theta_1}_{LNC_1} & \square,
\end{array}
$$

where θ_1 is a substitution such that $\theta_1\sigma = \theta_0$. Since σ does not affect $e_1, \ldots, e_{j-1}, e_{j+1}, \ldots, e_n$, $d_1, \ldots, d_{j-1}, d_{j+1}, \ldots, d_n$, and the RHSs of E and F, the goal $\Leftarrow \sigma(\Phi_2[\![y_1 = e_1, \ldots, y_{j-1} = e_{j-1}, y_{j+1} = e_{j+1}, \ldots, y_n = e_n, F]\!], t = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!])$ is the same as $\Leftarrow \Phi_2[\![s_1 = e_1, \ldots, s_{j-1} = e_{j-1}, s_{j+1} = e_{j+1}, \ldots, s_n = e_n, \sigma F]\!], \sigma t = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, \sigma E]\!]$. Furthermore, since $e_1, \ldots, e_{j-1}, e_{j+1}, \ldots, e_n$ and the RHSs of F does not contain variables occurring in the initial goal, $\Phi_2[\![s_1 = e_1, \ldots, s_{j-1} = e_{j-1}, s_{j+1} = e_{j+1}, \ldots, s_n = e_n, \sigma F]\!]$ is the same as $\Phi_3[\![s_1 = e_1, \ldots, s_{j-1} = e_{j-1}, s_{j+1} = e_{j+1}, \ldots, s_n = e_n, \sigma F]\!]$. Therefore, we obtain a refutation $\Leftarrow \Phi_3[\![s_1 = e_1, \ldots, s_{j-1} = e_{j-1}, s_{j+1} = e_{j+1}, \ldots, s_n = e_n, \sigma F, \sigma t = c(d_1, \ldots, d_n), \sigma E]\!] \rightarrow\!\!\!\rightarrow^{\theta_1}_{LNC_1} \square$. By the induction hypothesis, we obtain a refutation $\Leftarrow s_1 = e_1, \ldots, s_{j-1} = e_{j-1}, s_{j+1} = e_{j+1}, \ldots, s_n = e_n, \sigma F, t = c(d_1, \ldots, d_n), \sigma E \rightarrow\!\!\!\rightarrow^{\theta'_1}_{LNC_0} \square$ such that $\theta'_1{\uparrow}Var(G) = \theta_1{\uparrow}Var(G)$. We obtain

$$
\begin{array}{ll}
 & \Leftarrow f(s_1, \ldots, s_n) = c(d_1, \ldots, d_n), E \\
\rightarrow_{[\mathrm{on}]} & \Leftarrow s_1 = e_1, \ldots, \mathbf{s_j = x_j}, \ldots, s_n = e_n, F, t = c(d_1, \ldots, d_n), E \\
 & \text{where a new variant } f(e_1, \ldots, x_j, \ldots, e_n) = t \Leftarrow F \text{ of an } \mathcal{L}_1 \text{ conditional equation is used} \\
\rightarrow^{\sigma'}_{[\mathrm{v}]} & \Leftarrow s_1 = e_1, \ldots, s_{j-1} = e_{j-1}, s_{j+1} = e_{j+1}, \ldots, s_n = e_n, \sigma' F, \sigma' t = c(d_1, \ldots, d_n), \sigma' E \\
 & \text{where } \sigma' = \{s_j/x_j\}.
\end{array}
$$

Let $\theta'_0 = \theta'_1\sigma'$. We have the desired refutation $Basic(G_0) \rightarrow\!\!\!\rightarrow^{\theta'_0}_{LNC_0} \square$ such that $\theta'_0{\uparrow}Var(G) = \theta_0{\uparrow}Var(G)$.

[3,4] The proofs of the cases that the first step of Π is [vc] and [u] are straightforward.

∎

By Lemmas 5, 6 and Theorem 1, we obtain the soundness of LNC_1.

Theorem 5.(soundness of LNC_1) Let R be an $\mathcal{L}_1$ program and G be an $\mathcal{L}_1$ goal. The computed answer substitution of G and R w.r.t. LNC_1 is a constructor term substitution and a correct answer substitution of G and R.

The following example shows that the condition C3-1 is necessary for the soundness of LNC_1.

Example 4. Let R be an $\mathcal{L}_1$ program $\{f(y) = c(y)\}$ and G be an $\mathcal{L}_1$ goal $\Leftarrow f(x) = x$. G does not satisfy the condition C3-1. Then there would be a derivation of $Basic(G)(= G)$ and $Basic(R)(= R)$ w.r.t. LNC_1:

$$\Leftarrow f(x) = x \overset{\{x/y\}}{\rightarrow}_{[\mathrm{ln}]} \Leftarrow c(x) = x \overset{\{c(x)/x\}}{\rightarrow}_{[\mathrm{vc}]} \square$$

with a computed answer substitution $\{c(x)/x\}$ of G and R. $\{c(x)/x\}$ is not a correct answer substitution of G and R, however.

6.4 Completeness

We next consider the completeness of LNC_1. We will prove that, for any refutation $G \twoheadrightarrow_{LNC_0} \square$, there exists a corresponding refutation $G \twoheadrightarrow_{LNC_1} \square$. Then, the completeness of LNC_1 follows from the completeness of LNC_0 (Theorem 4). We first give following three technical lemmas.

Lemma 7. Let R be a conditional equational system, G be a goal, and σ and σ' be substitutions such that $\sigma \leftrightarrow\!\!\!\!\twoheadrightarrow_R \sigma'$. If there exists a refutation $\sigma G \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square$ then there exists a refutation $\sigma' G \overset{\theta'}{\twoheadrightarrow}_{LNC_0} \square$ such that $\theta \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta'$, and whose length is the same as the length of the refutation $\sigma G \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square$.

(Proof) The proof is straightforward by the induction on the length of the refutation $\sigma G \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square$

∎

Lemma 8. Let G be a goal $\Leftarrow s_1 = t_1, \ldots s_n = t_n$ and R be a conditional equational system that is level-confluent and level-normal. If θ is a normalizable correct answer substitution of G and R then for $i = 1, \ldots, n$, $\theta s_i \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta t_i$.

(Proof) By Theorem 3, there exists a refutation $G \overset{\theta'}{\twoheadrightarrow}_{NC} \square$ such that $\theta' \geq \theta\downarrow$. Let $\gamma\theta' = \theta\downarrow$. It is easy to prove by the induction on the length of the refutation that if $G \overset{\theta'}{\twoheadrightarrow}_{NC} \square$ then $\theta' s_i \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta' t_i$ for $i = 1, \ldots, n$. Therefore, we obtain $\theta s_i \twoheadrightarrow_R \gamma\theta' s_i \leftrightarrow\!\!\!\!\twoheadrightarrow_R \gamma\theta' t_i \twoheadleftarrow_R \theta t_i$ for $i = 1, \ldots, n$.

∎

The following lemma shows that in a refutation w.r.t. LNC_0 the variable elimination rule [v] can be applied earlier. Let $\theta_1 \geq_R \theta_2$ denote the relation between substitutions θ_1 and θ_2 such that for any variable x, $\sigma\theta_1 x \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta_2 x$ for some substitution σ.

Lemma 9. Let R be an $\mathcal{L}_1$ program and G be an $\mathcal{L}_1$ goal $\Leftarrow s = x, E$ such that $x \notin Var(s)$. If there exists a refutation

$$\Pi : \; G \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square,$$

where $\theta\!\upharpoonright Var(G)$ is a normalizable substitution, then there exists a refutation

$$G \overset{\sigma_1(=\{s/x\})}{\underset{[v]}{\rightarrow}} \Leftarrow \sigma_1 E \overset{\sigma_2}{\twoheadrightarrow}_{LNC_0} \square$$

such that $\sigma_2\sigma_1 \geq_R \theta$. Moreover, the length of the refutation $\Leftarrow \sigma_1 E \overset{\sigma_2}{\twoheadrightarrow}_{LNC_0} \square$ is shorter than the length of the refutation Π.

(Proof) By Lemma 3 we may assume that the goal $s = x$ is selected in the first step of the refutation Π. Hence we have

$$G \overset{\theta_0}{\twoheadrightarrow}_{LNC_0} \Leftarrow \theta_0 E \overset{\theta_1}{\twoheadrightarrow}_{LNC_0} \square \tag{2}$$

such that $\theta_1\theta_0 = \theta$. Since $x \notin Var(s)$, the variable elimination rule [v] can be applied to $s = x$, and hence we obtain the derivation

$$G \overset{\sigma_1}{\underset{[v]}{\rightarrow}} \Leftarrow \sigma_1 E, \text{where } \sigma_1 = \{s/x\}. \tag{3}$$

Since $\theta_0 \upharpoonright Var(G)$ is a computed answer substitution of the goal $\Leftarrow s = x$ and R w.r.t. LNC_0, by Theorem 1 it is a correct answer substitution of the goal $\Leftarrow s = x$ and R. Moreover, since $\theta_0 \upharpoonright Var(G)$ is normalizable, by Lemma 8 we have $\theta_0 s \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta_0 x$. Since $s = \sigma_1 x$, we have $\theta_0\sigma_1 x \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta_0 x$. For any variable $z(\neq x)$, we have $\theta_0\sigma_1 z = \theta_0 z$. Hence, letting $\theta'_0 = \theta_0\sigma_1$, we obtain $\theta'_0 \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta_0$. Applying Lemma 7 to the derivation starting from $\theta_0 E$ in (2), we obtain

$$\Leftarrow \theta'_0 E \twoheadrightarrow^{\theta'_1}_{LNC_0} \Box, \text{where } \theta_1 \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta'_1. \tag{4}$$

Since $\sigma_1 \geq \theta'_0$, we can apply Lemma 2 to the refutation (4) and obtain

$$\Leftarrow \sigma_1 E \twoheadrightarrow^{\sigma_2}_{LNC_0} \Box, \tag{5}$$

where $\sigma_2 \geq \theta'_1\theta_0$. Combining the derivation (3) and the refutation (5) we obtain a refutation

$$G \twoheadrightarrow^{\sigma_2\sigma_1}_{LNC_0} \Box. \tag{6}$$

What remains to be proved is $\sigma_2\sigma_1 \geq_R \theta$. Since $\sigma_2 \geq \theta'_1\theta_0 \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta_1\theta_0 = \theta$, we obtain

$$\sigma_2 \geq_R \theta,$$

and then

$$\sigma_2\sigma_1 \geq_R \theta\sigma_1. \tag{7}$$

We will prove $\theta\sigma_1 \geq_R \theta$. We have

$$\begin{array}{lll} \theta\sigma_1 x & = \theta s & \text{since } \sigma_1 = \{s/x\} \\ & = \theta_1\theta_0 s & \text{since } \theta = \theta_1\theta_0 \\ & \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta_1\theta_0 x & \text{since } \theta_0 s \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta_0 x \\ & = \theta x & \text{since } \theta = \theta_1\theta_0, \end{array}$$

and hence we obtain

$$\theta\sigma_1 x \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta x. \tag{8}$$

For any variable $z \neq x$

$$\theta\sigma_1 z = \theta z. \tag{9}$$

From (8) and (9) we obtain

$$\theta\sigma_1 \leftrightarrow\!\!\!\!\twoheadrightarrow_R \theta. \tag{10}$$

From (7) and (10) we conclude that $\sigma_1\sigma_2 \geq_R \theta$.

Lemmas 2 and 7 state the preservation of the length of the refutations, and hence the refutation (5) is shorter than the length of the refutation (2).

∎

Now, we are ready to prove the completeness of LNC_1 via LNC_0.

Lemma 10.(completeness of LNC_1 w.r.t. LNC_0) Let R be an $\mathcal{L}_1$ program and G be an $\mathcal{L}_1$ goal. If there exists a refutation $G \twoheadrightarrow^{\theta}_{LNC_0} \Box$ where $\theta \upharpoonright Var(G)$ is a normalizable constructor term substitution, then there exists a refutation $Basic(G) \twoheadrightarrow^{\theta'}_{LNC_1} \Box$ such that $\theta' \upharpoonright Var(G) \geq_R \theta \upharpoonright Var(G)$.

(Proof) We use the following sublemma.

Sublemma: Let G_0 be an $\mathcal{L}_1$ goal. Suppose that there exists a refutation $\Pi : G_0 \twoheadrightarrow^{\theta_0}_{LNC_0} \square$ where (i) $\theta_0{\uparrow}Var(G)$ is a normalizable constructor term substitution, and (ii) for all the equations of the form $s = x$ of G_0, x is in $Var(G)$. Then there exists a refutation $Basic(G_0) \twoheadrightarrow^{\theta'_0}_{LNC_1} \square$ such that $\theta'_0{\uparrow}Var(G) \geq_R \theta_0{\uparrow}Var(G)$.

In particular, when Π is $G \twoheadrightarrow^{\theta}_{LNC_0} \square$, the sublemma gives Lemma 10. The above condition (ii) is necessary to prove the cases [2] and [4].

Proof of the sublemma: The proof is by the induction on the length k of the refutation Π. For $k = 0$, the result immediately holds. For $k > 0$, we have to consider the following five cases according to the rules used in the first step of the refutation Π. From Lemma 3, it suffices to consider the case that the leftmost atom is selected in the first step of Π.

[1] (when the first step of Π is [d]) Since we are considering $\mathcal{L}_1$ goals, by Proposition 1 (1) G_0 is necessarily of the form $\Leftarrow c(d_1, \ldots, d_n) = c(d_1, \ldots, d_n), E$, where $d_1, \ldots, d_n$ are ground data terms. In this case Π is

$$\Leftarrow \mathbf{c(d_1, \ldots, d_n) = c(d_1, \ldots, d_n)}, E \rightarrow_{[d]} \Leftarrow d_1 = d_1, \ldots, d_n = d_n, E \twoheadrightarrow^{\theta_0}_{LNC_0} \square.$$

$Basic(G_0)$ is $\Leftarrow c(d_1, \ldots, d_n) = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!]$. Hence, we have a corresponding derivation w.r.t. LNC_1:

$$\begin{array}{ll} & \Leftarrow c(d_1, \ldots, d_n) = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!] \\ \rightarrow^{\sigma_1}_{[u]} & \Leftarrow \sigma_1(\Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!]) \text{ where } \sigma_1 = \{d_1/z_1, \ldots, d_n/z_n\} \\ & \text{(which is the same as } \Leftarrow \Phi_3[\![d_1 = d_1, \ldots, d_n = d_n, E]\!] \\ & \text{since } \sigma_1 \text{ does not affect } d_1, \ldots, d_n, E) \end{array}$$

Since (i) and (ii) hold for the refutation $\Leftarrow d_1 = d_1, \ldots, d_n = d_n, E \twoheadrightarrow^{\theta_0}_{LNC_0}$, by the induction hypothesis we obtain a refutation $Basic(G_0) \twoheadrightarrow^{\theta_1\sigma_1}_{LNC_1} \square$, where θ_1 is a substitution such that $\theta_1{\uparrow}Var(G) \geq_R \theta_0{\uparrow}Var(G)$. Let $\theta'_0 = \theta_1\sigma_1$. We see that $\theta'_0{\uparrow}Var(G) = \theta_1{\uparrow}Var(G) \geq_R \theta_0{\uparrow}Var(G)$.

[2] (when the first step of Π is [v]) Let G_0 be $\Leftarrow s = d, E$. The case that s is a function term and d is a variable is impossible by the assumptions (i) and (ii). Therefore, it suffices to consider the following two cases; the cases that s is a variable and d is a data term, and that s is a constructor term and d is a variable. We only prove the former case. The later case is trivial.

We write s as x and d as $c(d_1, \ldots, d_n)$. In this case Π is

$$\Leftarrow \mathbf{x = c(d_1, \ldots, d_n)}, E \rightarrow^{\sigma}_{[v]} \Leftarrow \sigma E \twoheadrightarrow^{\theta_1}_{LNC_0} \square,$$

where $\sigma = \{c(d_1, \ldots, d_n)/x\}$ and θ_1 is a substitution such that $\theta_1\sigma = \theta_0$. $Basic(G_0)$ is $\Leftarrow x = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!]$. We have a corresponding derivation w.r.t. LNC_1:

$$\begin{array}{lll} & \Leftarrow x = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!] & \\ \rightarrow^{\sigma_1}_{[vd]} & \Leftarrow \sigma_1(\Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!]) \text{ where } \sigma_1 = \{c(z_1, \ldots, z_n)/x\} & \\ & \text{(which is the same as } \Leftarrow \Phi_3[\![z_1 = d_1]\!], \ldots, \Phi_3[\![z_n = d_n]\!], \sigma_1\Phi_3[\![E]\!] & \\ & \quad \text{since } \sigma_1 \text{ does not affect } d_1, \ldots, d_n) & \\ \twoheadrightarrow^{\sigma_2}_{LNC_1} & \Leftarrow \sigma_2\sigma_1\Phi_3[\![E]\!] \text{ where } \sigma_2{\uparrow}\{z_1, \ldots, z_n\} = \{d_1/z_1, \ldots, d_n/z_n\} & \text{by Proposition 2} \\ & \text{(which is the same as } \Leftarrow \sigma\Phi_3[\![E]\!]). & \end{array}$$

By Proposition 1 (2), x does not appear in the RHSs of the equations of E. Hence, $\Leftarrow \sigma\Phi_3[\![E]\!]$ is the same as $\Leftarrow \Phi_3[\![\sigma E]\!]$. Since σx is a constructor term, (i) and (ii) hold for the refutation $\Leftarrow \sigma E \twoheadrightarrow^{\theta_1}_{LNC_0} \square$. Hence, by the induction hypothesis we obtain $Basic(G_0) \twoheadrightarrow^{\theta'_1\sigma_2\sigma_1}_{LNC_1} \square$, where $\theta'_1\uparrow Var(G) \geq_R \theta_1\uparrow Var(G)$. Let $\theta'_0 = \theta'_1\sigma_2\sigma_1$. We see that $\theta'_0\uparrow Var(G) \geq_R (\theta_1\sigma_2\sigma_1)\uparrow Var(G) = \theta_0\uparrow Var(G)$.

[3] (when the first step of Π is [on]) Since G_0 is an $\mathcal{L}_1$ goal, by Proposition 1 (1) the RHSs of the equations of G_0 are not function terms. Hence it suffices to consider the case that G_0 is $\Leftarrow f(s_1,\ldots,s_n) = c(e_1,\ldots,e_n), E$. In this case Π is

$$\begin{array}{rl} & \Leftarrow \mathbf{f(s_1,\ldots,s_n) = c(e_1,\ldots,e_n)}, E \\ \rightarrow_{[on]} & \Leftarrow s_1 = d_1,\ldots,s_n = d_n, F, t = c(e_1,\ldots,e_n), E \\ \twoheadrightarrow^{\theta_0}_{LNC_0} & \square, \end{array}$$

where a new variant $f(d_1,\ldots,d_n) = t \Leftarrow F$ of an $\mathcal{L}_1$ conditional equation in R is used. Assume that d_j is a variable. For simplicity we consider only the case that d_j is a variable and the others are non-variable terms. We write d_j as x_j. By Lemma 9, we have

$$\begin{array}{rl} & \Leftarrow s_1 = d_1,\ldots,\mathbf{s_j = x_j},\ldots,s_n = d_n, F, t = c(e_1,\ldots,e_n), E \\ \overset{\sigma=\{s_j/x_j\}}{\rightarrow_{[v]}} & \Leftarrow \sigma(s_1 = d_1,\ldots,s_{j-1} = d_{j-1}, s_{j+1} = d_{j+1},\ldots,s_n = d_n, F, t = c(e_1,\ldots,e_n), E) \\ \twoheadrightarrow^{\theta_1}_{LNC_0} & \square, \end{array}$$

where θ_1 is a substitution such that $\theta_1\sigma \geq_R \theta_0$. Since σ does not affect $d_1,\ldots,d_{j-1},d_{j+1},\ldots,d_n$, $s_1,\ldots,s_{j-1},s_{j+1},\ldots,s_n$, $e_1,\ldots,e_n$ and E, the goal $\Leftarrow \sigma(s_1 = d_1,\ldots,s_{j-1} = d_{j-1}, s_{j+1} = d_{j+1},\ldots,s_n = d_n, F, t = c(e_1,\ldots,e_n), E)$ is the same as $\Leftarrow s_1 = d_1,\ldots,s_{j-1} = d_{j-1}, s_{j+1} = d_{j+1},\ldots,s_n = d_n, \sigma F, \sigma t = c(e_1,\ldots,e_n), E$. $Basic(G_0)$ is $\Leftarrow f(s_1,\ldots,s_n) = c(z_1,\ldots,z_n), \Phi_3[\![z_1 = e_1,\ldots,z_n = e_n, E]\!]$. Using a new variant $f(y_1,\ldots,x_j,\ldots,y_n) = t \Leftarrow \Phi_2[\![y_1 = d_1,\ldots,y_{j-1} = d_{j-1}, y_{j+1} = d_{j+1},\ldots,y_n = d_n, F]\!]$ of a basic $\mathcal{L}_1$ conditional equation in $Basic(R)$ we obtain a corresponding derivation w.r.t. LNC_1:

$$\begin{array}{rl} & \Leftarrow f(s_1,\ldots,s_n) = c(z_1,\ldots,z_n), \Phi_3[\![z_1 = e_1,\ldots,z_n = e_n, E]\!] \\ \overset{\sigma_1}{\rightarrow_{[ln]}} & \Leftarrow \sigma_1(\Phi_2[\![y_1 = d_1,\ldots,y_{j-1} = d_{j-1}, y_{j+1} = d_{j+1},\ldots,y_n = d_n, F]\!], \\ & t = c(z_1,\ldots,z_n), \Phi_3[\![z_1 = e_1,\ldots,z_n = e_n, E]\!]), \end{array}$$

where $\sigma_1 = \{s_1/y_1,\ldots,s_{j-1}/y_{j-1}, s_j/x_j, s_{j+1}/y_{j+1},\ldots,s_n/y_n\}$. Since σ_1 does not affect d_1, ..., d_{j-1}, d_{j+1}, ..., d_n, $e_1,\ldots,e_n, E$ and the RHSs of F, $\Leftarrow \sigma_1(\Phi_2[\![y_1 = d_1,\ldots,y_{j-1} = d_{j-1}, y_{j+1} = d_{j+1},\ldots,y_n = d_n, F]\!], t = c(z_1,\ldots,z_n), \Phi_3[\![z_1 = e_1,\ldots,z_n = e_n, E]\!])$ is the same as $\Leftarrow \Phi_2[\![s_1 = d_1,\ldots,s_{j-1} = d_{j-1}, s_{j+1} = d_{j+1},\ldots,s_n = d_n, \sigma_1 F]\!], \sigma_1 t = c(z_1,\ldots,z_n), \Phi_3[\![z_1 = e_1,\ldots,z_n = e_n]\!], \Phi_3[\![E]\!]$. Since $\sigma_1\uparrow\{x_j\} = \sigma$, this is the same as $\Leftarrow \Phi_2[\![s_1 = d_1,\ldots,s_{j-1} = d_{j-1}, s_{j+1} = d_{j+1},\ldots,s_n = d_n, \sigma F]\!], \Phi_3[\![\sigma t = c(e_1,\ldots,e_n), E]\!]$. Furthermore, since the RHSs of F and $d_1,\ldots,d_{j-1},d_{j+1},\ldots,d_n$ contain no variable, this is the same as $\Leftarrow \Phi_3[\![s_1 = d_1,\ldots,s_{j-1} = d_{j-1}, s_{j+1} = d_{j+1},\ldots,s_n = d_n, \sigma F, \sigma t = c(e_1,\ldots,e_n), E]\!]$. σ does not affect $e_1,\ldots,e_n$ and E, and the RHSs of F and $d_1,\ldots,d_{j-1},d_{j+1},\ldots,d_n$ are non-variable terms. Therefore, (i) and (ii) hold for the refutation $\Leftarrow s_1 = d_1,\ldots,s_{j-1} = d_{j-1}, s_{j+1} = d_{j+1},\ldots,s_n = d_n, \sigma F, \sigma t = c(e_1,\ldots,e_n), E \twoheadrightarrow^{\theta_1}_{LNC_0} \square$. By the induction hypothesis we obtain a refutation $\Leftarrow Basic(G_0) \twoheadrightarrow^{\theta'_1\sigma_1}_{LNC_1} \square$, where $\theta'_1\uparrow Var(G) \geq_R \theta_1\uparrow Var(G)$. Let $\theta'_0 = \theta'_1\sigma_1$. We see that $\theta'_0\uparrow Var(G) \geq_R \theta_1\uparrow Var(G) = \theta_1\sigma\uparrow Var(G) \geq_R \theta_0\uparrow Var(G)$.

[4] (when the first step of Π is [im]) By a similar observation to the case [2], we see that G_0 contains no equation whose LHS is a function term and whose RHS is a variable. Hence it suffices to consider the following two cases; the cases that G_0 is of the form $\Leftarrow x = c(d_1, \ldots, d_n), E$ and that G_0 is of the form $\Leftarrow c(s_1, \ldots, s_n) = x, E$. We only prove the former case. The later case is trivial.

In this case Π is

$$\begin{array}{ll} & \Leftarrow \mathbf{x = c(d_1, \ldots, d_n)}, E \\ \rightarrow^{\sigma}_{[\mathrm{im}]} & \Leftarrow \sigma(d_1 = x_1, \ldots, d_n = x_n, E) \\ \twoheadrightarrow^{\theta_1}_{LNC_0} & \square, \end{array}$$

where $\sigma = \{c(x_1, \ldots, x_n)/x\}$ and $\theta_1\sigma = \theta_0$. By Lemma 9, we have

$$\begin{array}{ll} & \Leftarrow \sigma(\mathbf{d_1 = x_1}, \ldots, \mathbf{d_n = x_n}, E) \\ \twoheadrightarrow^{\sigma'}_{LNC_0} & \Leftarrow \sigma'\sigma E \\ \twoheadrightarrow^{\theta_2}_{LNC_0} & \square, \end{array}$$

where $\sigma' = \{d_1/x_1, \ldots, d_n/x_n\}$ and θ_2 is a substitution such that $\theta_2\sigma' \geq_R \theta_1$. $Basic(G_0)$ is $\Leftarrow x = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!]$. We obtain a corresponding derivation w.r.t. LNC_1:

$$\begin{array}{ll} & \Leftarrow x = c(z_1, \ldots, z_n), \Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!] \\ \rightarrow^{\sigma_1}_{[\mathrm{vd}]} & \Leftarrow \sigma_1(\Phi_3[\![z_1 = d_1, \ldots, z_n = d_n, E]\!]) \text{ where } \sigma_1 = \{c(z_1, \ldots, z_n)/x\} \\ & (\text{which is the same as } \Leftarrow \Phi_3[\![z_1 = d_1]\!], \ldots, \Phi_3[\![z_n = d_n]\!], \sigma_1\Phi_3[\![E]\!] \\ & \text{since } \sigma_1 \text{ does not affect } d_1, \ldots, d_n) \\ \twoheadrightarrow^{\sigma_2}_{LNC_1} & \Leftarrow \sigma_2\sigma_1\Phi_3[\![E]\!] \text{ where } \sigma_2\!\uparrow\!\{z_1, \ldots, z_n\} = \{d_1/z_1, \ldots, d_n/z_n\} \qquad \text{by Proposition 2} \\ & (\text{which is the same as } \Leftarrow \sigma'\sigma\Phi_3[\![E]\!]) \end{array}$$

By Proposition 1 (2), x does not appear in the RHSs of the equations of E. Hence, $\Leftarrow \sigma'\sigma\Phi_3[\![E]\!]$ is the same as $\Leftarrow \Phi_3[\![\sigma'\sigma E]\!]$. Since $\sigma'\sigma x$ is a constructor term, (i) and (ii) hold for the refutation $\Leftarrow \sigma\sigma'E \twoheadrightarrow^{\theta_2}_{LNC_0} \square$. Hence, by the induction hypothesis we obtain $Basic(G_0) \twoheadrightarrow^{\theta_2'\sigma_2\sigma_1}_{LNC_1} \square$, where $\theta_2'\!\uparrow Var(G) \geq_R \theta_2\!\uparrow Var(G)$. Let $\theta_0' = \theta_2'\sigma_2\sigma_1$. We see that $\theta_0'\!\uparrow Var(G) \geq_R (\theta_2\sigma'\sigma)\!\uparrow Var(G) \geq_R (\theta_1\sigma)\!\uparrow Var(G) = \theta_0\!\uparrow Var(G)$.

[5] (when the first step of Π is [t]) Since we consider $\mathcal{L}_1$ goals, an inference step by [t] in a derivation w.r.t. LNC_0 is simulated by several inference steps by [d]. Hence the result follows by the case [1].

∎

By Theorem 4 and Lemma 10, we obtain the completeness of LNC_1.

Theorem 6. (completeness of LNC_1) Let R be an $\mathcal{L}_1$ program and G be an $\mathcal{L}_1$ goal. For any correct answer substitution θ which is a normalizable constructor term substitution, there exists a computed answer substitution θ' of G and R w.r.t. LNC_1 such that $\theta' \geq_R \theta$.

Remark 4. In Theorem 6, $\theta' \geq_R \theta$ can be sharpened to $\gamma\theta' \twoheadrightarrow_R \theta\!\downarrow$ for some substitution γ.

The following example shows that the conditions C3-2 and C3-3 are necessary for the completeness of LNC_1.

Example 5. Let R be an $\mathcal{L}_1$ program $\{g = d, h = d, f = c'(c(g), c(h))\}$, where c and c' are constructor symbols and d is a data term. Let G be an $\mathcal{L}_1$ goal $\Leftarrow c(g) = x, c(h) = x$. G does not satisfy the condition C3-2. $\{c(d)/x\}$ is a correct answer substitution of G and R which is a normalizable constructor term substitution. The refutation of $Basic(G)$ and $Basic(R)$ w.r.t. LNC_1 would fail, however:

$$\begin{array}{rl} & \Leftarrow c(g) = x, c(h) = x \\ \overset{\{c(g)/x\}}{\underset{[vc]}{\rightarrow}} & \Leftarrow c(h) = c(g). \end{array}$$

Let G' be an $\mathcal{L}_1$ goal $\Leftarrow f = c'(x, x)$. G' does not satisfy the condition C3-3. $\{c(d)/x\}$ is a correct answer substitution of G' and R which is a normalizable constructor term substitution. The refutation of $Basic(G')$ and $Basic(R)$ w.r.t. LNC_1 would fail:

$$\begin{array}{rl} & \Leftarrow f = c'(z, w), z = x, w = x \\ \overset{\varepsilon}{\underset{[ln]}{\rightarrow}} & \Leftarrow c'(c(g), c(h)) = c'(z, w), z = x, w = x \\ \overset{\{c(g)/z, c(h)/w\}}{\underset{[u]}{\rightarrow}} & \Leftarrow c(g) = x, c(h) = x \\ \overset{\{c(g)/x\}}{\underset{[vc]}{\rightarrow}} & \Leftarrow c(h) = c(g). \end{array}$$

7 Concluding remarks

We have presented two lazy narrowing calculi for lazy functional-logic programming languages. The soundness and completeness of the lazy narrowing calculi are proved. We first showed a lazy narrowing calculus LNC_0 which defines a complete equation solving procedure for a conditional equational system. The non-deterministic choice of inference rules in the refutation of a goal w.r.t. LNC_0 will create a large search space in practice. Hence we are lead to a more efficient calculus LNC_1 at the cost of generality. The loss of generality is not a severe limitation since we are interested in a programming language, not in an equational theorem proving.

The language basic $\mathcal{L}_1$ of the calculus LNC_1 imposes several restrictions on the syntax of equations and on the variable occurrences. Some of those restrictions can be lifted by the program transformation. LNC_1 enables determinate choice of inference rules in the refutation. Hence it can be implemented efficiently on a conventional machine.

$\mathcal{L}_1$ can be a first-order functional-logic programming language. It (when the operational semantics is given by LNC_1) shares features with K-LEAF. However, $\mathcal{L}_1$ has a simpler semantics solely based on the narrowing calculus LNC_1. It is different from K-LEAF in that we allow equations in the body of conditional equations or in a goal.

References

[1] J. A. Bergstra and J. W. Klop. Conditional rewrite rules: confluence and termination. *J. Comput. Syst. Sci.*, 32:323–362, 1986.

[2] P. G. Bosco, C. Cecchi, E.Giovannetti, C. Moiso, and C. Palamidessi. Using resolution for a sound and efficient integration of logic and functional programming. In *Languages for Parallel Architectures*, chapter 4. John Wiley & Sons, 1989.

[3] P. G. Bosco, C. Cecchi, and C. Moiso. An extension of WAM for K-LEAF: a WAM-based compilation of conditional narrowing. In *Proc. 6th Int. Conf. Logic Prog., Lisboa*, pages 318–333, 1989.

[4] P. G. Bosco, E. Giovannetti, and C. Moiso. Narrowing vs. SLD-resolution. *Theor. Comput. Sci.*, (59):3–23, 1988.

[5] M. M. T. Chakravarty and H. C. R. Lock. The implementation of lazy narrowing. In *Proc. 3rd. PLILP'91*, pages 123–134, 1991. LNCS 528.

[6] P. H. Cheong. *Compiling lazy narrowing into Prolog.* Report delivered for the ESPRIT basic research action No.3020, 1990.

[7] J. Darlington and Y. K. Guo. Narrowing and unification in functional programming – an evaluation mechanism for absolute set abstraction. In *Proc. RTA '89*, pages 92–108, 1988. LNCS 355.

[8] N. Dershowitz and M. Okada. Conditional equational programming and the theory of conditional term rewriting. In *Proc. Int. Conf. 5th Generation Comp. Syst.*, pages 337–346, 1988.

[9] E.Giovannetti and C. Moiso. A completeness result for E-unification algorithms based on conditional narrowing. In *Foundation of Logic and Functional Programming*, pages 157–167, 1988. LNCS 306.

[10] L. Fribourg. SLOG: a logic programming languages interpreter based on clausal superposition and rewriting. In *Proc. 1985 Symp. Logic. Prog.*, pages 172–184, 1985.

[11] E. Giovannetti, G. Levi, C. Moiso, and C. Palamidessi. Kernel-LEAF: a logic plus functional language. *J. Compt. Syst. Sci.*, 42(2):139–185, 1991.

[12] J. A. Goguen and J. Meseguer. EQLOG: equality, types, and generic modules for logic programming. In DeGroot and Lindstrom, editors, *Logic Programming*, pages 295–364. Prentice Hall, 1986.

[13] Michael Hanus. Efficient implementation of narrowing and rewriting. In *Proc. Int. Workshop on Processing Declarative Knowledge.* To appear in LNCS.

[14] S. Hölldobler. Foundations of equational logic programming. *LNAI*, 353, 1989.

[15] G. Huet and D. C. Oppen. Equations and rewrite rules. In Book, editor, *Formal Languages: Perspectives and Open Problems.* Academic Press, 1980.

[16] J. Hullot. Canonical forms and unifications. In *5th. CADE*, pages 318–334, 1980. LNCS 87.

[17] M. Rodríguez-Artalejo J. J. Moreno-Navarro. BABEL: a functional and logic programming language based on constructor discipline and narrowing. In *Algebraic and Logic Programming*, pages 223–232, 1988. LNCS 343.

[18] H. Kuchen, R. Loogen, J. J. Moreno-Navarro, and Mario Rodríguez-Artalejo. Graph-based implementation of a functional logic languages. In *ESOP '90*, pages 271–290, 1990. LNCS 432.

[19] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1987.

[20] G. Mansfield, A. Togashi, and S. Noguchi. AMLOG: an amalgamated equational logic programming language. *J. Inf. Process.*, 11(4):278–287, 1988.

[21] A. Martelli and U. Montanari. An efficient unification algorithm. *ACM Trans. Prog. Lang. Syst.*, 4(2):258–282, 1982.

[22] S. Okui and T. Ida. *Narrowing calculi for lazy functional-logic programming languages* (in Japanese). submitted for publication.

[23] U. S. Reddy. Narrowing as the operational semantics of functional languages. In *Proc. 1985 Symp. Logic. Prog.*, pages 138–151, 1985.

[24] J. R. Slagel. Automatic theorem proving in theories with simplifiers, commutativity and associativity. *J. ACM*, 21:622–642, 1974.

[25] M. H. van Emden and J. W. Lloyd. A logical reconstruction of Prolog II. *J. Logic Prog.*, 4:265–288, 1974.

[26] M. H. van Emden and K. Yukawa. Logic programming with equations. *J. Logic Prog.*, 1:143–149, 1987.

A Proof of the completeness of LNC_0 w.r.t. NC

A.1 Proof of Lemma 2

Lemma 2.(lifting lemma for LNC_0) Let R be a conditional equational system, G be a goal and γ be a substitution. If there exists a refutation $\gamma G \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square$, then there exists a refutation $G \overset{\theta'}{\twoheadrightarrow}_{LNC_0} \square$ such that $\theta' \geq \theta\gamma$, and whose length is the same as the length of the refutation $\gamma G \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square$.

(Proof) The proof is by the induction on the length k of the refutation of γG. For $k = 0$, the result obviously holds. For $k > 0$, let G be $\Leftarrow E, s = t, E'$, and suppose that we have a refutation

$$\Pi : \Leftarrow \gamma(E, \mathbf{s = t}, E') \overset{\theta}{\twoheadrightarrow}_{LNC_0} \square.$$

We have to consider the following five cases according to the rules used in the first step of the refutation.

[1](when the first step is [on]) Let $s = t$ be $f(s_1, \ldots, s_n) \doteq t$. The refutation Π is the following, in which a new variant $f(t_1, \ldots, t_n) = r \Leftarrow F$ of a conditional equation in R is used

$$\begin{array}{ll} & \Leftarrow \gamma(E, \mathbf{f(s_1, \ldots, s_n) \doteq t}, E') \\ \rightarrow_{[\mathrm{on}]} & \Leftarrow \gamma(E, s_1 = t_1, \ldots, s_n = t_n, F, t = r, E') \\ \overset{\theta}{\twoheadrightarrow}_{LNC_0} & \square. \end{array} \tag{11}$$

Using the same conditional equation, we have a derivation:

$$\begin{array}{ll} & \Leftarrow E, \mathbf{f(s_1, \ldots, s_n) \doteq t}, E' \\ \rightarrow_{[\mathrm{on}]} & \Leftarrow E, s_1 = t_1, \ldots, s_n = t_n, F, t = r, E'. \end{array}$$

The result follows by the induction hypothesis applied to the refutation starting from the goal in (11).

[2](when the first step is [v]) Let $s = t$ be $s \doteq x$. The refutation Π is the following.

$$\begin{aligned} & \Leftarrow \gamma(E, \mathbf{s} \doteq \mathbf{x}, E') && (12)\\ \rightarrow^{\sigma}_{[\mathrm{v}]} \quad & \Leftarrow \sigma\gamma(E, E') && (13)\\ \rightarrow\!\!\!\rightarrow^{\theta_1}_{LNC_0} \quad & \square, \end{aligned}$$

where, for $y = \gamma x$, $y \notin Var(\gamma s)$ and $\sigma = \{\gamma s/y\}$ and θ_1 is a substitution such that

$$\theta_1\sigma = \theta. \tag{14}$$

We see $x \notin Var(s)$ since otherwise we would have $\gamma x \in Var(\gamma s)$, contradicting $y(= \gamma x) \notin Var(\gamma s)$. Hence there is a derivation

$$\Leftarrow E, \mathbf{s} \doteq \mathbf{x}, E' \rightarrow^{\sigma'}_{[\mathrm{v}]} \sigma'(E, E'),$$

where $\sigma' = \{s/x\}$, which corresponds to the derivation from (12) to the goal $\Leftarrow \sigma\gamma(E, E')$ in (13). Since $\sigma' x = s$ and $\sigma\gamma x = \sigma y = \gamma s$, and for variable $z \neq x$, $\sigma' z = z$, we have

$$\sigma' \geq \sigma\gamma. \tag{15}$$

Let

$$\gamma'\sigma' = \sigma\gamma \tag{16}$$

for some substitution γ'. By the induction hypothesis, we have a following lifted refutation of $\Leftarrow \sigma'(E, E')$ corresponding to $\sigma\gamma(E, E') \rightarrow\!\!\!\rightarrow^{\theta_1}_{LNC_0} \square$:

$$\Leftarrow \sigma'(E, E) \rightarrow\!\!\!\rightarrow^{\theta'_1}_{LNC_0} \square$$

such that

$$\theta'_1 \geq \theta_1\gamma'. \tag{17}$$

Let θ' be the substitution obtained in the lifted refutation $\Leftarrow E, s \doteq x, E' \rightarrow\!\!\!\rightarrow^{\theta'}_{LNC_0} \square$. We obtain

$$\begin{array}{lll} \theta' & = \theta'_1\sigma' & \text{by definition}\\ & \geq \theta_1\gamma'\sigma' & \text{by (17)}\\ & = \theta_1\sigma\gamma & \text{by (16)}\\ & = \theta\gamma & \text{by (14).} \end{array}$$

[3] (when the first step is [im]) The refutation Π is the following.

$$\begin{aligned} & \Leftarrow \gamma(E, \mathbf{f}(\mathbf{s_1}, \ldots, \mathbf{s_n}) \doteq \mathbf{x}, E')\\ \rightarrow^{\sigma}_{[\mathrm{im}]} \quad & \Leftarrow \sigma\gamma(E, s_1 = x_1, \ldots, s_n = x_n, E') && (18)\\ \rightarrow\!\!\!\rightarrow^{\theta_1}_{LNC_0} \quad & \square, \end{aligned}$$

where $\sigma = \{f(x_1, \ldots, x_n)/y\}$ for $y = \gamma x$ and $x_1, \ldots, x_n$ are fresh variables.

We obtain a derivation:

$$\begin{aligned} & \Leftarrow E, \mathbf{f}(\mathbf{s_1}, \ldots, \mathbf{s_n}) \doteq \mathbf{x}, E'\\ \rightarrow^{\sigma'}_{[\mathrm{im}]} \quad & \sigma'(E, s_1 = x_1, \ldots, s_n = x_n, E'), \end{aligned}$$

where $\sigma' = \{f(x_1, \ldots, x_n)/x\}$. By the similar reasoning as we had in obtaining (15) in the case [2], we have $\sigma' \geq \sigma\gamma$. Therefore, by the induction hypothesis applied to the refutation starting from the goal in (18), we obtain a desired refutation $\Leftarrow E, f(s_1, \ldots, s_n) \doteq x, E' \twoheadrightarrow^{\theta'}_{LNC_0} \square$ such that $\theta' \geq \theta\gamma$.

[4,5] (when the first step is [t] or [d]) In these cases the result is obvious.

Finally, in each case, we see that the length of the refutations is preserved on lifting.

∎

A.2 Proof of Lemma 3

Lemma 3.(switching lemma for LNC_0) Let R be a conditional equational system. If there exists a refutation $\Pi : \Leftarrow E, \mathbf{s} = \mathbf{t}, s' = t', E' \twoheadrightarrow^{\theta}_{LNC_0} \square$ where $s = t$ is selected in the first step and $s' = t'$ in the second step of the refutation, then there exists a refutation $\Leftarrow E, s = t, \mathbf{s}' = \mathbf{t}', E' \twoheadrightarrow^{\theta}_{LNC_0} \square$, which is the same as Π except that the first two selections are switched, and whose length is the same as the length of the refutation Π.

(Proof) When either the first or the second step of Π is [on], [d] or [t], the first and the second steps can be swapped since [on], [d] or [t] does not produce a new substitution. Therefore, it suffices to consider the following four cases:

1. The first and the second steps are [v].
2. The first step is [v], and the second step is [im].
3. The first step is [im], and the second step is [v].
4. The first and the second steps are [im].

We only prove case 1. Other cases are treated similarly.

(Proof of the case 1) It suffices to consider the case that s and s' are distinct variables. Let $s = t$ be $x \doteq t$ and $s' = t'$ be $y \doteq t'$ where $x \neq y$. We have to consider the following two cases according to whether y appears in t or not.

[1] (when $y \in Var(t)$) In this case $x \notin Var(t')$. The first two steps of the refutation Π are

$$\begin{aligned} & \Leftarrow E, \mathbf{x} \doteq \mathbf{t}, y \doteq t', E' \\ \xrightarrow{\sigma_1}_{[\mathrm{v}]} \; & \Leftarrow \sigma_1(E, \mathbf{y} \doteq \mathbf{t}', E') \\ \xrightarrow{\sigma_2}_{[\mathrm{v}]} \; & \Leftarrow \sigma_2\sigma_1(E, E'), \end{aligned}$$

where $x \notin Var(t)$, $\sigma_1 = \{t/x\}$, $y \notin Var(\sigma_1 t')$ and $\sigma_2 = \{\sigma_1 t'/\sigma_1 y\} = \{t'/y\}$. We see that $x \notin Var(\sigma_2 t)$ since $x \notin Vcod(\sigma_2)$ and $x \notin Var(t)$. Hence let $\sigma_3 = \{\sigma_2 t/x\}$. We obtain a derivation

$$\begin{aligned} & \Leftarrow E, x \doteq t, \mathbf{y} \doteq \mathbf{t}', E' \\ \xrightarrow{\sigma_2}_{[\mathrm{v}]} \; & \Leftarrow \sigma_2(E, \mathbf{x} \doteq \mathbf{t}, E') \\ \xrightarrow{\sigma_3}_{[\mathrm{v}]} \; & \Leftarrow \sigma_3\sigma_2(E, E') \end{aligned}$$

Since $\sigma_2\sigma_1 = \{\sigma_2 t/x, t'/y\} = \sigma_3\sigma_2$, we see that the first and second steps can be swapped.

[2] (when $y \notin Var(t)$) In this case the first two steps of the refutation Π are

$$\begin{aligned} & \Leftarrow E, \mathbf{x} \doteq \mathbf{t}, y \doteq t', E' \\ \xrightarrow{\sigma_1}_{[\mathrm{v}]} \; & \Leftarrow \sigma_1(E, \mathbf{y} \doteq \mathbf{t}', E') \\ \xrightarrow{\sigma_2}_{[\mathrm{v}]} \; & \Leftarrow \sigma_2\sigma_1(E, E'), \end{aligned}$$

where $x \notin Var(t)$, $\sigma_1 = \{t/x\}$, $y \notin Var(\sigma_1 t')$ and $\sigma_2 = \{\sigma_1 t'/\sigma_1 y\} = \{\sigma_1 t'/y\}$. We see that $y \notin Var(t')$, since otherwise $y \in Var(\sigma_1 t')$ as σ_1 does not affect the occurrences of y nor introduces y, and this contradicts $y \notin Var(\sigma_1 t')$. Hence let $\sigma_3 = \{t'/y\}$. We obtain a derivation

$$\begin{array}{ll} & \Leftarrow E, x \doteq t, \mathbf{y} \doteq \mathbf{t}', E' \\ \overset{\sigma_3}{\rightarrow}_{[v]} & \Leftarrow \sigma_3(E, \mathbf{x} \doteq \mathbf{t}, E') \\ \overset{\sigma_1}{\rightarrow}_{[v]} & \Leftarrow \sigma_1\sigma_3(E, E'), \end{array}$$

Since $\sigma_2\sigma_1 = \{t/x, \sigma_1 t'/y\} = \sigma_1\sigma_3$, we see that the first and second steps can be swapped.

It is obvious that in both cases the length of the refutations is not changed on switching.

■

A.3 Proof of Lemma 4

We prove the completeness of LNC_0 by relating refutations w.r.t. NC to corresponding refutations w.r.t. LNC_0. Since one step derivation by the narrowing rule [n] is not directly related to a derivation w.r.t. LNC_0, we need the following Lemma 11 which states that one step derivation by the narrowing rule [n] followed by a refutation w.r.t. LNC_0 is related to a corresponding refutation w.r.t. LNC_0.

Lemma 11 Let R be a conditional equational system and G be a goal $\Leftarrow E, s = t, E'$. If there exists a derivation of G and R, w.r.t. NC

$$G \overset{\sigma}{\rightarrow}_{[n]} \Leftarrow \sigma(E, F, s[u \leftarrow r] = t, E')$$

where a new variant $l = r \Leftarrow F$ of a conditional equation in R is used, and a refutation w.r.t. LNC_0

$$\Leftarrow \sigma(E, F, s[u \leftarrow r] = t, E') \twoheadrightarrow^{\theta_1}_{LNC_0} \square,$$

then there exists a refutation of G and R w.r.t. LNC_0

$$G \twoheadrightarrow^{\theta'}_{LNC_0} \square$$

such that $\theta' \geq \theta_1\sigma$.

(Proof) We first define a special derivation to be called quasi-derivation. A quasi-derivation of a goal G is a combination of the derivation $G \rightarrow_{[n]} G'$ and the derivation $G' \twoheadrightarrow_{LNC_0} G''$. We write $G \rightarrow_{[n]} G' \twoheadrightarrow_{LNC_0} G''$ as in ordinary derivation. When G'' is $\square$, a quasi-derivation of G is called a quasi-refutation (of G). Let Π be a quasi-refutation

$$\Leftarrow E, s \doteq t, E' \overset{\sigma}{\rightarrow}_{[n]} \Leftarrow \sigma(E, F, s[u \leftarrow r] = t, E') \underbrace{\twoheadrightarrow^{\theta_1}_{LNC_0}}_{l\ (\geq 1)\ \text{steps}} \square.$$

We associate the pair (u, l) with the quasi-refutation Π, where u is the occurrence of a subterm at which the narrowing rule [n] is applied in Π and l is the length of the refutation of $\Leftarrow \sigma(E, s[u \leftarrow r] = t, F, E')$ w.r.t. LNC_0 in Π. We define an ordering $\ll$ on the pairs as follows. For any occurrences u, u' and any natural numbers l, l', $(u, l) \ll (u', l')$ iff $u \prec u'$, or $u = u'$ and $l < l'$. Note that this ordering is well-founded. The proof is by the induction on (u, l).

(Induction base) For (Λ, l) such that l is an arbitrary positive natural number, Π is of the form

$$\begin{array}{ll} & \Leftarrow E, \mathbf{f(s_1, \ldots, s_n) \doteq t}, E' \\ \rightarrow^{\sigma}_{[n]} & \Leftarrow \sigma(E, F, r = t, E') \\ \rightarrow\!\!\rightarrow^{\theta_1}_{LNC_0} & \square, \end{array} \tag{19}$$

where a new variant of $f(t_1, \ldots, t_n) = r \Leftarrow F$ of conditional equation in R is used, and σ is a substitution such that $\sigma s_i = \sigma t_i$ for $i = 1, \ldots, n$. The outermost narrowing rule [on] can be applied to $f(s_1, \ldots, s_n) = t$ with $f(t_1, \ldots, t_n) = r \Leftarrow F$, and we obtain a derivation w.r.t. LNC_0:

$$\begin{array}{ll} & \Leftarrow E, \mathbf{f(s_1, \ldots, s_n) \doteq t}, E' \\ \rightarrow_{[on]} & \Leftarrow E, \mathbf{s_1 = t_1, \ldots, s_n = t_n}, F, r = t, E' \\ \rightarrow\!\!\rightarrow^{\sigma}_{UC} & \Leftarrow \sigma(E, F, r = t, E') \quad \text{by Lemma 1} \\ \rightarrow\!\!\rightarrow^{\theta_1}_{LNC_0} & \square \quad \text{by the refutation starting from the goal in (19).} \end{array}$$

Let θ' be $\theta_1\sigma$, and θ' satisfies the required condition.

(induction step) For (u, l) such that $u \succ \Lambda$, We have to consider the following five cases according to the rules used in the first step of the refutation $\Pi_1 : \Leftarrow \sigma(E, F, s[u \leftarrow r] = t, E') \underbrace{\rightarrow\!\!\rightarrow^{\theta_1}_{LNC_0}}_{l\ (\geq 1)\ \text{steps}} \square$. By Lemma 3 we may assume that $\sigma(s[u \leftarrow r] = t)$ is selected in the first step of the refutation Π_1. For simplicity, we consider the case that the prefix of u is 1. Other cases are treated similarly. Let u be $1.u'$.

[1] (when the first step of Π_1 is [t]) In this case quasi-refutation Π is

$$\begin{array}{ll} & \Leftarrow E, \mathbf{s \doteq t}, E' \\ \rightarrow^{\sigma}_{[n]} & \Leftarrow \sigma(E, F, \mathbf{s[u \leftarrow r] = t}, E') \\ \rightarrow_{[t]} & \Leftarrow \sigma(E, F, E') \\ \rightarrow\!\!\rightarrow^{\theta_1}_{LNC_0} & \square. \end{array} \tag{20}$$

Since $\sigma(s[u \leftarrow r])$ and σt are identical and $u \succ \Lambda$, the leftmost symbol of s and t are the same. Let s be $f(s_1, \ldots, s_n)$ and t be $f(t_1, \ldots, t_n)$. Then, we have the following derivation

$$\Leftarrow E, \mathbf{f(s_1, \ldots, s_n) \doteq f(t_1, \ldots, t_n)}, E' \rightarrow_{[d]} \Leftarrow E, s_1 = t_1, \ldots, s_n = t_n, E'$$

and quasi-refutation

$$\begin{array}{ll} & \Leftarrow E, \mathbf{s_1 = t_1}, \ldots, s_n = t_n, E' \\ \rightarrow^{\sigma}_{[n]} & \Leftarrow \sigma(E, F, \mathbf{s_1[u' \leftarrow r] = t_1}, s_2 = t_2, \ldots, s_n = t_n, E') \\ \rightarrow_{[t]} & \Leftarrow \sigma(E, F, \mathbf{s_2 = t_2}, s_3 = t_3, \ldots, s_n = t_n, E') \\ \vdots & \\ \rightarrow_{[t]} & \Leftarrow \sigma(E, F, E') \\ \rightarrow\!\!\rightarrow^{\theta_1}_{LNC_0} & \square \qquad \text{by the refutation starting from the goal in (20).} \end{array} \tag{21}$$

Since $u' \prec u$, by the induction hypothesis the above quasi-refutation is replaced by the refutation $\Leftarrow E, \mathbf{s_1 = t_1}, \ldots, s_n = t_n, E' \rightarrow\!\!\rightarrow^{\theta'}_{LNC_0} \square$, where $\theta' \geq \theta_1\sigma$. Hence we obtain

$$\Leftarrow E, f(s_1, \ldots, s_n) = f(t_1, \ldots, t_n), E' \rightarrow\!\!\rightarrow^{\theta'}_{LNC_0} \square$$

such that $\theta' \geq \theta_1\sigma$.

[2] (when the first step of Π_1 is [v]) In this case, $\sigma(s[u \leftarrow r])$ or σt in Π_1 is a variable. Let $s \doteq t$ be $s \doteq x$, and σx be y, where $y \notin Var(\sigma(s[u \leftarrow r]))$. In this case Π is

$$\begin{array}{ll} & \Leftarrow E, \mathbf{s \doteq x}, E' \\ \rightarrow^{\sigma}_{[n]} & \Leftarrow \sigma(E, F, \mathbf{s[u \leftarrow r] = x}, E') \\ \rightarrow^{\rho}_{[v]} & \Leftarrow \rho\sigma(E, F, E') \\ \twoheadrightarrow^{\theta'_1}_{LNC_0} & \square, \end{array}$$

where $\rho = \{\sigma(s[u \leftarrow r])/y\}$ and θ'_1 is a substitution such that $\theta'_1\rho = \theta_1$. Let s be $f(s_1, \ldots, s_n)$. Then we have the following derivation

$$\Leftarrow E, \mathbf{f(s_1, \ldots, s_n) \doteq x}, E' \rightarrow^{\rho'}_{[im]} \Leftarrow \rho'(E, s_1 = x_1, \ldots, s_n = x_n, E'),$$

where $\rho' = \{f(x_1, \ldots, x_n)/x\}$ and $x_1, \ldots, x_n$ are distinct fresh variables, and quasi-derivation

$$\begin{array}{ll} & \Leftarrow \rho'(E, \mathbf{s_1 = x_1}, \ldots, s_n = x_n, E') \qquad (22) \\ \rightarrow^{\sigma}_{[n]} & \Leftarrow \sigma\rho'(E, F, \mathbf{s_1[u' \leftarrow r] = x_1}, s_2 = x_2, \ldots, s_n = x_n, E') \\ \rightarrow^{\rho_1}_{[v]} & \Leftarrow \rho_1\sigma\rho'(E, F, \mathbf{s_2 = x_2}, s_3 = x_3, \ldots, s_n = x_n, E') \\ \vdots & \\ \rightarrow^{\rho_n}_{[v]} & \Leftarrow \rho_n \ldots \rho_1\sigma\rho'(E, F, E') \qquad (23) \end{array}$$

where $\rho_1 = \{\sigma\rho'(s_1[u' \leftarrow r])/x_1\}$, and $\rho_i = \{\rho_{i-1} \ldots \rho_1\sigma\rho' s_i/x_i\}$ for $i = 2, \ldots, n$. Note that since $x_1, \ldots, x_n$ are distinct fresh variables, we have $\rho_n \ldots \rho_1 = \rho_n \cup \ldots \cup \rho_1$. Since

$$\begin{array}{lll} \rho_n \ldots \rho_1\sigma\rho' x & = \rho_n \ldots \rho_1\sigma f(x_1, \ldots, x_n) & \text{by the definition of } \rho' \\ & = \rho_n \ldots \rho_1 f(x_1, \ldots, x_n) & \text{since } x_1, \ldots, x_n \notin Dom(\sigma) \\ & = f(s_1[u' \leftarrow r], s_2, \ldots, s_n) & \text{since } \rho_n \ldots \rho_1 = \rho_n \cup \ldots \cup \rho_1 \\ & \geq \sigma(s[u \leftarrow r]) & \\ & = \rho y & \text{by the definition of } \rho \\ & = \rho\sigma x & \text{since } y = \sigma x, \end{array}$$

we obtain $\rho_n \ldots \rho_1\sigma\rho' x \geq \rho\sigma x$. For any variable $z(\neq x)$, $\rho_n \ldots \rho_1\sigma\rho' z = \sigma z \geq \rho\sigma z$. Therefore, we obtain $\rho_n \ldots \rho_1\sigma\rho' \geq \rho\sigma$. Let γ be a substitution such that $\gamma\rho_n \ldots \rho_1\sigma\rho' = \rho\sigma$. By Lemma 2, we obtain a refutation

$$\Leftarrow \rho_n \ldots \rho_1\sigma\rho'(E, F, E') \twoheadrightarrow^{\theta''_1}_{LNC_0} \square,$$

where $\theta''_1 \geq \theta'_1\gamma$. Combining this refutation with the quasi-derivation (22) we obtain a quasi-refutation:

$$\begin{array}{ll} & \Leftarrow \rho'(E, s_1 = x_1, \ldots, s_n = x_n, E') \\ \rightarrow^{\sigma}_{[n]} & \Leftarrow \sigma\rho'(E, F, s_1[u' \leftarrow r] = x_1, s_2 = x_2, \ldots, s_n = x_n, E') \\ \twoheadrightarrow^{\theta''_1\rho_n \ldots \rho_1}_{LNC_0} & \square. \qquad (24) \end{array}$$

Since $u' \prec u$, by the induction hypothesis, corresponding to the quasi-refutation (24), there exists a refutation $\Leftarrow E, \mathbf{s_1 = x_1}, \ldots, s_n = x_n, E \twoheadrightarrow^{\theta_2}_{LNC_0} \square$, where $\theta_2 \geq \theta''_1\rho_n \ldots \rho_1\sigma$. Therefore, we

have a refutation of $\Leftarrow E, f(s_1,\ldots,s_n) = x, E' \overset{\theta_2\rho'}{\twoheadrightarrow}_{LNC_0} \square$. We see that $\theta_2\rho' \geq \theta_1''\rho_n \ldots \rho_1\sigma\rho' \geq \theta_1'\gamma\rho_n \ldots \rho_1\sigma\rho' \geq \theta_1'\rho\sigma = \theta_1\sigma$.

[3] (when the first step of Π_1 is [on]) We have to consider the two cases; the cases that [on] is applied to the LHS of $\sigma(s[u \leftarrow r]) = \sigma t$ in Π', and that [on] is applied to the RHS.

We begin with the former case. Let s be $f(s_1,\ldots,s_n)$. In this case Π is

$$
\begin{array}{ll}
 & \Leftarrow E, \mathbf{f(s_1,\ldots,s_n) \doteq t}, E' \\
\overset{\sigma}{\rightarrow}_{[n]} & \Leftarrow \sigma(E, F, \mathbf{f(s_1[u' \leftarrow r], s_2,\ldots,s_n) = t}, E') \\
 & \text{where a new variant } l = r \Leftarrow F \text{ is used} \\
\rightarrow_{[on]} & \Leftarrow \sigma(E, F, s_1[u' \leftarrow r] = s_1', \ldots, s_n = s_n', F', r' = t, E') \\
 & \text{where a new variant } f(s_1',\ldots,s_n') = r' \Leftarrow F' \text{ is used.} \\
\overset{\theta_1}{\twoheadrightarrow}_{LNC_0} & \square,
\end{array}
\tag{25}
$$

Then we obtain the following derivation

$$\Leftarrow E, \mathbf{f(s_1,\ldots,s_n) \doteq t}, E' \rightarrow_{[on]} \Leftarrow E, s_1 = s_1', \ldots, s_n = s_n', F', r' = t, E'$$

and quasi-refutation

$$
\begin{array}{ll}
 & \Leftarrow E, \mathbf{s_1 = s_1'}, \ldots, s_n = s_n', F', r' = t, E' \\
\overset{\sigma}{\rightarrow}_{[n]} & \Leftarrow \sigma(E, F, s_1[u' \leftarrow r] = s_1', \ldots, s_n = s_n', F', r' = t, E') \\
\overset{\theta_1}{\twoheadrightarrow}_{LNC_0} & \square \text{ by the refutation starting from the goal in (25).}
\end{array}
$$

Since $u' \prec u$, by the induction hypothesis we obtain a refutation:

$$
\begin{array}{ll}
 & \Leftarrow E, \mathbf{f(s_1,\ldots,s_n) = t}, E' \\
\rightarrow_{[on]} & \Leftarrow E, \mathbf{s_1 = s_1'}, \ldots, s_n = s_n', F', r' = t, E' \\
\overset{\theta'}{\twoheadrightarrow}_{LNC_0} & \square
\end{array}
$$

such that $\theta' \geq \theta_1\sigma$.

In the latter case, let t be $g(t_1,\ldots,t_m)$. In this case Π is

$$
\begin{array}{ll}
 & \Leftarrow E, \mathbf{s \doteq g(t_1,\ldots,t_m)}, E' \\
\overset{\sigma}{\rightarrow}_{[n]} & \Leftarrow \sigma(E, F, \mathbf{s[u \leftarrow r] = g(t_1,\ldots,t_m)}, E') \\
 & \text{where a new variant } l = r \Leftarrow F \text{ is used} \\
\rightarrow_{[on]} & \Leftarrow \sigma(E, F, t_1 = t_1', \ldots, t_m = t_m', F'', r'' = s[u \leftarrow r], E') \\
 & \text{where a new variant } g(t_1',\ldots,t_m') = r'' \Leftarrow F'' \text{ is used} \\
\underbrace{\overset{\theta_1}{\twoheadrightarrow}_{LNC_0}}_{l-1 \text{ steps}} & \square.
\end{array}
\tag{26}
$$

Then we have the following derivation

$$\Leftarrow E, \mathbf{s \doteq g(t_1,\ldots,t_m)}, E' \rightarrow_{[on]} \Leftarrow E, t_1 = t_1', \ldots, t_m = t_m', F'', r'' = s, E'$$

and quasi-refutation

$$
\begin{array}{ll}
 & \Leftarrow E, t_1 = t_1', \ldots, t_m = t_m', F'', \mathbf{r'' = s}, E' \qquad (27) \\
\overset{\sigma}{\rightarrow}_{[n]} & \Leftarrow \sigma(E, t_1 = t_1', \ldots, t_m = t_m', F'', F, r'' = s[u \leftarrow r], E') \\
\underbrace{\overset{\theta_1}{\twoheadrightarrow}_{LNC_0}}_{l-1 \text{ steps}} & \square \text{ by the refutation starting from the goal in (26) and by lemma 3.} \qquad (28)
\end{array}
$$

The length of the quasi-refutation starting from the goal in (27) is shorter than the length of the quasi-refutation Π. Hence by the induction hypothesis, we obtain a refutation:

$$\begin{array}{ll} & \Leftarrow E, \mathbf{s} = \mathbf{g}(\mathbf{t_1}, \ldots, \mathbf{t_m}), E' \\ \rightarrow_{[\mathrm{on}]} & \Leftarrow E, t_1 = t'_1, \ldots, t_m = t'_m, F'', \mathbf{r}'' = \mathbf{s}, E' \\ \overset{\theta'}{\twoheadrightarrow}_{LNC_0} & \square \end{array}$$

such that $\theta' \geq \theta_1\sigma$.

[4] (when the first step of Π_1 is [im]) In this case $\sigma(s[u \leftarrow r])$ or σt in Π' is a variable. Let $s \doteq t$ be $f(s_1, \ldots, s_n) \doteq x$ and y be σx. In this case Π is

$$\begin{array}{ll} & \Leftarrow E, \mathbf{f}(\mathbf{s_1}, \ldots, \mathbf{s_n}) \doteq \mathbf{x}, E' \\ \overset{\sigma}{\rightarrow}_{[\mathrm{n}]} & \Leftarrow \sigma(E, F, f(s_1[u' \leftarrow r], \ldots, s_n) = x, E') \\ & \text{where a new variant } l = r \Leftarrow F \text{ is used} \\ \overset{\rho}{\rightarrow}_{[\mathrm{im}]} & \Leftarrow \rho\sigma(E, F, s_1[u' \leftarrow r] = x_1, \ldots, s_n = x_n, E') \\ \overset{\theta'_1}{\twoheadrightarrow}_{LNC_0} & \square, \end{array}$$

where $\rho = \{f(x_1, \ldots, x_n)/y\}$ ($x_1, \ldots, x_n$ are distinct fresh variables) and θ'_1 is a substitution such that $\theta'_1\rho = \theta_1$. Let $\rho' = \{f(x_1, \ldots, x_n)/x\}$. Note that $\rho'(s_1/u')$ and l is unifiable with an mgu σ since $\sigma(s_1/u') = \sigma l$ and σx is a variable (i.e., y). $\sigma\rho' x = \rho\sigma x$ for x, and $\sigma\rho' z = \sigma z \geq \rho\sigma z$ for any variable $z(\neq x)$. Hence $\sigma\rho' \geq \rho\sigma$. Let γ be a substitution such that $\gamma\sigma\rho' = \rho\sigma$. By Lemma 2 we obtain a refutation

$$\Leftarrow \sigma\rho'(E, F, s_1[u' \leftarrow r] = x_1, \ldots, s_n = x_n, E') \overset{\theta''_1}{\twoheadrightarrow}_{LNC_0} \square, \tag{29}$$

where $\theta''_1 \geq \theta'_1\gamma$. Therefore, we obtain the following derivation

$$\Leftarrow E, \mathbf{f}(\mathbf{s_1}, \ldots, \mathbf{s_n}) \doteq \mathbf{x}, E' \overset{\rho'}{\rightarrow}_{[\mathrm{im}]} \Leftarrow \rho'(E, s_1 = x_1, \ldots, s_n = x_n, E')$$

and quasi-refutation

$$\begin{array}{lll} & \Leftarrow \rho'(E, \mathbf{s_1} = \mathbf{x_1}, \ldots, s_n = x_n, E') & \\ \overset{\sigma}{\rightarrow}_{[\mathrm{n}]} & \Leftarrow \sigma\rho'(E, F, s_1[u' \leftarrow r] = x_1, \ldots, s_n = x_n, E') & \\ & \qquad \text{since } \sigma\rho'(s_1/u') = \sigma l & \\ \overset{\theta''_1}{\twoheadrightarrow}_{LNC_0} & \square \qquad \text{by (29).} & \end{array}$$

Since $u' \prec u$, by the induction hypothesis we have

$$\begin{array}{ll} & \Leftarrow E, \mathbf{f}(\mathbf{s_1}, \ldots, \mathbf{s_n}) = \mathbf{x}, E' \\ \overset{\rho'}{\rightarrow}_{[\mathrm{im}]} & \Leftarrow \rho'(E, \mathbf{s_1} = \mathbf{x_1}, \ldots, s_n = x_n, E') \\ \overset{\theta_2}{\twoheadrightarrow}_{LNC_0} & \square, \end{array}$$

where $\theta_2 \geq \theta''_1\sigma$. Let $\theta' = \theta_2\rho'$. Then we have $\theta' = \theta_2\rho' \geq \theta''_1\sigma\rho' \geq \theta'_1\gamma\sigma\rho' = \theta'_1\rho\sigma = \theta_1\sigma$.

The case of [d] is similarly treated to the case of [on], hence is omitted.

∎

Lemma 4. (completeness of LNC_0 w.r.t. NC) Let R be a conditional equational system and G be a goal. If there exists a refutation $G \overset{\theta}{\twoheadrightarrow}_{NC} \square$ then there exists a refutation $G \overset{\theta'}{\twoheadrightarrow}_{LNC_0} \square$ such that $\theta' \geq \theta$.

(Proof) The proof is by the induction on the length k of the refutation $\Pi : G \overset{\theta}{\twoheadrightarrow}_{NC} \square$. For $k = 0$, the result immediately holds. For $k > 0$, let G be $\Leftarrow E, s \doteq t, E'$. We have to consider the following two cases.

[1](the first step of the refutation Π is [f]) In this case Π is of the form:

$$\Leftarrow E, \mathbf{s} \doteq \mathbf{t}, E' \overset{\sigma}{\rightarrow}_{[f]} \Leftarrow \sigma(E, E') \overset{\theta_1}{\twoheadrightarrow}_{NC} \square$$

where σ is an mgu of s and t. By Lemma 1, we obtain a derivation

$$\Leftarrow E, \mathbf{s} \doteq \mathbf{t}, E' \overset{\sigma}{\twoheadrightarrow}_{UC} \Leftarrow \sigma(E, E')$$

The result follows immediately by the induction hypothesis.

[2](the first step of the refutation Π is [n]) In this case Π is of the form:

$$\Leftarrow E, \mathbf{s} \doteq \mathbf{t}, E' \overset{\sigma}{\rightarrow}_{[n]} \Leftarrow \sigma(E, F, s[u \leftarrow r] = t, E') \overset{\theta_1}{\twoheadrightarrow}_{NC} \square,$$

where a new variant $l = r \Leftarrow F$ of a conditional equation in R is used. By the induction hypothesis, we have

$$\Leftarrow E, \mathbf{s} \doteq \mathbf{t}, E' \overset{\sigma}{\rightarrow}_{[n]} \Leftarrow \sigma(E, F, s[u \leftarrow r] = t, E') \overset{\theta'_1}{\twoheadrightarrow}_{LNC_0} \square,$$

and $\theta'_1 \geq \theta_1$. By Lemma 11, we obtain a corresponding refutation

$$\Leftarrow E, \mathbf{s} \doteq \mathbf{t}, E' \overset{\theta'}{\twoheadrightarrow}_{LNC_0} \square,$$

such that $\theta' \geq \theta'_1\sigma$. $\theta' \geq \theta'_1\sigma \geq \theta_1\sigma = \theta$, and we are done.

∎