Completion after Program Inversion of Injective Functions

Naoki Nishida [1] Masahiko Sakai [2]

Graduate School of Information Science, Nagoya University
Nagoya, Japan

Abstract

Given a constructor term rewriting system that defines injective functions, the inversion compiler proposed by Nishida, Sakai and Sakabe generates a conditional term rewriting system that defines the inverse relations of the injective functions, and then the compiler unravels the conditional system into an unconditional term rewriting system. In general, the resulting unconditional system is not (innermost-)confluent even if the conditional system is (innermost-)confluent. In this paper, we propose a modification of the Knuth-Bendix completion procedure, which is used as a post-processor of the inversion compiler. Given a confluent and operationally terminating conditional system, the procedure takes the resulting unconditional systems as input. When the procedure halts successfully, it returns convergent systems that are computationally equivalent to the conditional systems. To adapt the modified procedure to the conditional systems that are not confluent but innermost-confluent, we propose a simplified variant of the modified procedure. We report that the implementations of the procedures succeed in generating innermost-convergent inverse systems for all the examples we tried.

Keywords: unraveling, conditional term rewriting system, convergence, innermost reduction

1 Introduction

Inverse computation of an n-ary function f is, given an output v, the calculation of the possible input $v_1, \ldots, v_n$ of f such that $f(v_1, \ldots, v_n) = v$. Two approaches for inverse computation are distinguished [1]: *inverse interpreters* [4,1] that performs inverse computation, and *inversion compilers* [18,28,9,25,24,7,19,20,2] that performs program inversion.

Given a constructor term rewriting system (constructor TRS), the inversion compiler proposed in [24,25] first generates a deterministic conditional TRS (DCTRS) as an intermediate result, and then transforms the DCTRS into a TRS that is equivalent to the DCTRS with respect to inverse computation. The first phase of the compiler performs a local *inversion*: for every constructor TRS, the first phase generates a DCTRS, called an *inverse system*, which represents the complete inverse relation for the reduction relation of the constructor TRS. The second phase

[1] Email: nishida@is.nagoya-u.ac.jp
[2] Email: sakai@is.nagoya-u.ac.jp

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

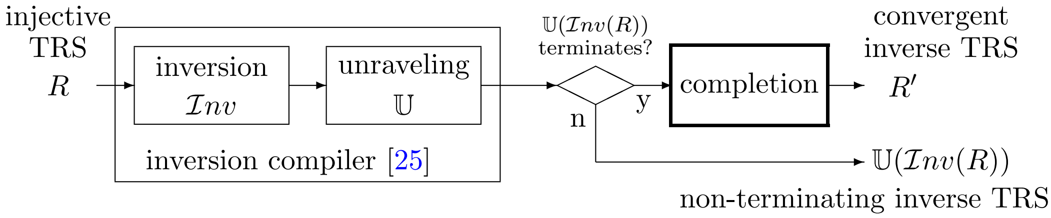

Fig. 1. Overview of the partial inversion with completion.

employs (a variant of) Ohlebusch's *unraveling* [26]. *Unravelings* are transformations based on Marchiori's approach [15] that transform DCTRSs into TRSs.

Unfortunately, the compiler cannot always generate TRSs that are *computationally equivalent* to the corresponding DCTRSs due to a characteristic of unravelings [15,27,30,22]. The characteristic is that the unraveled TRSs of DCTRSs may have unexpected normal forms that represent dead ends of wrong choices at branches of evaluating conditional parts of the DCTRSs (see the example $\mathcal{I}nv(R_1)$ shown later in this section). These wrong choices are captured by critical pairs of the unraveled TRSs, each of which originates from two (conditional) rewrite rules corresponding to the 'correct' and 'wrong' choices. Note that any rules looking like 'wrong choice' must be necessary elsewhere, and that it is decidable whether or not a normal form is expected: a normal form of the unraveled TRSs is an unexpected one if it contains an extra defined symbol introduced by the unraveling.

In program inversion by the inversion compiler [25,24], this problem arises even if all functions defined in the given constructor TRSs are injective. For this reason, the resulting TRSs do not define functions and thus the inversion compiler is less applicable to injective functions in practical functional programming languages — it is easy to translate functional programs into constructor TRSs, but difficult to translate the resulting TRSs of the compiler back into functional programs.

In this paper, we propose a modification of the *Knuth-Bendix completion procedure* in order to transform the unraveled TRSs of confluent and operationally terminating DCTRSs into convergent (and possibly non-overlapping) TRSs that are computationally equivalent to the DCTRSs. Unfortunately, the procedure does not always halt just as in the case of the ordinary completion procedure. However, if the procedure halts successfully and the resulting convergent TRSs are non-overlapping, then the resulting systems can be translated back into functional programs due to the non-overlapping property. When all functions defined in the input TRSs are injective, we take the modified completion procedure as a post-processor into the inversion compiler (Fig. 1 and Section 5). Through this approach, we show that unravelings are useful not only in analyzing properties of DCTRSs [15,27] but also in generating programs that can be used for computation instead of the corresponding original programs, such as program inversion of functional programs.

Consider the following functional program written in Standard ML where $\mathsf{Snoc}(xs, y)$ produces the list obtained from xs by adding y as the last element:

```
fun Snoc(   []   , y ) = [y]
  | Snoc( x::xs, y ) = x :: Snoc( xs, y );
```

We can easily translate the above program into the following constructor TRS:

$$R_1 = \{\, \mathsf{Snoc}(\mathsf{nil}, y) \to [y], \quad \mathsf{Snoc}(x :: xs, y) \to x :: \mathsf{Snoc}(xs, y) \,\}$$

where nil and $::$ are list constructors as usual, $[t_1, t_2, \ldots, t_n]$ abbreviates the list $t_1 :: (t_2 :: \cdots :: (t_n :: \mathsf{nil}) \cdots)$. The compiler inverts R_1 into the following DCTRS in the first phase: [3]

$$\mathcal{I}nv(R_1) = \begin{cases} \mathsf{InvSnoc}([y]) \to \langle \mathsf{nil}, y \rangle \\ \mathsf{InvSnoc}(x :: ys) \to \langle x :: xs, y \rangle \Leftarrow \mathsf{InvSnoc}(ys) \to \langle xs, y \rangle \end{cases}$$

where each tuple of n terms $t_1, \ldots, t_n$ is denoted by $\langle t_1, \ldots, t_n \rangle$ that can be represented as terms by introducing an n-ary constructor. The compiler unravels the DCTRS $\mathcal{I}nv(R_1)$ into the following TRS in the second phase:

$$\mathbb{U}(\mathcal{I}nv(R_1)) = \left\{ \begin{array}{c} \mathsf{InvSnoc}([y]) \to \langle \mathsf{nil}, y \rangle, \\ \mathsf{InvSnoc}(x :: ys) \to U_1(\mathsf{InvSnoc}(ys), x, ys), \\ U_1(\langle xs, y \rangle, x, ys) \to \langle x :: xs, y \rangle \end{array} \right\}$$

The introduced symbol U_1 is used for evaluating the conditional part $\mathsf{InvSnoc}(ys) \to \langle xs, y \rangle$ of the second rule in $\mathcal{I}nv(R_1)$. The term $\mathsf{Snoc}([\mathsf{a}, \mathsf{b}], \mathsf{c})$ has a unique normal form $[\mathsf{a}, \mathsf{b}, \mathsf{c}]$ but $\mathsf{InvSnoc}([\mathsf{a}, \mathsf{b}, \mathsf{c}])$ has two normal forms: a solution $\langle [\mathsf{a}, \mathsf{b}], \mathsf{c} \rangle$ of inverse computation and an unexpected normal form $U_1(U_1(U_1(\mathsf{InvSnoc}(\mathsf{nil}), \mathsf{c}, \mathsf{nil}), \mathsf{b}, [\mathsf{c}]), \mathsf{a}, [\mathsf{b}, \mathsf{c}])$. The restricted inversion compiler in [2] for generating non-overlapping systems is not applicable to this case because R_1 is out of its scope. In this example, it appears to be easy to translate from the TRS $\mathbb{U}(\mathcal{I}nv(R_1))$ or the CTRS $\mathcal{I}nv(R_1)$ into a functional program because we can easily determine an appropriate priority of rules, for instance, the common first rule $\mathsf{InvSnoc}[y] \to \langle \mathsf{nil}, y \rangle$ may have the highest priority. However, such a translation based on priorities of rules is difficult in general because we cannot decide which rules have priority of the application to terms. On the other hand, it is probably impossible that one transforms input systems into equivalent systems from which the compiler generates the inverse systems without overlapping.

To avoid this problem, it has been shown in [22] that the transformation in [30] is suitable as the second phase of the compiler, in the sense of producing convergent systems. However, the generated systems contain some special symbols and *overlapping* rules. For this reason, it is difficult to translate the convergent but *overlapping* TRS into a functional program (see Section 6).

Roughly speaking, non-confluence of $\mathbb{U}(\mathcal{I}nv(R_1))$ comes from the critical pair $(\langle \mathsf{nil}, x \rangle, U_1(\mathsf{InvSnoc}(\mathsf{nil}), x, \mathsf{nil}))$ between the first and second rules in $\mathbb{U}(\mathcal{I}nv(R_1))$. In this case, the application of the first rule is 'correct' and that of the second is 'wrong', that is, $\langle \mathsf{nil}, x \rangle$ is the expected result and $U_1(\mathsf{InvSnoc}(\mathsf{nil}), x, \mathsf{nil})$ is the unexpected recursive call of U_1 containing the dead end $\mathsf{InvSnoc}(\mathsf{nil})$. From this observation, by adding the rule $U_1(\mathsf{InvSnoc}(\mathsf{nil}), x, \mathsf{nil}) \to \langle \mathsf{nil}, x \rangle$, the unexpected normal form of $\mathsf{InvSnoc}([\mathsf{a}, \mathsf{b}, \mathsf{c}])$ can be reduced to the solution. This added rule provides a path from the wrong branch of inverse computation back to the correct branch. Due to this rule, the new TRS is confluent. This process just corresponds

[3] To simplify discussions, we omit describing special rules in the form of $\mathsf{Inv}F(F(x_1, \ldots, x_n)) \to \langle x_1, \ldots, x_n \rangle$ [25,24] because they are meaningless for inverse computation in dealing with functional programs on call-by-value interpretation. The special rules are necessary only for inverse computation of normalizing computation in term rewriting.

to the behavior of *completion*. Therefore, *completion* is expected to solve the non-confluence of TRSs obtained by the inversion compiler.

In Section 3, we propose a notion of *operationally innermost reduction* of DCTRSs that corresponds to *call-by-value interpretation* of functional programs, and we show that *simulation-completeness* with respect to innermost reduction is preserved by Ohlebusch's unraveling if the DCTRSs are restricted to functional programs having `let`-like structures.

In Section 4, we propose a modification of the Knuth-Bendix completion procedure, by adding a side condition to the *orientation* phase. Given a confluent and operationally terminating DCTRS, the modified completion procedure takes the unraveled TRSs as input. When the procedure halts successfully, it returns a convergent TRS that is computationally equivalent to the DCTRS. To obtain innermost-convergent TRSs from the unraveled TRSs of operationally terminating DCTRSs that are not confluent but innermost-confluent, we simplify the modified completion procedure by prohibiting the modified procedure to use two basic functions (*composition* and *simplification*), and by giving an additional side condition to the orientation phase. The additional condition restricts orientable equations to equations that are oriented without overlaps with other rewrite rules.

In Section 5, we first show a sufficient condition of constructor TRSs from which the inversion compiler generates (innermost-)convergent DCTRSs. Next, we describe an implementation of the modified completion procedure, and the experiments for the unraveled TRSs of DCTRSs obtained by the inversion compiler [24] from injective functions shown by Kawabe et al. [9]. Finally, we illustrate an informal translation of the *non-overlapping* TRSs obtained by the procedure back into functional programs.

In this paper, we do not consider *sorts*. However, the framework in this paper can be extended to many-sorted systems as usual. All proofs can be found in the full version of this paper [21].

2 Preliminaries

Here, we will review the following basic notations of term rewriting [3,27].

Throughout this paper, we use $\mathcal{V}$ as a countably infinite set of *variables*. The set of all *terms* over a *signature* $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F},\mathcal{V})$. The set of all variables appearing in the terms $t_1,\dots,t_n$ is represented by $\mathcal{V}ar(t_1,\dots,t_n)$. The *identity* of terms s and t is denoted by $s \equiv t$. For a term t and a position p of t, the notation $t|_p$ represents the subterm of t at p. The function symbol at the *root position* ε of t is denoted by $\mathtt{root}(t)$. The notation $C[t_1,\dots,t_n]_{p_1,\dots,p_n}$ represents the term obtained by replacing each $\Box$ at position p_i of an n-hole *context* $C[\,]$ with term t_i for $1 \le i \le n$. The *domain* and *range* of a *substitution* σ are denoted by $\mathcal{D}om(\sigma)$ and $\mathcal{R}an(\sigma)$, respectively, and the application $\sigma(t)$ of σ to t is abbreviated to $t\sigma$. The *composition* $\sigma\theta$ of substitutions σ and θ is defined as $\sigma\theta(x) = \theta(\sigma(x))$. Given terms s and t, we write $s \sqsupseteq t$ if there are some $C[\,]$ and θ such that $s \equiv C[t\theta]$.

An *(oriented) conditional rewrite rule* over $\mathcal{F}$ is a triple (l,r,c), denoted by $l \to r \Leftarrow c$, such that l is a non-variable term in $\mathcal{T}(\mathcal{F},\mathcal{V})$, r is a term in $\mathcal{T}(\mathcal{F},\mathcal{V})$, and c is of the form of $s_1 \to t_1 \wedge \cdots \wedge s_n \to t_n$ ($n \ge 0$) with terms s_i and t_i

in $\mathcal{T}(\mathcal{F},\mathcal{V})$. In particular, the conditional rewrite rule $l \to r \Leftarrow c$ is said to be an *(unconditional) rewrite rule* if $n = 0$, and we may abbreviate it to $l \to r$. We sometimes attach a unique label ρ to a rule $l \to r \Leftarrow c$ by denoting $\rho : l \to r \Leftarrow c$, and we use the label to refer to the rule. An *(oriented) conditional rewriting system* (CTRS, for short) R over a signature $\mathcal{F}$ is a finite set of conditional rewrite rules over $\mathcal{F}$. Note that R is a *TRS* if all rules in R are unconditional. The *rewrite relation* of R is denoted by $\to_R$. To specify the applied position p and rule ρ, we write $\to_R^p$ or $\to_R^{[p,\rho]}$. We write $\to_R^{\varepsilon<}$ if p is not the root position ε. A conditional rewrite rule $\rho : l \to r \Leftarrow s_1 \to t_1 \cdots s_k \to t_k$ is called *deterministic* if $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l, t_1, \ldots, t_k)$ and $\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(l, t_1, \ldots, t_{i-1})$ for $1 \leq i \leq k$. The CTRS R is called a *deterministic CTRS* (DCTRS, for short) if all rules in R are deterministic. A notion of *operational termination* of DCTRSs is defined via the absence of infinite well-formed proof trees in some inference system [14]: a CTRS R is *operationally terminating* (OP-SN, for short) if for any terms s and t, any proof tree attempting to prove that $s \xrightarrow{*}_R t$ cannot be infinite.

Let $\to$ be a reduction over terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$. Then, the set of normal forms with respect to $\to$ is denoted by $NF_{\to}(\mathcal{F},\mathcal{V})$. The binary relation $\xrightarrow{*}^{!}$ is defined as $\{ (s,t) \mid s \xrightarrow{*} t, t \in NF_{\to}(\mathcal{F},\mathcal{V}) \}$.

Let R be a CTRS over $\mathcal{F}$. The sets $\mathcal{D}_R$ and $\mathcal{C}_R$ of all *defined symbols* and all *constructors* of R are defined as $\mathcal{D}_R = \{\mathtt{root}(l) \mid l \to r \Leftarrow c \in R\}$ and $\mathcal{C}_R = \mathcal{F} \setminus \mathcal{D}_R$, respectively. Terms in $\mathcal{T}(\mathcal{C}_R,\mathcal{V})$ are called *constructor terms* of R. The CTRS R is called a *constructor system* if every rule $f(t_1, \ldots, t_n) \to r \Leftarrow c$ in R satisfies $\{t_1, \ldots, t_n\} \subseteq \mathcal{T}(\mathcal{C}_R,\mathcal{V})$.

We use the notion of *context-sensitive reduction* in [13]. A *replacement mapping* μ is a mapping from a signature $\mathcal{F}$ to a set of natural numbers such that $\mu(f) \subseteq \{1, \ldots, n\}$ for n-ary symbols f *in* $\mathcal{F}$. When $\mu(f)$ is not defined explicitly, we assume that $\mu(f) = \{1, \ldots, n\}$. The set $\mathcal{O}_\mu(t)$ of *reducible positions* in t is defined as follows: $\mathcal{O}_\mu(x) = \emptyset$ where $x \in \mathcal{V}$, and $\mathcal{O}_\mu(f(t_1, \ldots, t_n)) = \{ip \mid i \in \mu(f), p \in \bigcup_{j \in \mu(f)} \mathcal{O}_\mu(t_j)\}$. The *context-sensitive reduction* of the *context-sensitive TRS* (R, μ) of a TRS R and a replacement map μ is denoted by $\to_{(R,\mu)}$: $\to_{(R,\mu)} = \{(s,t) \mid s \to_R^p t, p \in \mathcal{O}_\mu(s)\}$. The innermost reduction of $\to_{(R,\mu)}$ is denoted by $\underset{\mathrm{i}}{\to}_{(R,\mu)}$: $\underset{\mathrm{i}}{\to}_{(R,\mu)} = \{(s,t) \mid s \to_R^p t, p \in \mathcal{O}_\mu(s), (\forall q > p.\ q \in \mathcal{O}_\mu(s)$ implies that $s|_q$ is irreducible$)\}$.

Let $l_i \to r_i$ $(i = 1, 2)$ be two rules whose variables have been renamed such that $\mathcal{V}ar(l_1, r_1) \cap \mathcal{V}ar(l_2, r_2) = \emptyset$. Let p be a position in l_1 such that $l_1|_p$ is not a variable and let θ be a most general unifier of $l_1|_p$ and l_2. This determines a *critical pair* $(r_1\theta, (l_1\theta)[r_2\theta]_p)$. If the two rules are renamed versions of the same rewrite rule, we do not consider the case $p = \varepsilon$. If $p = \varepsilon$, then the critical pair is called an *overlay*. If two rules give rise to a critical pair, we say that they *overlap*. We denote the set of critical pairs constructed by rules in a TRS R by $CP(R)$. We also denote the set of critical pairs between rules in R and another TRS R' by $CP(R, R')$. Moreover, $CP_\varepsilon(R)$ denotes the set of *overlays* of R.

Let R and R' be CTRSs such that their normal forms are computable, and T be a set of terms. Roughly speaking, R' is *computationally equivalent* to R with respect to T if there exist mappings ϕ and ψ such that if R terminates on a term $s \in T$ admitting a unique normal form t, then R' also terminates on $\phi(s)$ and for any of its normal forms t', we have $\psi(t') = t$ [30]. In this paper, we assume that ϕ

and ψ are the identity mappings.

Let $\xrightarrow[1]{}$ and $\xrightarrow[2]{}$ be two binary relations on terms, and T' and T'' be sets of terms. We say that $\xrightarrow[1]{} = \xrightarrow[2]{}$ *in* $T' \times T''$ ($\xrightarrow[1]{} \supseteq \xrightarrow[2]{}$ *in* $T' \times T''$, respectively) if $\xrightarrow[1]{} \cap (T' \times T'') = \xrightarrow[2]{} \cap (T' \times T'')$ ($\xrightarrow[1]{} \cap (T' \times T'') \supseteq \xrightarrow[2]{} \cap (T' \times T'')$, respectively). Especially, we say that $\xrightarrow[1]{} = \xrightarrow[2]{}$ *in* T' (and $\xrightarrow[1]{} \supseteq \xrightarrow[2]{}$ *in* T') if $T' = T''$.

An *equation* over a signature $\mathcal{F}$ is a pair (s,t), denoted by $s \approx t$, such that s and t are terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$. We write $s \simeq t$ for representing $s \approx t$ or $t \approx s$. The *equational relation* with respect to a set E of equations is defined as $\leftrightarrow_E = \{\, (C[s\sigma], C[t\sigma]) \mid s \simeq t \in E \,\}$.

Finally, we introduce the Knuth-Bendix completion procedure [11,3,31].

Definition 2.1 Let E be a finite set of equations over a signature $\mathcal{F}$, and $\succ$ be a reduction order. Let $E_{(0)} = E$, $R_{(0)} = \emptyset$ and $i = 0$, we apply the following steps:

1. (Orientation) select $s \simeq t \in E_{(i)}$ such that $s \succ t$;
2. (Composition) $R' := \{ l \to r' \mid l \to r \in R_{(i)}, r \xrightarrow[\mathrm{i}]{*}\!{}^{!}_{R_{(i)} \cup \{s \to t\}} r' \}$;
3. (Deduction) $E' := (E_{(i)} \setminus \{s \simeq t\}) \cup \mathit{CP}(\{s \to t\}, R' \cup \{s \to t\})$;
4. (Collapse) $R_{(i+1)} := \{s \to t\} \cup \{l \to r \mid l \to r \in R', l \not\trianglerighteq s\}$;
5. (Simplification & Deletion)
   $E_{(i+1)} := \{s'' \approx t'' \mid s' \approx t' \in E', s' \xrightarrow[\mathrm{i}]{*}\!{}^{!}_{R_{(i+1)}} s'' \not\equiv t'' \xleftarrow[\mathrm{i}]{*}\!{}^{!}_{R_{(i+1)}} t' \}$;
6. if $E_{(i+1)} \neq \emptyset$ then $i := i + 1$ and go to step 1, otherwise output $R_{(i+1)}$.

Note that the procedure does not always halt. Suppose that the procedure halts successfully at $i+1 = k$ (hence $E_{(k)} = \emptyset$). Then, $R_{(k)}$ is convergent, and $R_{(k)}$ satisfies $\xleftrightarrow{*}_E = \xleftrightarrow{*}_{R_{(k)}}$ [3]. Note that when there is no rule to select at the Orientation step, the procedure halts in failure. Note that Composition and Collapse are used for efficiency, and the resulting systems are convergent even if Composition and Collapse are skipped.

3 Unraveled TRSs with Call-by-Value Interpretation

In this section, we propose a notion of *operationally innermost reduction* of DCTRSs that corresponds to *call-by-value interpretation* of functional programs, and we show that *simulation-completeness* with respect to innermost reduction is preserved by Ohlebusch's unraveling if the DCTRSs are restricted to functional programs having `let`-like structures.

We first give the definition of Ohlebusch's unraveling [26]. Given a finite set X of variables, we denote by $\overrightarrow{X}$ the sequence of variables in X without repetitions (in some fixed order).

Definition 3.1 Let R be a DCTRS over a signature $\mathcal{F}$. For every conditional rewrite rule $\rho : l \to r \Leftarrow s_1 \to t_1 \wedge \cdots \wedge s_k \to t_k$, let $|\rho|$ denote the number k of conditions in ρ. For every conditional rule $\rho \in R$, we prepare k 'fresh' function symbols $U^\rho_1, \ldots, U^\rho_k$ not in $\mathcal{F}$, called *U symbols*, in the transformation. We transform ρ into a set $\mathbb{U}(\rho)$ of $k+1$ unconditional rewrite rules as follows:

$$\mathbb{U}(\rho) = \left\{ \, l \to U^\rho_1(s_1, \overrightarrow{X_1}),\ U^\rho_1(t_1, \overrightarrow{X_1}) \to U^\rho_2(s_2, \overrightarrow{X_2}),\ \cdots,\ U^\rho_k(t_k, \overrightarrow{X_k}) \to r \, \right\}$$

where $X_i = \mathcal{V}ar(l, t_1, \ldots, t_{i-1})$. The system $\mathbb{U}(R) = \bigcup_{\rho \in R} \mathbb{U}(\rho)$ is a TRS over the extended signature $\mathcal{F}_{\mathbb{U}} = \mathcal{F} \cup \mathcal{D}_{\mathbb{U}}$ where $\mathcal{D}_{\mathbb{U}} = \{U^\rho_i \mid \rho \in R, 1 \leq i \leq |\rho|\}$.

Note that the definition of $\mathbb{U}$ is essentially equivalent to that in [26,29].

An unraveling U is *simulation-sound* (*simulation-preserving* and *simulation-complete*, respectively) for a DCTRS R over $\mathcal{F}$ if $\xrightarrow{*}_R \subseteq \xrightarrow{*}_{\mathbb{U}(R)}$ in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ ($\xrightarrow{*}_R \supseteq \xrightarrow{*}_{\mathbb{U}(R)}$ and $\xrightarrow{*}_R = \xrightarrow{*}_{\mathbb{U}(R)}$ in $\mathcal{T}(\mathcal{F}, \mathcal{V})$, respectively). Note that the simulation-preserving property is sometimes called simulation-completeness in some papers, and it is a necessary condition of being unravelings. Roughly speaking, the computational equivalence is equivalent to the combination of simulation-completeness and normal-form uniqueness. The unraveling $\mathbb{U}$ is not simulation-sound for every DCTRS [27]. To avoid this difficulty of non-'simulation-soundness' of $\mathbb{U}$, a restriction to the rewrite relations of the unraveled TRSs is shown in [29], which is done by the *context-sensitive* condition given by the replacement map μ such that $\mu(U^\rho_i) = \{1\}$ for every U^ρ_i in Definition 3.1. We denote the context-sensitive TRS $(\mathbb{U}(R), \mu)$ by $\mathbb{U}_{\mathrm{cs}}(R)$. We consider $\mathbb{U}_{\mathrm{cs}}$ as an unraveling from DCTRSs to context-sensitive TRSs.

Theorem 3.2 ([29]) *For every DCTRS R over $\mathcal{F}$, $\mathbb{U}_{\mathrm{cs}}$ is simulation-complete, that is, $\xrightarrow{*}_R = \xrightarrow{*}_{\mathbb{U}_{\mathrm{cs}}(R)}$ in $\mathcal{T}(\mathcal{F}, \mathcal{V})$.*

To apply completion procedures to unraveled TRSs, we expect that the unraveling $\mathbb{U}$ is simulation-complete without the context-sensitivity. To this end, we propose an 'innermost-like' reduction of DCTRSs, called *operationally innermost reduction*. Let R be an operationally terminating (OP-SN) DCTRS. The *n-level operationally innermost reduction* $\xrightarrow[(n),\mathrm{i}]{}_R$ is defined as follows:

- $\xrightarrow[(0),\mathrm{i}]{}_R = \emptyset$, and
- $\xrightarrow[(n+1),\mathrm{i}]{}_R = \xrightarrow[(n),\mathrm{i}]{}_R \cup \{\, (C[l\sigma], C[r\sigma]) \mid l \to r \Leftarrow s_1 \to t_1 \wedge \cdots s_k \to t_k \in R, \forall u \lhd l\sigma.\ u \in NF_{\to_R}(\mathcal{F}, \mathcal{V}), \forall i.\ s_i\sigma \xrightarrow[(n),\mathrm{i}]{*}{}^{!}_R t_i\sigma \,\}$.

The *operationally innermost reduction* $\xrightarrow[\mathrm{i}]{}_R$ of R is defined as $\bigcup_{i \geq 0} \xrightarrow[(i),\mathrm{i}]{}_R$. Note that if R is a TRS then $\xrightarrow[\mathrm{i}]{}_R$ is equivalent to the ordinary innermost reduction. Note that the ordinary definition of innermost reduction is not well-defined for every CTRS [8]. However, both the ordinary and operationally innermost reductions of OP-SN CTRSs are well-defined. R is called *innermost-confluent* (*innermost-convergent*) if $\xrightarrow[\mathrm{i}]{}_R$ is confluent.

Let R be a DCTRS. Terms in $\{u_1, \ldots, u_n, t_1, \ldots, t_k \mid f(u_1, \ldots, u_n) \to r \Leftarrow s_1 \to t_1 \wedge \cdots \wedge s_k \to t_k \in R\} \setminus \mathcal{V}$ are called *patterns* (in R). We denote the set of patterns in R by $\mathsf{Pat}(R)$. It follows from the definition of $\mathbb{U}$ that $\mathsf{Pat}(R) = \mathsf{Pat}(\mathbb{U}(R))$ up to variable renaming. Patterns represents structures of data by means of matching. Especially, in innermost reductions, patterns matches normal forms only.

Unfortunately, $\mathbb{U}_{\mathrm{cs}}$ is not simulation-preserving for every DCTRS with respect to the normalizing innermost reduction $\xrightarrow[\mathrm{i}]{*}{}^{!}$. This is because not all normal form of R are normal form of $\mathbb{U}_{\mathrm{cs}}(R)$, that is, $NF_{\to_R}(\mathcal{F}, \mathcal{V}) \not\subseteq NF_{\to_{\mathbb{U}_{\mathrm{cs}}(R)}}(\mathcal{F}, \mathcal{V})$. To preserve the simulation-preserving property, $\mathbb{U}_{\mathrm{cs}}(R)$ must have the same pattern-matching capability with R, that is, if an instance $p\theta$ of a pattern p is irreducible by R then $p\theta'$ is also irreducible by $\mathbb{U}_{\rm cs}(R)$ for every substitution θ' such that $x\theta \xrightarrow[\rm i]{*}\!{}^!_{\mathbb{U}_{\rm cs}(R)} x\theta'$ for all $x \in \mathcal{D}om(\theta)$. When all patterns are constructor terms (that is, $\mathbb{U}(R)$ is a constructor system), R and $\mathbb{U}(R)$ have the same pattern-matching capability. However, in examples of program inversion, a primitive operator `du` that requires *equality check* is used: $\mathtt{du}(\langle x\rangle) = \langle x, x\rangle$, $\mathtt{du}(\langle x, x\rangle) = \langle x\rangle$, and $\mathtt{du}(\langle x, y\rangle) = \langle x, y\rangle$ if $x \neq y$. This operator is encoded as the following terminating TRS:

$$R_{\mathsf{du}} = \left\{ \begin{array}{ccc} \mathsf{Du}(\langle x\rangle) \to \langle x, x\rangle, & \mathsf{Du}(\langle x, y\rangle) \to \mathsf{EqChk}(\mathsf{EQ}(x, y)), & \\ \mathsf{EqChk}(\langle x\rangle) \to \langle x\rangle, & \mathsf{EqChk}(\mathsf{EQ}(x, y)) \to \langle x, y\rangle, & \mathsf{EQ}(x, x) \to \langle x\rangle \end{array} \right\}$$

Note that any system containing Du is not a constructor system. Since R_{du} has no *overlay*, R_{du} is *locally innermost-confluent*, and hence, R_{du} is *innermost-confluent* [12]. Under the innermost reduction, R_{du} can simulate computation of `du`.

One of the sufficient conditions to have the same pattern-matching capability is to satisfy all of the following conditions:

- all rules defining $g \in \{g \in \mathcal{D}_R \mid g \text{ appears in } \mathsf{Pat}(R)\}$ are unconditional and every proper subterm of the left-hand sides is a variable, and
- every rule $l \to r \Leftarrow s_1 \to t_1 \wedge \cdots \wedge s_k \to t_k$ represents a `let`-*like structure*, that is, $\mathcal{V}ar(t_i) \cap \mathcal{V}ar(l, t_1, \ldots, t_{i-1}) = \emptyset$ for $1 \leq i \leq k$.

We call R *pattern-stable* if R satisfies all of these conditions. The `let`-like structure guarantees that $\mathcal{V}ar(t_i) \cap \{x_1, \ldots, x_n\} = \emptyset$ for every $U_i^\rho(t_i, x_1, \ldots, x_n)$ [23,25]. Pattern-stability is essential for DCTRSs that are used for modeling functional programs with `let`-like structures and equality check.

Theorem 3.3 *Let R be a pattern-stable OP-SN DCTRS over a signature $\mathcal{F}$, and s and t be terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$. Then, $s \xrightarrow[\rm i]{*}\!{}^!_R t$ implies $s \xrightarrow[\rm i]{*}\!{}^!_{\mathbb{U}_{\rm cs}(R)} u$ for some u in $\mathcal{T}(\mathcal{F}_{\mathbb{U}}, \mathcal{V})$ such that $t \xrightarrow[\rm i]{*}\!{}^!_{\mathbb{U}_{\rm cs}(R)} u$.*

Pattern-stability is also a sufficient condition for simulation-soundness. On the other hand, the non-erasing property of R is another sufficient condition. Here, we call R *strongly non-erasing* if every rule $l \to r \Leftarrow s_1 \to t_1 \wedge \cdots \wedge s_k \to t_k$ satisfies all of the following conditions [23,25]:

- $\mathcal{V}ar(l) \subseteq \mathcal{V}ar(r, s_1, t_1, \ldots, s_k, t_k)$, and
- $\mathcal{V}ar(t_i) \subseteq \mathcal{V}ar(r, s_{i+1}, t_{i+1}, \ldots, s_k, t_k)$ for $1 \leq i \leq k$.

Any U symbol is not consumed by pattern-matching. The non-erasing property guarantees that no normal form containing U symbols appears along the reduction $s \xrightarrow[\rm i]{*}\!{}^!_{\mathbb{U}_{\rm cs}(R)} t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$; if a normal form containing a U symbol appears in the sequence, the non-erasing property ensures that it remains in t.

Theorem 3.4 *Let R be a pattern-stable or strongly non-erasing OP-SN DCTRS over a signature $\mathcal{F}$, and s and t be terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$. Then, $s \xrightarrow[\rm i]{*}\!{}^!_{\mathbb{U}_{\rm cs}(R)} t$ implies $s \xrightarrow[\rm i]{*}\!{}^!_R t$.*

Context-sensitivity is not necessary for innermost reduction of $\mathbb{U}_{\rm cs}(R)$.

Theorem 3.5 *For every DCTRS R over $\mathcal{F}$, $\xrightarrow[\rm i]{*}_{\mathbb{U}(R)} = \xrightarrow[\rm i]{*}_{\mathbb{U}_{\rm cs}(R)}$ in $\mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathcal{T}(\mathcal{F}_{\mathbb{U}}, \mathcal{V})$.*

Thanks to Theorem 3.5, when evaluating terms by the innermost reduction of $\mathbb{U}_{\mathrm{cs}}(R)$, we can treat $\mathbb{U}(R)$ without the context-sensitivity determined by $\mathbb{U}$.

For pattern-stable OP-SN DCTRSs, we have the following simulation-soundness and weak simulation-preserving property.

Corollary 3.6 *Let R be a pattern-stable OP-SN DCTRS over a signature $\mathcal{F}$, and s and t be terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$. Then,*

(i) *$s \xrightarrow[\mathrm{i}]{*}{}^{!}_{R} t$ implies $s \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}(R)} u$ for some u in $\mathcal{T}(\mathcal{F}_{\mathbb{U}},\mathcal{V})$ such that $t \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}(R)} u$, and*

(ii) *$s \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}(R)} t$ implies $s \xrightarrow[\mathrm{i}]{*}{}^{!}_{R} t$.*

Corollary 3.6 does not mean that $s \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}(R)} u$ implies $s \xrightarrow[\mathrm{i}]{*}_{R} t$ for some t such that $t \xrightarrow[\mathrm{i}]{*}_{\mathbb{U}(R)} u$ and t is a normal form of R. This weakness of the simulation-preserving property does not happen when $\mathbb{U}(R)$ is innermost-confluent. Therefore, getting innermost-confluence is important for unraveled TRSs.

4 Completion of Unraveled TRSs

In this section, by adding a side condition to Orientation, we propose a modification of the ordinary Knuth-Bendix completion procedure for the unraveled TRSs of convergent DCTRSs. The modified procedure transforms the unraveled TRSs into convergent TRSs that are computationally equivalent to the DCTRSs. Moreover, to adapt the modified procedure to DCTRSs that are not confluent but innermost-confluent, we add another side condition to Orientation.

The usual purpose of completion procedures is to generate convergent TRSs that are equivalent to given equation sets. In contrast to the usual purpose, we expect completion procedures to transform unraveled TRSs $\mathbb{U}(R)$ into convergent TRSs that are computationally equivalent to the original DCTRSs R. To this end, we start the completion procedure from the initial pair $(CP(\mathbb{U}(R)), \{\, l \to r \in \mathbb{U}(R) \mid \nexists l' \to r' \in \mathbb{U}(R), l \sqsupset l' \,\})$ where $\mathbb{U}(R) \subseteq \succ$. Moreover, consistency of the normal forms of $\mathbb{U}(R)$ (that is, they are also normal forms of the resulting system) is necessary for preserving computational equivalence of R. For this requirement, we add the side condition '$\mathtt{root}(s)$ is a U symbol' to Orientation:

1. (Orientation†) select $s \approx t \in E_{(i)}$ such that $s \succ t$ and $\mathtt{root}(s)$ is a U symbol;

Due to the side condition of Orientation†, and due to the basic characteristic of the ordinary completion procedure [3], the *modified* completion procedure produces convergent TRSs that are computationally equivalent to the input TRSs when it halts successfully.

Theorem 4.1 *Let R be an OP-SN DCTRS over $\mathcal{F}$, and $\succ$ be a reduction order such that $\mathbb{U}(R) \subseteq \succ$. Let $E_0 = CP(\mathbb{U}(R))$, $R_0 = \{l \to r \in \mathbb{U}(R) \mid \nexists l' \to r \in \mathbb{U}(R), l \sqsupset l'\}$, and R' be a TRS obtained by the modified completion procedure from (E_0, R_0) with $\succ$. Then, (1) R' is convergent, (2) $NF_{\to_{\mathbb{U}(R)}}(\mathcal{F},\mathcal{V}) = NF_{\to_{R'}}(\mathcal{F},\mathcal{V})$, and (3) $\xrightarrow{*}{}^{!}_{\mathbb{U}(R)} = \xrightarrow{*}{}^{!}_{R'}$ in $\mathcal{T}(\mathcal{F},\mathcal{V})$.*

Since $NF_{\to_S}(\mathcal{F},\mathcal{V}) = NF_{\xrightarrow[\mathrm{i}]{}_S}(\mathcal{F},\mathcal{V})$ (S is either $\mathbb{U}(R)$ or R), it holds in Theorem 4.1 that $\xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}(R)} = \xrightarrow[\mathrm{i}]{*}{}^{!}_{R'}$ in $\mathcal{T}(\mathcal{F},\mathcal{V})$.

Example 4.2 Consider the non-convergent TRS $\mathbb{U}(\mathcal{I}nv(R_1))$ in Section 1 again. Given the *lexicographic path order* (LPO) $\succ_{\mathsf{lpo}}$ determined by the precedence $>$ with $\mathsf{InvSnoc} > U_1 > \,::\, > \mathsf{nil} > \langle\ \rangle$, we obtain the following convergent and non-overlapping TRS by the modified completion procedure (in 4 cycles):

$$R_2 = \left\{ \begin{array}{ll} \mathsf{InvSnoc}(x :: ys) \to U_1(\mathsf{InvSnoc}(ys), x, ys), & \\ U_1(\langle xs, y\rangle, x, ys) \to \langle x :: xs, y\rangle, & U_1(\mathsf{InvSnoc}(\mathsf{nil}), x, \mathsf{nil}) \to \langle \mathsf{nil}, x\rangle \end{array} \right\}$$

Since the procedure removes the rule $\mathsf{InvSnoc}([y]) \to \langle \mathsf{nil}, y\rangle$ from $\mathbb{U}(\mathcal{I}nv(R_1))$, the resulting TRS R_2 is non-overlapping.

Unfortunately, the modified completion procedure does not always halt even if the inputs are restricted to unraveled TRSs. For example, the modified procedure does not halt for the unraveled TRS obtained from Example 7.1.5 in [27] although there exists an appropriate convergent TRS that is computationally equivalent to the corresponding DCTRS.

Confluence of R is necessary for the modified completion procedure to halt 'successfully'. Note that confluence of R is not sufficient for the procedure to 'halt'. In other words, the procedure halts (or keeps running) 'unsuccessfully' if R is not confluent. If R is not confluent, then we have $t_1 \mathrel{\overset{*}{\leftarrow}_{\mathbb{U}(R)}} s \mathrel{\overset{*}{\to}_{\mathbb{U}(R)}} t_2$ and $t_1 \not\equiv t_2$ for some s, t_1 and t_2 in $\mathcal{T}(\mathcal{F}, \mathcal{V})$. The added side condition '$\mathtt{root}(s)$ is a U symbol' prevents t_1 and t_2 from being joinable. From this observation, the modified procedure can be considered as a method to show confluence of R: if the procedure succeeds, then R is confluent.

As stated above, we would like to transform DCTRSs on call-by-value interpretation into convergent TRSs that are computationally equivalent to the DCTRSs. Moreover, the modified completion procedure always fails for DCTRSs that are not confluent but innermost-confluent, such as DCTRSs containing R_{du}.

To obtain innermost-convergent systems that are computationally equivalent to TRSs containing R_{du}, applying completion procedures to the TRSs appears to be effective just as in the case of convergent TRSs. However, there is a difficulty associated with innermost reduction. The difficulty is that innermost reduction is not closed under substitutions. When applying the completion procedure to R_{du}, the rules $\mathsf{Du}(\langle x, y\rangle) \to \mathsf{EqChk}(\mathsf{EQ}(x, y))$ is transformed into $\mathsf{Du}(\langle x, y\rangle) \to \langle x, y\rangle$. Given a ground normal form t, the resulting system cannot simulate the reduction $\mathsf{Du}(t, t) \mathrel{\overset{*}{\underset{\mathrm{i}}{\to}}_{R_{\mathsf{du}}}} \langle t\rangle$ due to the lack of $\mathsf{Du}(\langle x, y\rangle) \to \mathsf{EqChk}(\mathsf{EQ}(x, y))$. To remove this troublesome problem from the modified completion procedure for innermost reduction, we prohibit the procedure to use the two operations Composition and Simplification, and give an additional side condition to Orientation† as follows:

1. (Orientation‡) select $s \approx t \in E_{(i)}$ such that $s \succ t$, $\mathtt{root}(s)$ is a U symbol, and $CP(\{s \to t\}, R_{(i)} \cup \{s \to t\}) = \emptyset$;

The additional condition means that the oriented rule $s \to t$ is not overlapping with other rules in $R_{(i)} \cup \{s \to t\}$. Thus, Deduction does not add any equations to the equation set $E_{(i)}$ but removes an equation. Since no U symbol appears in the left-hand side l in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ from the definition of $\mathbb{U}$, and since the added rules are not overlapping with other rules, Collapse removes no rules from the rule set $R_{(i)}$. If

E has no equation of the form $s \approx s$, Deletion step removes no equations from the equation set. From this observation, the modified procedure with Orientation‡ is simplified as Definition 4.3 shown later.

Before simplifying the modified procedure, we describe the relation between $\mathbb{U}(R)$ and S with respect to the innermost reduction. No rule $l \to r \in \mathbb{U}(R)$ such that there exists a rule $l' \to r'$ with $l \rhd l'\theta$ for some substitution θ is used in $\xrightarrow[\mathrm{i}]{}_{\mathbb{U}(R)}$ because no instance of l is an innermost redex. For this reason, we restrict the initial set of rules to $\mathbb{U}(R) \setminus S$. Roughly speaking $\mathbb{U}(R) \setminus S$ is the set of rules that are usable for $\xrightarrow[\mathrm{i}]{}_{\mathbb{U}(R)}$.

Definition 4.3 Let R be an OP-SN DCTRS over $\mathcal{F}$, and $\succ$ be a reduction order such that $\mathbb{U}(R) \subseteq \succ$. Let $S = \{\, l \to r \in \mathbb{U}(R) \mid \nexists l' \to r \in \mathbb{U}(R), l \sqsupset\!\!\!\!\sim l' \,\}$, $E_{(0)} = \{s \approx t \mid s \simeq t \in CP_{\varepsilon}(\mathbb{U}(R) \setminus S), s \not\equiv t\}$, $R_{(0)} = \{\, l \to r \in \mathbb{U}(R) \setminus S \mid \nexists l' \to r \in \mathbb{U}(R) \setminus S, l \sqsupset\!\!\!\!\sim l' \,\}$, and $i = 0$, then we apply the following steps:

1. (Orientation‡) select $s \approx t \in E_{(i)}$ such that $s \succ t$, $\texttt{root}(s)$ is a U symbol, and $CP(\{s \to t\}, R_{(i)} \cup \{s \to t\}) = \emptyset$;
2. $R_{(i+1)} := \{s \to t\} \cup R_{(i)}$, and $E_{(i+1)} := E_{(i)} \setminus \{s \simeq t\}$;
3. if $E_{(i+1)} \neq \emptyset$ then $i := i + 1$ and go to step 1, otherwise output $R_{(i+1)}$.

We call this procedure *the simplified completion procedure*.

It is clear that $E_{(i)} \supset E_{(i+1)}$ for every $i \geq 0$. Therefore, the simplified completion procedure always halts. Note that the simplified procedure doest not succeed for all input.

Theorem 4.4 *Let R be a pattern-stable OP-SN DCTRS over $\mathcal{F}$, and $\succ$ be a reduction order such that $\mathbb{U}(R) \subseteq \succ$. Let R' be a TRS obtained by the simplified completion procedure from R and $\succ$. Then all of the following hold: (1) R' is innermost-convergent, (2) $NF_{\to_{\mathbb{U}(R)}}(\mathcal{F}, \mathcal{V}) = NF_{\to_{R'}}(\mathcal{F}, \mathcal{V})$, and (3) $\xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}(R)} = \xrightarrow[\mathrm{i}]{*}{}^{!}_{R'}$ in $\mathcal{T}(\mathcal{F}, \mathcal{V})$.*

Note that (1) and (3) implies (2). The simplified procedure succeeds for $\mathbb{U}(\mathcal{I}nv(R_1))$ as well as for Example 4.2.

Similarly to the modified completion procedure, innermost-confluence of R is necessary for the simplified completion procedure to halt 'successfully'. Therefore, the simplified procedure is a method to show innermost-confluence of R;

5 Completion after Program Inversion

In this section, we apply the modified and simplified completion procedures to DCTRSs generated by the partial inversion compiler [25], that is, we apply the procedures as a *post-processor* of $\mathbb{U}(\mathcal{I}nv(\cdot))$ to the unraveled TRSs. First, we briefly introduce the feature of inverse systems for injective functions. Then, we show the results of experiments by an implementation of the framework.

We employ the partial inversion $\mathcal{I}nv$ in [25] that generates a partial inverse CTRS from a pair of a given constructor TRS and a specification, which we do not describe in detail here. For a defined symbol F, the defined symbol $\mathsf{Inv}F$

introduced by $\mathcal{I}nv$ represents a full inverse of F. We assume that constructor TRSs define main injective functions, and that the specifications require full inverses of the main functions.

5.1 Inverse DCTRSs of Injective Functions

We first define *injectivity* of TRSs [22], and then give a sufficient condition for input constructor TRSs whose inverse DCTRSs generated by $\mathcal{I}nv$ are convergent.

Definition 5.1 Let R be a terminating and innermost-confluent constructor TRS. A defined symbol F of R is called *injective (with respect to normal forms)* if the binary relation $\{(\langle s_1, \ldots, s_n\rangle, t) \mid s_1, \ldots, s_n, t \in NF_{\rightarrow_R}(\mathcal{F}, \mathcal{V}), F(s_1, \ldots, s_n) \xrightarrow{*}_R t\}$ is an injective mapping. R is called *injective (with respect to normal forms)* if all of its defined symbols are injective.

For example, the TRS R_1 in Section 1 is injective. Note that every injective TRS is non-erasing [22].

The following defined symbol $\mathsf{Reverse}$ computes the reverses of given lists:

$$R_4 = \left\{ \begin{array}{ll} \mathsf{Reverse}(xs) \to \mathsf{Rev}(xs, \mathsf{nil}), & \\ \mathsf{Rev}(\mathsf{nil}, ys) \to ys, & \mathsf{Rev}(x :: xs, ys) \to \mathsf{Rev}(xs, x :: ys) \end{array} \right\}$$

$\mathsf{Reverse}$ is injective but Rev is not. Thus, R_4 is not injective. In this case, the inverse TRS $\mathbb{U}(\mathcal{I}nv(R_4))$ is not terminating because $\mathbb{U}(\mathcal{I}nv(R_4))$ contains the rule $\mathsf{InvRev}(z) \to U_4(\mathsf{InvRev}(z), z)$. For this reason, we restrict ourselves to injective functions whose inverse TRSs are terminating. In [22], a sufficient condition has been shown for the full inversion compiler in [24] to generate convergent inverse DCTRSs from injective TRSs. The condition is also effective for the partial inversion compiler $\mathcal{I}nv$ [25].

Theorem 5.2 *Let R be a non-erasing, terminating and innermost-confluent constructor TRS.*

(i) *If $F \in \mathcal{D}_R$ is injective, then for all t, t_1 and $t_2 \in NF_{\rightarrow_{\mathcal{I}nv(R)}}(\mathcal{F}, \mathcal{V})$, $t_1 \mathrel{{}_{\mathrm{i}}\!\xleftarrow{*}}_{\mathcal{I}nv(R)} \mathsf{Inv}F(t) \xrightarrow[\mathrm{i}]{*}_{\mathcal{I}nv(R)} t_2$ implies $t_1 \equiv t_2$.*

(ii) *Suppose that for every rule $F(u_1, \ldots, u_n) \to r$ in R, if r is not a variable then the root symbol of r does not depend*[4] *on F. If $\mathcal{I}nv(R) \cap R = \emptyset$ then the DCTRS $\mathcal{I}nv(R)$ is OP-SN.*

Note that if the DCTRS $\mathcal{I}nv(R)$ is OP-SN then the TRS $\mathbb{U}(\mathcal{I}nv(R))$ is terminating [14]. Theorem 5.2 (i) shows that if $\mathcal{I}nv(R)$ is OP-SN, then $\mathcal{I}nv(R)$ has innermost-confluence that is necessary for successful runs of the simplified completion procedure. Note that $\mathcal{I}nv(R)$ is confluent if R is convergent [25]. When R does not satisfy the condition in Theorem 5.2 (ii), we directly check the termination of $\mathbb{U}(\mathcal{I}nv(R))$. In other words, when R satisfies the condition in Theorem 5.2 (ii), we are free of the termination check of $\mathbb{U}(\mathcal{I}nv(R))$ that is less efficient than the check of satisfying the condition.

[4] An n-ary symbol G of R *depends on a symbol* F if (G, F) is in the transitive closure of the relation $\{ (G', F') \mid G'(\cdots) \to C[F'(\cdots)] \in R \}$.

Table 1
the results of the experiments

example	CR by [2]	SN by Th.5.2	modified completion proc.			simplified completion proc.		
			result (cycles, time)	call	¬OVL	result	call	¬OVL
`du`			fail (1c, 0.71s)	1	—	success (0c, 0.71s)	1	
`snoc`		√	success (1c, 2.08s)	2	√	success (1c, 2.07s)	2	√
`snocrev`		√	success (2c, 4.29s)	3	√	success (2c, 4.28s)	3	√
`double`			fail (1c, 2.84s)	2	—	success (1c, 2.83s)	2	
`mirror`			fail (3c, 5.97s)	3	—	success (2c, 5.96s)	3	
`zip`	√	√	success (0c, 1.04s)	1	√	success (0c, 1.03s)	1	√
`inc`		√	success (1c, 2.80s)	2	√	success (1c, 2.80s)	2	√
`octbin`	√	√	success (0c, 7.33s)	1	√	success (0, 7.33s)	1	√
`treelist`		√	success (4c, 159.02s)	5		fail (2c, 5.47s)	2	—
`print-sexp`			success (6c, 28.20s)	7	√	success (6c, 28.28s)	7	√
`print-xml`			fail (3c, 9.49s)	3	—	success (2c, 9.47s)	3	

5.2 *Experiments*

In this subsection, we report the results of applying implementations of the modified and simplified completion procedures to 10 of 15 examples shown in [9].[5] These 15 examples are introduced for the experiments of the inversion compiler LRinv [9,10] where LRinv succeeds in inverting all of them. Those examples are written in the scheme script Gauche. The inverse TRSs of the scripts `snoc`, `snocrev` and `reverse` correspond to the TRSs $\mathbb{U}(\mathcal{I}nv(R_1))$, R_3 and $\mathbb{U}(\mathcal{I}nv(R_4))$, respectively. The constructor TRSs corresponding to the 5 scripts (`reverse` and so on) are not injective and the inverse TRSs obtained from them are not terminating. For this reason, we excluded those non-terminating examples from our experiments.

For some examples, there exists no appropriate LPO to guarantee termination of the input TRSs. For this reason, we employ the termination check '$(\bigcup_{j=0}^{i} R_{(i)}) \cup \{s \to t\}$ is terminating' instead of the input reduction orders, following the approach in [34]. The implementations are written in Standard ML of New Jersey, and they were executed under OS Vine Linux 4.2, on an Intel Pentium 4 CPU at 3 GHz and 1 GByte of primary memory. By the system call in SML/NJ, the implementations consult with AProVE 1.2 [6] as a termination prover at the ORIENTATION step. The implementations check termination of input TRSs in advance of the completion procedures. The timeout for checking termination is 300 seconds in every call of the prover. Note that 60 seconds timeout is enough, except for `treelist`.

The examples (`double`, `mirror` and `print-xml`) contain the special primitive operator `du` described in Section 4. Hence, they are not confluent but innermost-confluent. The operator `du` is an inverse of itself [9,10]. Thus, the TRS R_{du} is also an inverse system of itself. For this reason, exceptionally, the inversion compiler does not produce any rules of InvDup but introduces Du instead of InvDup.

Due to the syntactic properties provided by the inversion compiler, all inverse DCTRSs in the experiments are pattern-stable and strongly non-erasing. Thus, the procedures in this paper are applicable to all of them.

Table 1 summarizes the results of the experiments for our approach running on

[5] Unfortunately, the site shown in [9] is not accessible now. The examples are also described briefly as functional programs in [10], and some of the detailed programs can be found in [10].

10 of the 15 examples previously mentioned, which were translated by hand into TRSs.[6] The second column labeled with 'CR by [2]' shows whether the input TRS of the example is in the class shown in [2], in which the corresponding inverse TRS is orthogonal and thus confluent. In that case, the implementations only check termination of the inverse TRS. The third column labeled with 'SN by Th. 5.2' shows whether the input TRS satisfies the conditions in Theorem 5.2 (ii), that is, the corresponding inverse TRSs are terminating. Columns 4–6 show the results of the modified completion procedure. The fourth column shows the results of the modified completion ('success' or fail') with the numbers of running 'cycles' in the sense of Definition 2.1, and with the average time (seconds) of 5 trials. The number of cycles is the same as the number of applications of Orientation. As described above, the implementation checks the termination of input TRSs before the completion procedure starts. Thus, we have the results 'success (0c,$\cdots$) and 1 call of provers'. The sixth column labeled by '$\neg$OVL' shows whether or not the resulting TRSs are non-overlapping ($\surd$ means the resulting is non-overlapping, and '—' means no resulting TRS). None of the resulting TRSs has overlays while some of them are overlapping. Columns 7–9 show the results of the simplified completion procedure, and the meaning of those columns is the same as columns 4–6.

5.3 Translation Back into Functional Programs

In general, it is difficult to decide a priority of rewrite rules. However, we do not have to consider such a priority for R_2 that is computationally equivalent to $\mathcal{I}nv(R_1)$ because R_2 is not only confluent but also non-overlapping. On the other hand, every convergent constructor TRS can be easily translated back into a functional program. However, it is not easy to translate convergent TRSs that are not constructor systems, into functional programs even if the TRSs are non-overlapping. The reason is that some rules contains non-'well-formed' patterns in their left-hand sides, for instance, $\mathsf{InvSnoc}(\mathsf{nil})$ in $\mathbb{U}(\mathcal{I}nv(R_1))$.

In this subsection, we show a translation from R_2 into a SML program. Such a translation has not been automated yet but we believe that the automation is feasible.

The U symbols U_i^ρ introduced by the unraveling are often considered to express `let`, `if` or `case` clauses in functional programming languages. In the rewrite rules of R_2, the U symbol U_1 plays the role of a `case` clause as follows:

```
case InvSnoc( ys ) of (xs,y) => ( x::xs, y )
                    | InvSnoc( [] ) => (  []   , y )
```

where `InvSnoc( [] )` is not well-formed in the syntax of Standard ML. It is natural to write this fragment by introducing the extra `case` clause for `ys` as follows:

```
case ys of [] => ( [], y )
         | _  => (case InvSnoc( ys ) of (xs,y) => ( x::xs, y ) )
```

Thus, we translate the TRS R_2 into the following program:

```
fun InvSnoc( x::ys ) =
```

[6] The detail will be available from "`http://www.trs.cm.is.nagoya-u.ac.jp/repius/experiments/`".

```
case ys of [] => ( [], x )
         | _ => (case InvSnoc(ys) of (xs,y) => ( x::xs, y ) );
```

Other approaches to translations are possible. For example, we can consider U_1 as the composition of `if` and `let` clauses or as a 'local function' defined in $\mathsf{InvSnoc}$.

In all of the 10 examples, we succeeded in translating by hand the resulting convergent TRSs back into SML programs by means of the mechanism in this subsection although the resulting systems of `double`, `mirror`, `treelist`, and `print-xml` have overlapping.

6 Concluding Remarks

In this paper, we have shown that completion procedures are useful in generating (innermost-)convergent inverse TRSs of injective TRSs. The completion procedures can be also used for checking whether or not a (innermost-)convergent constructor TRS is injective. This is because if a given convergent constructor TRS is not injective, then the procedures never succeeds for the TRS. It is known to be undecidable in general whether or not a function is injective [5]. In [17], however, it is shown that injectivity of linear *treeless* functions is decidable. On the other hand, some of the examples we mentioned in the experiments are non-linear or non-treeless while the method in this paper is not decidable.

Completion procedures are effective for solving word problems, for transforming equations into equivalent convergent systems, or for proving inductive theorems. As far as we know, there is no application of completion to program modification, and there is no program transformation based on unravelings in order to produce computationally equivalent systems.

The modified completion procedure in this paper does not succeed for every confluent and OP-SN DCTRSs while the latest transformation [30] based on Viry's approach [33] always succeeds. Consider the example in Section 1 again. By the transformation in [30], we obtain the following convergent TRS instead of $\mathbb{U}(\mathcal{I}nv(R_1))$:

$$\left\{\begin{array}{r}
\mathsf{InvSnoc}([y],z) \to \{\langle \mathsf{nil}, y\rangle\},\qquad\qquad\qquad\qquad \\
\mathsf{InvSnoc}(x::ys,\bot) \to \mathsf{InvSnoc}(x::ys,\{\mathsf{InvSnoc}(ys,\bot)\}), \\
\mathsf{InvSnoc}(x::ys,\{\langle xs,y\rangle\}) \to \{\langle x::xs,y\rangle\},\qquad\qquad\qquad\quad \\
\mathsf{InvSnoc}(\{xs\},z) \to \{\mathsf{InvSnoc}(xs,\bot)\},\quad \{\{x\}\} \to \{x\}
\end{array}\right\}$$
$$\cup\,\{\, c(x_1,\ldots,\{x_i\},\ldots,x_n) \to \{c(x_1,\ldots,x_n)\} \mid c \in \{::, \langle,\rangle\}\,\}$$

where $\{\ \}$ and $\bot$ are special function symbols not in the original signature. In this system, the term $\mathsf{InvSnoc}([\mathsf{a},\mathsf{b},\mathsf{c}],\bot)$ has a unique normal form $\{\langle[\mathsf{a},\mathsf{b}],\mathsf{c}\rangle\}$. As described in Section 1, however, it is difficult to translate the convergent TRS into a functional program because the system contains special symbols $\{\ \}$ and $\bot$, and *overlapping* rules. On the other hand, the modified completion procedure in this paper unexpectedly succeeded for all the experiments where the DCTRSs are confluent, and the resulting systems of the procedure are often non-overlapping. Moreover, for the DCTRSs that are not confluent but innermost-confluent, we proposed the simplified completion procedure but it is not yet known whether or not the transformation in [30] is applicable.

The inversion compiler LRinv, the closest one to the method in this paper, has been proposed for injective functions written in a functional language [9,7,10]. This compiler translates source programs into programs in a grammar language, and then inverts the grammar programs into inverse grammar programs. To eliminate nondeterminism in the inverse programs, their compiler applies *LR parsing* to the inverse programs. The classes for which LR parsing and the completion procedure work successfully are not well known, which makes it difficult to compare LRinv and our method. However, LRinv succeeds in generating inverse functions from the 5 scripts (`reverse` and so on) that we excluded from the experiments, where the main functions call non-injective functions such as the accumulator Rev. From this fact, LRinv seems to be stronger than the method in this paper but there must be plenty of room on improving the principle of inversion used in the partial inversion complier in [25]. As future work, we plan to extend the partial inversion compiler for functions with accumulators such as Rev, and we also improve the modified and simplified completion procedures.

Acknowledgement

We are grateful to Germán Vidal and the anonymous reviewers for valuable comments for improving this paper. We also thank Tsubasa Sakata and Kazutoshi Seki for discussing correctness of the completion procedure on innermost reduction, and for helping the experiments. This work is partly supported by MEXT. KAKENHI #18500011, #20300010 and #20500008 and Kayamori Foundation of Informational Science Advancement.

References

[1] Abramov, S. and R. Glück, *The universal resolving algorithm and its correctness: Inverse computation in a functional language*, Science of Computer Programming **43** (2002), pp. 193–229.

[2] Almendros-Jiménez, J. M. and G. Vidal, *Automatic partial inversion of inductively sequential functions*, in: *Proc. of the 18th International Symposium on Implementation and Application of Functional Languages*, Lecture Notes in Computer Science **4449** (2006), pp. 253–270.

[3] Baader, F. and T. Nipkow, "Term Rewriting and All That," Cambridge University Press, United Kingdom, 1998.

[4] Dershowitz, N. and S. Mitra, *Jeopardy*, in: *Proceedings of the 10th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science **1631**, 1999, pp. 16–29.

[5] Fülöp, Z., *Undecidable properties of deterministic top-down tree transducers*, Theoretical Computer Science **134** (1994), pp. 311–328.

[6] Giesl, J., P. Schneider-Kamp and R. Thiemann, *Automatic termination proofs in the dependency pair framework*, in: *Proc. of the 3rd International Joint Conference on Automated Reasoning*, Lecture Notes in Computer Science **4130** (2006), pp. 281–286.

[7] Glück, R. and M. Kawabe, *A method for automatic program inversion based on LR(0) parsing*, Fundam. Inform. **66** (2005), no. 4, pp. 367–395.

[8] Gramlich, B., *On the (non-)existence of least fixed points in conditional equational logic and conditional rewriting*, in: *Proc. 2nd Int. Workshop on Fixed Points in Computer Science – Extended Abstracts* (2000), pp. 38–40.

[9] Kawabe, M. and Y. Futamura, *Case studies with an automatic program inversion system*, in: *Proc. of the 21st Conference of Japan Society for Software Science and Technology*, 6C-3, 2004, pp. 1–5.

[10] Kawabe, M. and R. Glück, *The program inverter LRinv and its structure*, in: *Proc. of the 7th International Symposium on Practical Aspects of Declarative Languages*, Lecture Notes in Computer Science **3350** (2005), pp. 219–234.

[11] Knuth, D. E. and P. B. Bendix, *Simple word problems in universal algebra*, in: *Computational Problems in Abstract Algebra* (1970), pp. 263–297.

[12] Krishna Rao, M. R. K., *Relating confluence, innermost-confluence and outermost-confluence properties of term rewriting systems*, Acta Informatica **33** (1996), no. 6, pp. 595–606.

[13] Lucas, S., *Context-sensitive computations in functional and functional logic programs*, Journal of Functional and Logic Programming **1998** (1998), no. 1.

[14] Lucas, S., C. Marché and J. Meseguer, *Operational termination of conditional term rewriting systems*, Information Processing Letters **95** (2005), no. 4, pp. 446–453.

[15] Marchiori, M., *Unravelings and ultra-properties*, in: *Proc. of the 5th International Conference on Algebraic and Logic Programming*, Lecture Notes in Computer Science **1139** (1996), pp. 107–121.

[16] Marchiori, M., *On deterministic conditional rewriting*, Computation Structures Group, Memo 405, MIT Laboratory for Computer Science (1997).

[17] Matsuda, K., Z. Hu, K. Nakano, M. Hamana and M. Takeichi, *Bidirectionalization transformation based on automatic derivation of view complement functions*, in: *Proc. of the 12th ACM SIGPLAN International Conference on Functional Programming* (2007), pp. 47–58.

[18] McCarthy, J., *The inversion of functions defined by Turing machines*, in: *Automata Studies*, Princeton University Press, 1956 pp. 177–181.

[19] Mogensen, T. Æ., *Semi-inversion of guarded equations.*, in: *Proc. of the 4th International Conference on Generative Programming and Component Engineering*, Lecture Notes in Computer Science **3676** (2005), pp. 189–204.

[20] Mogensen, T. Æ., *Semi-inversion of functional parameters*, in: *Proc. of the 2008 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation* (2008), pp. 21–29.

[21] Nishida, N. and M. Sakai, *Completion as Post-Process in Program Inversion of Injective Functions*, `http://www.trs.cm.is.nagoya-u.ac.jp/~nishida/papers/` (2008), the full version of this paper.

[22] Nishida, N., M. Sakai and T. Kato, *Convergent term rewriting systems for inverse computation of injective functions*, in: *Proc. of the 9th International Workshop on Termination* (2007), pp. 77–81.

[23] Nishida, N., M. Sakai and T. Sakabe, *On simulation-completeness of unraveling for conditional term rewriting systems*, IEICE Technical Report SS2004-18, (2004), vol. 104, No. 243, pp. 25–30.

[24] Nishida, N., M. Sakai and T. Sakabe, *Generation of inverse computation programs of constructor term rewriting systems*, The IEICE Trans. Inf.& Syst. **J88-D-I** (2005), no. 8, pp. 1171–1183 (in Japanese).

[25] Nishida, N., M. Sakai and T. Sakabe, *Partial inversion of constructor term rewriting systems*, in: *Proc. of the 16th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science **3467** (2005), pp. 264–278.

[26] Ohlebusch, E., *Termination of logic programs: Transformational methods revisited*, Applicable Algebra in Engineering, Communication and Computing **12** (2001), no. 1-2, pp. 73–116.

[27] Ohlebusch, E., "Advanced Topics in Term Rewriting," Springer-Verlag, 2002.

[28] Romanenko, A., *Inversion and metacomputation*, in: *Proc. of the Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, SIGPLAN Notices **26** (1991), pp. 12–22.

[29] Schernhammer, F. and B. Gramlich, *On proving and characterizing operational termination of deterministic conditional rewrite systems*, in: *Proc. of the 9th International Workshop on Termination* (2007), pp. 82–85.

[30] Serbanuta, T.-F. and G. Rosu, *Computationally equivalent elimination of conditions*, in: *Proc. of the 17th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science **4098** (2006), pp. 19–34.

[31] Terese, "Term Rewriting Systems," Cambridge University Press, 2003.

[32] Toyama, Y., *How to prove equivalence of term rewriting systems without induction*, Theoretical Computer Science **90** (1991), no. 2, pp. 369–390.

[33] Viry, P., *Elimination of conditions*, Journal of Symbolic Computation **28** (1999), no. 3, pp. 381–401.

[34] Wehrman, I., A. Stump and E. M. Westbrook, *Slothrop: Knuth-Bendix completion with a modern termination checker*, in: *Proc. of the 17th International Conference on Term Rewriting and Applications*, Lecture Notes in Computer Science **4098** (2006), pp. 287–296.

Refl

$$\overline{s \xrightarrow{*} t}$$

Tran

$$\frac{s \rightarrow t \quad t \xrightarrow{*} u}{s \xrightarrow{*} u}$$

Repl

$$\frac{s_1\sigma \xrightarrow{*} t_1\sigma \quad \cdots \quad s_n\sigma \xrightarrow{*} t_n\sigma}{C[l\sigma] \rightarrow C[r\sigma]} \quad \text{if } l \rightarrow r \Leftarrow s_1 \rightarrow t_1 \wedge \cdots \wedge s_n \rightarrow t_n \in R$$

Fig. A.1. Derivation rules for the reduction of R.

A Operational Termination of DCTRSs

Following the notion of *operational termination* proposed in [14], we here give a definition of operational termination for operational innermost reduction.

Definition A.1 Let R be an OP-SN DCTRS. The set of (finite) *proof trees* for R and the *head* of a proof tree are inductively defined as follows.

- An *open goal* G, where G is either $s \rightarrow t$ or $s \xrightarrow{*} t$ for some terms s and t, is a proof tree. In this case, $\mathsf{head}(G) = G$ is the head of the proof tree.
- A *derivation tree* T, denoted by $\dfrac{T_1 \quad \cdots \quad T_n}{G}(\Delta)$, is a proof tree, where G is as in the first case, Δ is one of the derivation rules in Fig. A.1, and $T_1, \ldots, T_n$ are proof trees such that $\dfrac{\mathsf{head}(T_1) \quad \cdots \quad \mathsf{head}(T_n)}{G}$ is an instance of Δ. In this case, $\mathsf{head}(T) = G$.

A proof tree is said to be *closed* if it does not contain any open goals.

Definition A.2 A proof tree T is a *prefix* of a proof tree T', written in $T \subset T'$, if there are one or more open goals $G_1, \ldots, G_n$ in T such that T' is obtained from T by replacing each G_i with a derivation tree T_i such that $\mathsf{head}(T_i) = G_i$. An *infinite proof tree* is an infinite sequence $T_0, T_1, \cdots$ of finite proof trees such that $T_i \subset T_{i+1}$ for all $i \geq 0$.

Definition A.3 A proof tree T is *well-formed* if it is either an open goal, a closed proof tree, or a derivation tree of the form $\dfrac{T_1 \quad \cdots \quad T_n}{G}(\Delta)$ where T_j is a well-formed proof tree for all $1 \leq j \leq n$ and there is an $i \leq n$ such that T_i is not closed, T_j is closed for all $j < i$, and T_k is an open goal for all $k > i$. An infinite proof tree is *well-formed* if it consists of well-formed proof trees.

Operational termination is characterized by the absence of infinite well-formed proof trees.

Definition A.4 A DCTRS R is *operationally terminating* if there are no infinite well-formed proof trees.

B Proofs

B.1 Proof of Theorem 3.3

Lemma B.1 *Let $\mathcal{F}$ be a signature, $\mathcal{F}'$ be an extended signature of $\mathcal{F}$ ($\mathcal{F} \subseteq \mathcal{F}'$), μ be a replacement map for $\mathcal{F}' \setminus \mathcal{F}$ such that $\mu(F) = \{1, \ldots, n\}$ for any n-ary symbol $F \in \mathcal{F}$, R' be a pattern-stable OP-SN DCTRS over $\mathcal{F}$, and R be a TRS over $\mathcal{F}'$ such that $NF_{\underset{\mathrm{i}}{\to}_R}(\mathcal{F}, \mathcal{V}) \supseteq NF_{\underset{\mathrm{i}}{\to}_{(R',\mu)}}(\mathcal{F}, \mathcal{V})$, and $\{l \to r \in R' \mid \mathtt{root}(l)$ appears in $\mathsf{Pat}(R)\} \subseteq R$. Let σ and σ' be substitutions such that $\mathcal{R}an(\sigma) \subseteq NF_{\underset{\mathrm{i}}{\to}_R}(\mathcal{F}, \mathcal{V})$, $\mathcal{R}an(\sigma') \subseteq NF_{\underset{\mathrm{i}}{\to}_R}(\mathcal{F}', \mathcal{V})$ and $x\sigma \overset{*}{\underset{\mathrm{i}}{\to}}{}^{!}_{(R',\mu)} x\sigma'$ for all $x \in \mathcal{D}om(\sigma)$. For all patterns $p \in \mathsf{Pat}(R)$, $p\sigma$ is irreducible with respect to $\underset{\mathrm{i}}{\to}_R$ if and only if $p\sigma'$ is irreducible with respect to $\underset{\mathrm{i}}{\to}_{(R',\mu)}$.*

Proof. We prove this lemma by induction on term structure of p. Since the case of $\mathtt{root}(p) \notin \mathcal{D}_R$ can be shown straightforwardly by the induction hypothesis, we only show the renaming case of $\mathtt{root}(p) \in \mathcal{D}_R$.

Let $p \equiv F(p_1, \ldots, p_n)$ where $F \in \mathcal{D}_R$. Suppose that $p\sigma$ is irreducible with respect to $\underset{\mathrm{i}}{\to}_R$. Then, $p_1\sigma, \ldots, p_n\sigma$ are irreducible with respect to $\underset{\mathrm{i}}{\to}_R$ since $p\sigma$ is irreducible. By the induction hypothesis, $p_1\sigma', \ldots, p_n\sigma'$ are irreducible with respect to $\underset{\mathrm{i}}{\to}_{(R',\mu)}$. Assume that $p\sigma'$ is reducible with respect to $\underset{\mathrm{i}}{\to}_{(R',\mu)}$. Then, it follows that pattern-stability of R' that there is a rule $F(x_1, \ldots, x_n) \to r \in R'$. It follows from the assumption that $F(x_1, \ldots, x_n) \to r \in R$. Hence, $p\sigma$ is reducible with respect to $\underset{\mathrm{i}}{\to}_R$. This contradicts the assumption that $p\sigma$ is irreducible. Therefore, $p\sigma'$ is irreducible with respect to $\underset{\mathrm{i}}{\to}_{(R',\mu)}$.

On the other hand, in the case that $p\sigma'$ is irreducible with respect to $\underset{\mathrm{i}}{\to}_{(R',\mu)}$, we can similarly show that $p\sigma$ is irreducible with respect to $\underset{\mathrm{i}}{\to}_R$. □

Note that Lemma B.1 is applicable to not only pairs of R and $\mathbb{U}_{\mathrm{cs}}(R)$ but also pairs of TRSs without μ.

Theorem 3.3 is a direct consequence of the following lemma.

Lemma B.2 *Let R be a pattern-stable OP-SN DCTRS over a signature $\mathcal{F}$, s and t be terms in $\mathcal{T}(\mathcal{F}, \mathcal{V})$, and σ and σ' be substitutions such that $\mathcal{R}an(\sigma) \subseteq NF_{\to_R}(\mathcal{F}, \mathcal{V})$, $\mathcal{D}om(\sigma|_{\mathcal{V}ar(s)}) \cap \mathcal{D}om(\sigma|_{\mathcal{V}ar(t)}) = \emptyset$, and $x\sigma \overset{*}{\underset{\mathrm{i}}{\to}}_{\mathbb{U}_{\mathrm{cs}}(R)} x\sigma'$ for all $x \in \mathcal{D}om(\sigma)$. Then, if $s\sigma \overset{*}{\underset{\mathrm{i}}{\to}}{}^{!}_R t\sigma$, then $s\sigma' \overset{*}{\underset{\mathrm{i}}{\to}}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} t\sigma'$.*

Proof. We allow $\underset{\mathrm{i}}{\to}_R$ to perform as parallel reductions. We prove this lemma by induction on the lexicographic products of the term structure s, the level n and length k of $\xrightarrow[(n),\mathrm{i}]{k}_R$. We only show the most difficult case (cf. [23]).

Consider the following sequence:

$$s\sigma \overset{*}{\underset{\mathrm{i}}{\to}}{}^{\varepsilon<,!}_R l\sigma \xrightarrow[(n),\mathrm{i}]{}_R r\sigma \overset{*}{\underset{\mathrm{i}}{\to}}{}^{!}_R t\sigma$$

where $\rho : l \to r \Leftarrow s_1 \to t_1 \in R$, $s_1\sigma \xrightarrow[(n-1),\mathrm{i}]{*}{}^{!}_R t_1\sigma$, $\{l \to U_1^\rho(s_1, \overrightarrow{X}), U_1^\rho(t_1, \overrightarrow{X}) \to r\} \subseteq \mathbb{U}(R)$, $X = \mathcal{V}ar(l)$, $\mathcal{V}ar(l, r, s_1, t_1) \cap \mathcal{V}ar(s, t) = \emptyset$, and $\mathcal{R}an(\sigma) \subseteq NF_{\to_R}(\mathcal{F}, \mathcal{V})$.

Let σ' be a substitution such that $x\sigma \overset{*}{\underset{\mathrm{i}}{\to}}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} x\sigma'$ for all $x \in \mathcal{D}om(\sigma)$. Then, by the induction hypothesis, we have $s\sigma' \overset{*}{\underset{\mathrm{i}}{\to}}{}^{\varepsilon<,!}_{\mathbb{U}_{\mathrm{cs}}(R)} l\sigma'$, $s_1\sigma' \overset{*}{\underset{\mathrm{i}}{\to}}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} t_1\sigma'$, and $r\sigma'$

$\xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} t\sigma'$. It follows from Lemma B.1 that $l\sigma'$ is an innermost redex of $\mathbb{U}_{\mathrm{cs}}(R)$ and $t_1\sigma'$ is a normal form of $\mathbb{U}_{\mathrm{cs}}(R)$. Therefore, we have the following sequence:

$$\begin{aligned} s\sigma \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<}_{\mathbb{U}_{\mathrm{cs}}(R)} s\sigma' \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_{\mathbb{U}_{\mathrm{cs}}(R)} l\sigma' \xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)} U_1^\rho(s_1, \overrightarrow{X})\sigma \\ \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_{\mathbb{U}_{\mathrm{cs}}(R)} U_1^\rho(t_1, \overrightarrow{X})\sigma \xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)} r\sigma' \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} t\sigma'. \end{aligned}$$

□

B.2 Proof of Theorem 3.4

Theorem 3.4 is a direct consequence of the following lemmas.

Lemma B.3 *Let R be a pattern-stable OP-SN DCTRS over a signature $\mathcal{F}$, s and t be terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$, and θ and θ' be substitutions such that $\mathcal{R}an(\theta) \subseteq NF_{\rightarrow_R}(\mathcal{F},\mathcal{V})$, $\mathcal{R}an(\theta') \subseteq NF_{\rightarrow_{\mathbb{U}_{\mathrm{cs}}(R)}}(\mathcal{F}_{\mathbb{U}},\mathcal{V})$, and $x\theta \xrightarrow[\mathrm{i}]{*}_{\mathbb{U}_{\mathrm{cs}}(R)} x\theta'$ for all $x \in \mathcal{D}om(\theta)$. If $s \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} t\theta'$, then $s \xrightarrow[\mathrm{i}]{*}_R t\theta$.*

Proof. We allow $\xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)}$ to perform as parallel reductions. We prove this lemma by induction on the lexicographic products of the term structure s and the length k of $\xrightarrow[\mathrm{i}]{k}_{\mathbb{U}_{\mathrm{cs}}(R)}$. We only show the most difficult case (cf. [23]).

Consider the following sequence:

$$s \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_{\mathbb{U}_{\mathrm{cs}}(R)} l\sigma' \xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)} U_1^\rho(s_1, \overrightarrow{X})\sigma' \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_{\mathbb{U}_{\mathrm{cs}}(R)} U_1^\rho(t_1, \overrightarrow{X})\sigma' \xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)} r\sigma' \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} t\theta'$$

where $\rho : l \to r \Leftarrow s_1 \to t_1 \in R$, $X = \mathcal{V}ar(l)$, and $\mathcal{R}an(\sigma') \subseteq NF_{\rightarrow_{\mathbb{U}_{\mathrm{cs}}(R)}}(\mathcal{F}_{\mathbb{U}},\mathcal{V})$.

Let σ be a substitution such that $\mathcal{R}an(\sigma) \subseteq NF_{\xrightarrow[\mathrm{i}]{}_R}(\mathcal{F},\mathcal{V})$, and $x\sigma \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} x\sigma'$ for all $x \in \mathcal{D}om(\sigma)$. Then, we have $r\sigma \xrightarrow[\mathrm{i}]{*}_{\mathbb{U}_{\mathrm{cs}}(R)} r\sigma' \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} t\theta'$, $s_1\sigma \xrightarrow[\mathrm{i}]{*}_{\mathbb{U}_{\mathrm{cs}}(R)} s_1\sigma'$, and $l\sigma \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_{\mathbb{U}_{\mathrm{cs}}(R)} l\sigma'$. Since $t_1\sigma'$ is irreducible with respect to $\xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)}$, it follows from Lemma B.1 that $t_1\sigma$ is irreducible with respect to $\xrightarrow[\mathrm{i}]{}_R$, and hence $s_1\sigma \xrightarrow[\mathrm{i}]{*}_{\mathbb{U}_{\mathrm{cs}}(R)} s_1\sigma' \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} t_1\sigma'$. Then, it follows from the induction hypothesis that $r\sigma \xrightarrow[\mathrm{i}]{*}{}^{!}_R t\theta$, $s_1\sigma \xrightarrow[\mathrm{i}]{*}{}^{!}_R t_1\sigma$, and $s \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_R l\sigma$. Therefore, we have $s \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_R l\sigma \xrightarrow[\mathrm{i}]{>}_R r\sigma \xrightarrow[\mathrm{i}]{*}_R t\theta$. □

Lemma B.4 *Let R be a strongly non-erasing OP-SN DCTRS over a signature $\mathcal{F}$, and s and t be terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$. Then, $s \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} t$ implies $s \xrightarrow[\mathrm{i}]{*}{}^{!}_R t$.*

Proof. We prove this lemma by induction on the lexicographic products of the term structure s and the length k of $\xrightarrow[\mathrm{i}]{k}_{\mathbb{U}_{\mathrm{cs}}(R)}$. We only show the most difficult case (cf. [23]).

Consider the following sequence:

$$s \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_{\mathbb{U}_{\mathrm{cs}}(R)} l\theta \xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)} U_1^\rho(s_1, \overrightarrow{X})\theta \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_{\mathbb{U}_{\mathrm{cs}}(R)} U_1^\rho(t_1, \overrightarrow{X})\theta \xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)} r\theta \xrightarrow[\mathrm{i}]{*}{}^{!}_{\mathbb{U}_{\mathrm{cs}}(R)} t$$

where $\rho : l \to r \Leftarrow s_1 \to t_1 \in R$, $X = \mathcal{V}ar(l)$, and $\mathcal{R}an(\theta) \subseteq NF_{\rightarrow_{\mathbb{U}_{\mathrm{cs}}(R)}}(\mathcal{F}_{\mathbb{U}},\mathcal{V})$. Since any U symbol is not consumed, it follows from non-erasingness and context-sensitivity that $\mathcal{R}an(\theta) \subseteq \mathcal{T}(\mathcal{F},\mathcal{V})$. It is clear that $NF_{\rightarrow_{\mathbb{U}_{\mathrm{cs}}(R)}}(\mathcal{F},\mathcal{V}) \subseteq NF_{\rightarrow_R}(\mathcal{F},\mathcal{V})$. Thus, by the induction hypothesis, we have $s \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_R l\theta$, $s_1\theta \xrightarrow[\mathrm{i}]{*}{}^{!}_R t_1\theta$, and $r\theta \xrightarrow[\mathrm{i}]{*}{}^{!}_R t$. Therefore, we have $s \xrightarrow[\mathrm{i}]{*}{}^{\varepsilon<,!}_R l\theta \xrightarrow[\mathrm{i}]{}_R r\theta \xrightarrow[\mathrm{i}]{*}{}^{!}_R t$. □

B.3 Proof of Theorem 3.5

Theorem 3.5 is a direct consequence of the following lemmas.

Lemma B.5 *Let R be a DCTRS over a signature $\mathcal{F}$, and s and t be terms in $\mathcal{T}(\mathcal{F}_{\mathbb{U}}, \mathcal{V})$. Suppose that every term at irreducible positions of s is a normal form of $\mathbb{U}(R)$. Then, all of the following hold:*

- *$s \xrightarrow[\mathrm{i}]{}_{\mathbb{U}(R)} t$ if and only if $s \xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)} t$, and*
- *if $s \xrightarrow[\mathrm{i}]{}_{\mathbb{U}(R)} \cup \xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)} t$, then every term at irreducible positions of t is a normal form of $\mathbb{U}(R)$.*

Proof. This lemma follows from the definitions of $\xrightarrow[\mathrm{i}]{}_{\mathbb{U}(R)}$ and $\xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)}$. $\square$

Lemma B.6 *Let R be a DCTRS over a signature $\mathcal{F}$, s be a term in $\mathcal{T}(\mathcal{F}, \mathcal{V})$, and t be a term in $\mathcal{T}(\mathcal{F}_{\mathbb{U}}, \mathcal{V})$. If $s \xrightarrow[\mathrm{i}]{*}_{\mathbb{U}(R)} \cup \xrightarrow[\mathrm{i}]{*}_{\mathbb{U}_{\mathrm{cs}}(R)} t$, then every term at irreducible positions of t is a normal form of $\mathbb{U}(R)$.*

Proof. This lemma follows from the replacement map of $\mathbb{U}_{\mathrm{cs}}(R)$, and the definitions of $\xrightarrow[\mathrm{i}]{}_{\mathbb{U}(R)}$ and $\xrightarrow[\mathrm{i}]{}_{\mathbb{U}_{\mathrm{cs}}(R)}$. $\square$

B.4 Proof of Theorem 4.1

It follows from the side condition '$\mathtt{root}(s)$ is a U symbol' of the Orientation that $NF_{\to_{\mathbb{U}(R)}}(\mathcal{F}, \mathcal{V}) = NF_{\to_{R'}}(\mathcal{F}, \mathcal{V})$. It is clear that there is a run of completion from $(\mathbb{U}(R), \emptyset)$ to (E_0, R_0). It follows from the correctness of the completion (Theorem 7.3.5 in [3]) that R' is convergent and $\xleftrightarrow{*}_{\mathbb{U}(R)} \subseteq \xrightarrow{*}_{R'} \cdot \xleftarrow{*}_{R'}$. Let $\xrightarrow[1]{} = \{(s,t) \mid s,t \in \mathcal{T}(\mathcal{F}, \mathcal{V}),\ s \xrightarrow{+}_{R} t\}$ and $\xrightarrow[2]{} = \{(s,t) \mid s,t \in \mathcal{T}(\mathcal{F}, \mathcal{V}),\ s \xrightarrow{+}_{R'} t\}$. Then, we have $\xleftrightarrow[1]{*} \subseteq \xleftrightarrow[2]{*}$, confluence of $\xrightarrow[2]{}$, termination of $\xrightarrow[1]{}$, and $NF_1 = NF_2$ where NF_i is the set of normal forms with respect to $\xrightarrow[i]{}$. Therefore, it follows from Theorem 3.3 in [32] that $\xrightarrow[1]{*} = \xrightarrow[2]{*}$ in $\mathcal{T}(\mathcal{F}, \mathcal{V}) \times NF_2$, and hence $\xrightarrow{*!}_{\mathbb{U}(R)} = \xrightarrow{*!}_{R'}$ in $\mathcal{T}(\mathcal{F}, \mathcal{V})$. $\square$

B.5 Proof of Theorem 4.4

We denote $\mathbb{U}(R) \setminus S$ by $\Delta_{\mathsf{in}}(\mathbb{U}(R))$.

It follows from the side condition '$\mathtt{root}(s)$ is a U symbol' of the Orientation that (2) $NF_{\to_{\mathbb{U}(R)}}(\mathcal{F}, \mathcal{V}) = NF_{\to_{R'}}(\mathcal{F}, \mathcal{V})$.

We first show that (1) R' is innermost-convergent and (4) $\xrightarrow[\mathrm{i}]{}_{\Delta_{\mathsf{in}}(\mathbb{U}(R))} \subseteq (\xrightarrow[\mathrm{i}]{+!}_{R'} \circ \xleftarrow[\mathrm{i}]{*!}_{R'}) \cup (\xrightarrow[\mathrm{i}]{*!}_{R'} \circ \xleftarrow[\mathrm{i}]{+!}_{R'})$.

B.5.1 Theorem 4.4 (1)

It is clear that R' is terminating.

We discuss a property of added rules $(R' \setminus \Delta_{\mathsf{in}}(\mathbb{U}(R)))$. It follows from the side condition of Orientation‡ that every added rule $s \to t \in R' \setminus \Delta_{\mathsf{in}}(\mathbb{U}(R))$ satisfies all of the following:

- there is a critical pair $s \simeq t \in E_{(0)} = CP_{\varepsilon}(\Delta_{\mathsf{in}}(\mathbb{U}(R)))$,

- every instance $s\theta$ is always the innermost redex of R' that is reducible only to $t\theta$ if $\mathcal{R}an(\theta) \subseteq NF_{\to_{R'}}(\mathcal{F}_{\mathbb{U}}, \mathcal{V})$, and
- $CP(\{s \to t\}, R') = \emptyset$.

It follows from $CP(\{s \to t\}, R') = \emptyset$ that $CP_\varepsilon(R') = CP_\varepsilon(\Delta_{\mathsf{in}}(\mathbb{U}(R)))$. Let $s \simeq t \in CP_\varepsilon(R')$. Suppose that $s \to t \in R' \setminus \Delta_{\mathsf{in}}(\mathbb{U}(R))$. Then, it follows from the above second claim that $s\theta \xrightarrow[\mathrm{i}]{}_{R'} t\theta$ for every substitution θ such that $\mathcal{R}an(\theta) \subseteq NF_{\to_{R'}}$. It is shown in [12] that a terminating TRS S is innermost-confluent if $s'\theta'$ and $t'\theta'$ are joinable with respect to $\xrightarrow[\mathrm{i}]{*}_S$ for every $s' \simeq t' \in CP_\varepsilon(S)$ and substitution θ' such that $\mathcal{R}an(\theta') \subseteq NF_S$. Hence, R' is innermost-convergent.

B.5.2 (4)

It follows from the definition of the simplified completion procedure that R' contains all rules in $R_{(0)}$. Consider a rule $l \to r \in R_{(0)}$. It follows from the pattern-preserving property and the properties of added rules that if $l\sigma$ is an innermost redex of $R_{(0)}$ then $l\sigma'$ is also an innermost redex of R' where $x\sigma \xrightarrow[\mathrm{i}]{*}{}^!_{R'} x\sigma'$ for all $x \in \mathcal{D}om(\sigma)$. Thus, we have $l\sigma \xrightarrow[\mathrm{i}]{*}{}^!_{R'} l\sigma' \xrightarrow[\mathrm{i}]{}_{\{l\to r\}} r\sigma'$ and $r\sigma \xrightarrow[\mathrm{i}]{*}_{R'} r\sigma'$. Since R' is terminating, $r\sigma'$ has a normal form of R', and hence $l\sigma \xrightarrow[\mathrm{i}]{+}{}^!_{R'} \circ \xleftarrow[\mathrm{i}]{*}{}^!_{R'} r\sigma$.

Consider a reduction $l\sigma \xrightarrow[\mathrm{i}]{}_{\Delta_{\mathsf{in}}(\mathbb{U}(R))} r\sigma$ by a removed rule $l \to r \in \Delta_{\mathsf{in}}(\mathbb{U}(R)) \setminus R_{(0)}$ such that l is an instance of another rule $l' \to r' \in R_{(0)}$. Let $l \equiv l'\theta$. Then, we have another reduction $l\sigma \equiv l'\theta\sigma \xrightarrow[\mathrm{i}]{}_{\Delta_{\mathsf{in}}(\mathbb{U}(R))} r'\theta\sigma$. For this reduction, we have $r'\theta\sigma \xrightarrow[\mathrm{i}]{*}{}^!_{R'} r'\theta\sigma'$ where $x\theta\sigma \xrightarrow[\mathrm{i}]{*}{}^!_{R'} x\sigma'$ for all $x \in \mathcal{D}om(\sigma)$. Then, we have another reduction $r\sigma \xrightarrow[\mathrm{i}]{*}_{R'} r\sigma'$. As described above, $E_{(0)}$ contains their critical pair $r \simeq r'\theta$. When the procedure halts successfully, it follows from the assumption that R' has either $r'\theta \to r$ or $r \to r'\theta$.

- Case of $r'\theta \to r \in R'$. As described above, $r'\theta\sigma'$ is an innermost redex of R'. Thus, we have $r'\theta\sigma' \xrightarrow[\mathrm{i}]{}_{R'} r\sigma'$. Since R' is terminating, $r\sigma'$ has a normal form of R', and hence $r'\theta\sigma' \xrightarrow[\mathrm{i}]{+}{}^!_{R'} \circ \xleftarrow[\mathrm{i}]{*}{}^!_{R'} r\sigma'$.
- Case of $r \to r'\theta \in R'$. Similarly to the first case, we have $r'\theta\sigma' \xrightarrow[\mathrm{i}]{*}{}^!_{R'} \circ \xleftarrow[\mathrm{i}]{+}{}^!_{R'} r\sigma'$.

Therefore, we have $l\sigma \xrightarrow[\mathrm{i}]{*}_R r'\theta\sigma' \ (\xrightarrow[\mathrm{i}]{+}{}^!_{R'} \circ \xleftarrow[\mathrm{i}]{*}{}^!_{R'}) \cup (\xrightarrow[\mathrm{i}]{*}{}^!_{R'} \circ \xleftarrow[\mathrm{i}]{+}{}^!_{R'})\ r\sigma' \xleftarrow[\mathrm{i}]{*}_{R'} r\sigma$.

From the above discussion and $\xrightarrow[\mathrm{i}]{}_{\mathbb{U}(R)} = \xrightarrow[\mathrm{i}]{}_{\Delta_{\mathsf{in}}(\mathbb{U}(R))}$, we have $\xrightarrow[\mathrm{i}]{}_{\Delta_{\mathsf{in}}(\mathbb{U}(R))} \subseteq (\xrightarrow[\mathrm{i}]{+}{}^!_{R'} \circ \xleftarrow[\mathrm{i}]{*}{}^!_{R'}) \cup (\xrightarrow[\mathrm{i}]{*}{}^!_{R'} \circ \xleftarrow[\mathrm{i}]{+}{}^!_{R'})$.

B.5.3 Theorem 4.4 (3)

Let $\xrightarrow[1]{} = \{(s,t) \mid s,t \in \mathcal{T}(\mathcal{F},\mathcal{V}),\ s \xrightarrow[\mathrm{i}]{}_{\mathbb{U}(R)} t\}$ and $\xrightarrow[2]{} = \{(s,t) \mid s,t \in \mathcal{T}(\mathcal{F},\mathcal{V}),\ s \xrightarrow[\mathrm{i}]{+}_{R'} t\}$. Then, we have $\xleftrightarrow[1]{*} \subseteq \xleftrightarrow[2]{*}$, confluence of $\xrightarrow[2]{}$, termination of $\xrightarrow[1]{}$, and $NF_1 = NF_2$ where NF_i is the set of normal forms with respect to $\xrightarrow[i]{}$. Therefore, it follows from Theorem 3.3 in [32] that $\xrightarrow[1]{*} = \xrightarrow[2]{*}$ in $\mathcal{T}(\mathcal{F},\mathcal{V})$, and hence $\xrightarrow[\mathrm{i}]{*}{}^!_{\mathbb{U}(R)} = \xrightarrow[\mathrm{i}]{*}{}^!_{R'}$ in $\mathcal{T}(\mathcal{F},\mathcal{V})$. $\square$

B.6 Proof of Theorem 5.2 (i)

All normal forms over $\mathcal{F}$ that are reachable from $\mathsf{Inv}F(t)$ are either of the form $\langle t_1,\ldots,t_n\rangle$ or itself ([25]). In the latter case, $t_1 \equiv t_2 \equiv \mathsf{Inv}F(t)$. We only consider the former case.

Assume that for some F and normal form t, there are tuples of normal forms $\langle s_1,\ldots,s_n\rangle$ and $\langle t_1,\ldots,t_n\rangle$ such that $\langle s_1,\ldots,s_n\rangle \overset{*}{\underset{\mathrm{i}}{\leftarrow}}_{\mathcal{I}nv(R)} \mathsf{Inv}F(t) \xrightarrow{*}_{\mathcal{I}nv(R)} \langle t_1,\ldots,t_n\rangle$ and $\langle s_1,\ldots,s_n\rangle \not\equiv \langle t_1,\ldots,t_n\rangle$. Then, it follows from the correctness of $\mathcal{I}nv$ (shown in [25]) and Theorem 3.6 that $F(s_1,\ldots,s_n) \xrightarrow{*}_R t \xleftarrow{*}_R F(t_1,\ldots,t_n)$. Injectivity of F implies $\langle s_1,\ldots,s_n\rangle \equiv \langle t_1,\ldots,t_n\rangle$. However, this contradicts $\langle s_1,\ldots,s_n\rangle \not\equiv \langle t_1,\ldots,t_n\rangle$. □

B.7 Proof of Theorem 5.2 (ii)

The outline of the proof follows the proof of the corresponding theorem in [22].

We show *quasi-simplifyingness* of $\mathcal{I}nv(R)$. Then, operational termination of $\mathcal{I}nv(R)$ follows from quasi-simplifyingness.

We first give a definition of *quasi-simplifyingness* [27].

Definition B.7 A deterministic 3-CTRS S over a signature $\mathcal{F}$ is called *quasi-simplifying* if there is an extension $\mathcal{F}'$ of the signature $\mathcal{F}$ (so $\mathcal{F} \subseteq \mathcal{F}'$) and a simplification ordering $\succ$ on $\mathcal{T}(\mathcal{F}',\mathcal{V})$ that satisfies the following conditions for every rule $l \to r \Leftarrow s_1 \to t_1 \wedge \cdots \wedge s_k \to t_k \in S$, every substitution σ: $\mathcal{V} \to \mathcal{T}(\mathcal{F}',\mathcal{V})$, and every $0 \le i < k$:

(i) if $s_j\sigma \succeq t_j\sigma$ for every $1 \le j \le i$, then $l\sigma \succ s_{i+1}\sigma$,

(ii) if $s_j\sigma \succeq t_j\sigma$ for every $1 \le j \le k$, then $l\sigma \succ r\sigma$.

Lemma B.8 ([27,14]) *Quasi-simplifyingness implies operational termination.*

We abbreviate the sequence $t_1,\ldots,t_n$ of terms to $\overrightarrow{t}$.

The following properties of $\mathcal{I}nv(R)$ follows from [24,25].

Proposition B.9 *Let R be a constructor TRS. Suppose that $\mathcal{I}nv(R) \cap R = \emptyset$. Then, every rewrite rule $F(\overrightarrow{u}) \to r$ in R is transformed by the inversion $\mathcal{I}nv$ into a deterministic conditional rule*

$$\mathsf{Inv}F(r',\overrightarrow{u'}) \to \langle\overrightarrow{w}\rangle \Leftarrow \textstyle\bigwedge_{i=1}^{k} \mathsf{Inv}F_i(y_i,\overrightarrow{u'_i}) \to \langle\overrightarrow{w_i}\rangle$$

where

- *each F_i is a defined of R,*
- *F_i and F_j $(i \ne j)$ appear at different positions of r,*
- *r' is a constructor term of R,*
- *$\overrightarrow{u'}$, $\overrightarrow{w}$, $\overrightarrow{u'_1},\ldots,\overrightarrow{u'_k}$, $\overrightarrow{w_1},\ldots,\overrightarrow{w_k}$ are sequences of constructor terms,*
- *each variable y_i is not in $\mathcal{V}ar(r,\overrightarrow{u})$,*
- *y_i and y_j $(i \ne j)$ are different,*
- *each y_i appears exactly once in either r' or $\overrightarrow{w_j}$ $(j < i)$ and not in $\overrightarrow{w_l}$ and $\overrightarrow{w}$ $(i \le l)$,*

- $\mathcal{V}ar(\overrightarrow{u'_i}) \subseteq \mathcal{V}ar(r', \overrightarrow{u'})$, *and*
- $\mathcal{V}ar(w) \subseteq \mathcal{V}ar(r', \overrightarrow{u'}, \overrightarrow{w_1}, \dots, \overrightarrow{w_k}) \setminus \{y_1, \dots, y_k\}$.

Moreover, the conditional rule has the following properties:

(a) *if the original rule is non-erasing, then each variable in $\mathcal{V}ar(\overrightarrow{u})$ occurs in either r' or $\overrightarrow{w_i}$,*

(b) *if r is a constructor term of R, then $k = 0$ and $r' \equiv r$, and*

(c) *if the root symbol G of r is a defined symbol of R, then $r' \equiv y_1$ and $G = F_1$.*

Lemma B.10 *Let $s \in \mathcal{T}(\mathcal{F}, \mathcal{V}) \setminus \mathcal{V}$, $t \in \mathcal{T}(\mathcal{G}, V)$ for signatures $\mathcal{F}$ and $\mathcal{G}$ ($\subseteq \mathcal{F}$), σ be a substitution, $>_{\mathsf{lpo}}$ be the lexicographic path ordering determined by a precedence $>$ on $\mathcal{F}$. If $\mathtt{root}(s) > G$ for all $G \in \mathcal{G}$ and $s\sigma >_{\mathsf{lpo}} x\sigma$ for all $x \in \mathcal{V}ar(t)$, then $s\sigma >_{\mathsf{lpo}} t\sigma$.*

Proof. We prove this by induction on structure of t.

- Case of $t \equiv x \in \mathcal{V}$. It follows from the assumption that $s\sigma >_{\mathsf{lpo}} x\sigma \equiv t\sigma$.
- Let $t \equiv G(t_1, \dots, t_m)$ where $G \in \mathcal{G}$. By the induction hypothesis, we have $s\sigma >_{\mathsf{lpo}} t_i\sigma$. Now we have $\mathtt{root}(s) > G$ and $s\sigma >_{\mathsf{lpo}} t_i\sigma$. Thus, it follows from the definition of LPOs that $s\sigma >_{\mathsf{lpo}} G(t_1\sigma, \dots, t_m\sigma) \equiv t\sigma$.

□

Lemma B.11 *Let R be a non-erasing constructor TRS that satisfies the assumption in Theorem 5.2. Then, $\mathcal{I}nv(R)$ is quasi-simplifying.*

Proof. Let $\mathcal{F}_{\mathcal{I}nv}$ be the set of defined symbols of $\mathcal{I}nv(R)$ such that $\{\mathsf{Inv}F \mid F \in \mathcal{D}_R\}$. We suppose that tuples symbols $\langle\rangle$ are in $\mathcal{F}$.

Let $>_{\mathsf{lpo}}$ be the lexicographic path ordering determined by the precedence $>$ that satisfies all of the following:

- $\mathsf{Inv}F > G$ for all $\mathsf{Inv}F \in \mathcal{F}_{\mathcal{I}nv}$ and $G \in \mathcal{F}$, and
- if $F \in \mathcal{D}_R$ calls $G \in \mathcal{D}_R$ and G does not depend on F, then $\mathsf{Inv}F > \mathsf{Inv}G$. Otherwise, $\mathsf{Inv}F = \mathsf{Inv}G$.

It is clear that the special rules $\mathsf{Inv}F(F(\overrightarrow{x})) \to \langle\overrightarrow{x}\rangle \in \mathcal{I}nv(R)$ satisfy $\mathsf{Inv}F(F(\overrightarrow{x})) >_{\mathsf{lpo}} \langle\overrightarrow{x}\rangle$. We only show the rule obtained from $F(\overrightarrow{u}) \to r \in R$ satisfies the conditions of quasi-simplifyingness.

Let the conditional rule obtained from $F(\overrightarrow{u}) \to r \in R$ be $\mathsf{Inv}F(r', \overrightarrow{u'}) \to \langle\overrightarrow{w}\rangle \Leftarrow \bigwedge_{i=1}^{k} \mathsf{Inv}F_i(y_i, \overrightarrow{u'_i}) \to \langle\overrightarrow{w_i}\rangle$. Consider the case that $r \in \mathcal{V}$. Let $r \equiv x$. It follows from Proposition B.9 that $k = 0$, $r' \equiv x$ and $\mathcal{V}ar(\overrightarrow{u}) = \mathcal{V}ar(\overrightarrow{w}) = \{x\}$, and hence $\mathsf{Inv}F(x, \overrightarrow{u'}) >_{\mathsf{lpo}} \langle\overrightarrow{w}\rangle$ by Lemma B.10. Therefore, the conditional rule satisfies the conditions of quasi-simplifyingness.

Consider the remaining case that $r \notin \mathcal{V}$. We first prove the following claim for every i ($1 \leq i \leq k$) by induction on i:

if $\mathsf{Inv}F_j(y_j, \overrightarrow{u'_j})\sigma \geq_{\mathsf{lpo}} \langle\overrightarrow{w_j}\rangle\sigma$ for $1 \leq j < i$, then $\mathsf{Inv}F(r', \overrightarrow{u'})\sigma >_{\mathsf{lpo}} \mathsf{Inv}F_i(y_i, \overrightarrow{u'_i})\sigma$.

- *Base case* ($i = 1$). It follows from Proposition B.9 that $\mathtt{root}(r')$ is a constructor of R, $y_1 \in \mathcal{V}ar(r')$, $\mathcal{V}ar(u'_1) \subseteq \mathcal{V}ar(r', \overrightarrow{u'})$, and $\overrightarrow{u'_1}$ is a sequence of constructor terms.

- Case that $\texttt{root}(r)$ is a constructor of R. By the assumption on $>$, we have $\mathsf{Inv}F \geq \mathsf{Inv}F_1$. It follows from the definition of LPOs and Lemma B.10 that $\mathsf{Inv}F(r', \overrightarrow{u'}) >_{\mathsf{lpo}} \mathsf{Inv}F_1(y_1, \overrightarrow{u'_1})$. Since $>_{\mathsf{lpo}}$ is closed under substitutions, $\mathsf{Inv}F(r', \overrightarrow{u'})\sigma >_{\mathsf{lpo}} \mathsf{Inv}F_1(y_1, \overrightarrow{u'_1})\sigma$.
- The remaining case that $\texttt{root}(r)$ is a defined symbol of R. By assumption, F_1 does not depend on F. Thus, it follows from the construction of $>$ that $\mathsf{Inv}F > \mathsf{Inv}F_1$, and hence $\mathsf{Inv}F(y_1, \overrightarrow{u'}) >_{\mathsf{lpo}} \mathsf{Inv}F_1(y_1, \overrightarrow{u'_1})$. Therefore, we have $\mathsf{Inv}F(r', \overrightarrow{u'})\sigma >_{\mathsf{lpo}} \mathsf{Inv}F_1(y_1, \overrightarrow{u'_1})\sigma$.

- *Induction case* ($i > 1$). Suppose that $\mathsf{Inv}F_j(y_j, \overrightarrow{u'_1})\sigma \geq_{\mathsf{lpo}} \langle \overrightarrow{w_j} \rangle\sigma$ for $1 \leq j < i$. It follows from Proposition B.9 that $\mathcal{V}ar(\overrightarrow{u'_i}) \subseteq \mathcal{V}ar(r', \overrightarrow{u'})$, and there exist some j ($< i$) such that $y_i \in \mathcal{V}ar(\overrightarrow{w_j})$. By the induction hypothesis, we have $\mathsf{Inv}F(r', \overrightarrow{u'})\sigma >_{\mathsf{lpo}} \mathsf{Inv}F_j(y_j, \overrightarrow{u'_j})\sigma$. It is clear that $\mathsf{Inv}F_j(y_j, \overrightarrow{u'_j})\sigma >_{\mathsf{lpo}} \langle \overrightarrow{w_j} \rangle\sigma >_{\mathsf{lpo}} y_i\sigma$. It follows from the construction of $>$ that $\mathsf{Inv}F \geq \mathsf{Inv}F_i$. Therefore, it follows from Lemma B.10 that $\mathsf{Inv}F(r', \overrightarrow{u'})\sigma >_{\mathsf{lpo}} \mathsf{Inv}F_i(y_i, \overrightarrow{u'_i})\sigma$.

Therefore, it follows from the above claim that the conditional rule satisfies the first condition of quasi-simplifyingness.

Next we show that the conditional rule satisfies the second condition of quasi-simplifyingness. Suppose that $\mathsf{Inv}F_j(y_j, \overrightarrow{u'_j})\sigma \geq_{\mathsf{lpo}} \langle \overrightarrow{w_j} \rangle\sigma$ for $1 \leq j \leq k$. Then we have $\mathsf{Inv}F_j(y_j, \overrightarrow{u'_j})\sigma >_{\mathsf{lpo}} \langle \overrightarrow{w_j} \rangle\sigma$ because $\mathsf{Inv}F_j > \langle\rangle$. It follows from non-erasingness of R and Proposition B.9 that $\mathcal{V}ar(\overrightarrow{w}) \subseteq \mathcal{V}ar(r', \overrightarrow{u'}, \overrightarrow{w_1}, \ldots, \overrightarrow{w_k}) \setminus \{y_1, \ldots, y_k\}$. Let $x \in \mathcal{V}ar(\overrightarrow{w})$.

- *Case of $x \in \mathcal{V}ar(r')$.* It is clear that $\mathsf{Inv}F(r', \overrightarrow{u'}) \rhd x$, and hence $\mathsf{Inv}F(r')\sigma >_{\mathsf{lpo}} x\sigma$.
- *The remaining case.* There exist some j ($1 \leq j \leq k$) such that $x \in \mathcal{V}ar(\overrightarrow{w_j})$, and hence $\langle \overrightarrow{w_j} \rangle \rhd x$. It follows from $\mathsf{Inv}F_j(y_j, \overrightarrow{u'_j})\sigma >_{\mathsf{lpo}} \langle \overrightarrow{w_j} \rangle\sigma$ and the first condition of quasi-simplifyingness that $\mathsf{Inv}F(r', \overrightarrow{u'})\sigma >_{\mathsf{lpo}} \mathsf{Inv}F_j(y_j, \overrightarrow{u'_j})\sigma$, and hence $\mathsf{Inv}F(r', \overrightarrow{u'})\sigma >_{\mathsf{lpo}} \langle \overrightarrow{w_j} \rangle\sigma >_{\mathsf{lpo}} x\sigma$.

Thus we have $\mathsf{Inv}F(r', \overrightarrow{u'})\sigma >_{\mathsf{lpo}} x\sigma$. It follows from Lemma B.10 that $\mathsf{Inv}F(r', \overrightarrow{u'})\sigma >_{\mathsf{lpo}} \langle \overrightarrow{w} \rangle\sigma$. Therefore, the conditional rule satisfies the second condition of quasi-simplifyingness. □

Theorem 5.2 (ii) follows from Lemmas B.8 and B.11.