Determinization of Conditional Term Rewriting Systems

Masanori Nagashima[a], Masahiko Sakai[a], Toshiki Sakabe[a]

[a] *Graduate School of Information Science, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464–8603 Japan*

Abstract

This paper discusses determinization of conditional term rewriting systems with oriented constructor rules. We present a rule-based transformation system, which transforms a non-deterministic one into a deterministic one, together with examples of the transformation. We prove that the transformation system is simulation sound and simulation complete. We also prove that the transformation system is complete for some class by introducing a strategy for the transformation system.

Keywords: deterministic conditional term rewriting system, functional program, rule-based transformation

1. Introduction

Conditional term rewriting systems (CTRSs, for short) are extended term rewriting systems whose rewrite rules may be guarded by conditions. CTRSs play an important role in integration of logic and functional programming as well as in algebraic specification of abstract data types. CTRSs are classified by the form of conditions into several classes. The class of oriented CTRSs is one of them and is suitable as a model of functional programs since rules guarded by oriented conditions correspond to the 'let' construct of functional programming languages. For example, consider the following ML-like functional program:

```
datatype Nat = Zero | S of Nat;
fun add(x, Zero) = x
```

Email addresses: nagashima@sakabe.i.is.nagoya-u.ac.jp (Masanori Nagashima), sakai@is.nagoya-u.ac.jp (Masahiko Sakai), sakabe@is.nagoya-u.ac.jp (Toshiki Sakabe)

```
|  add(x, S(y)) =
       let val z = add(x, y)
       in  S(z)
       end;
```

This program is naturally represented as the following CTRS:

$$\mathcal{R}_{\mathsf{add}} = \left\{ \begin{array}{l} \mathsf{add}(x, \mathsf{0}) \to x, \\ \mathsf{add}(x, \mathsf{s}(y)) \to \mathsf{s}(z) \Leftarrow \mathsf{add}(x, y) \to z \end{array} \right\},$$

where s represents the successor function.

The determinacy [1] of CTRSs is a basic property of oriented CTRSs so that important properties such as operational termination [2] are discussed on deterministic CTRSs. A rule $l \to r \Leftarrow s_1 \to t_1; \cdots; s_n \to t_n$ is *deterministic* if each variable occurring in s_i also appears in at least one of l, t_1, ..., and t_{i-1}. This means that the values of variables of s_i are determined when the condition $s_i \to t_i$ is evaluated as far as the conditions of the rule are evaluated from left to right. Since $\mathcal{R}_{\mathsf{add}}$ is deterministic, i.e., its two rules are deterministic, the evaluation can naturally progress. The variable z in the *right-hand side* of the second rule is called an *extra variable*: a variable not occurring in the *left-hand side* of the same rule. The value of such a variable is, however, determined properly when evaluating a term. For example, $\mathsf{add}(\mathsf{0}, \mathsf{s}(\mathsf{0}))$ is rewritten to $\mathsf{s}(\mathsf{0})$ by the second rule because z is determined to be $\mathsf{0}$ from the *conditional part*.

On the other hand, consider the following non-deterministic CTRS $\mathcal{E}_{\mathsf{sub}} \cup \mathcal{R}_{\mathsf{add}}$, where

$$\mathcal{E}_{\mathsf{sub}} = \left\{ \; \mathsf{sub}(z, y) \to x \Leftarrow \mathsf{add}(x, y) \to z \; \right\}.$$

There is a problem in applying the rule $\mathsf{sub}(z, y) \to x \Leftarrow \mathsf{add}(x, y) \to z$ to $\mathsf{sub}(\mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0}))$. By matching $\mathsf{sub}(z, y)$ to $\mathsf{sub}(\mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0}))$ the values of variables z and y are determined, but not the value of x in the right-hand side of the rewrite rule, and hence the condition $\mathsf{add}(x, y) \to z$ cannot be evaluated. The term $\mathsf{sub}(\mathsf{s}(\mathsf{0}), \mathsf{s}(\mathsf{0}))$ is, nevertheless, rewritten to $\mathsf{0}$ with the substitution $\{x \mapsto \mathsf{0},\ y \mapsto \mathsf{s}(\mathsf{0}),\ z \mapsto \mathsf{s}(\mathsf{0})\}$ since $\mathsf{add}(\mathsf{0}, \mathsf{s}(\mathsf{0}))$ is reduced to $\mathsf{s}(\mathsf{0})$.

Our interest is in constructing a systematic way that transforms non-deterministic CTRSs like $\mathcal{E}_{\mathsf{sub}}$ into equivalent deterministic CTRSs like $\mathcal{R}_{\mathsf{sub}}$:

$$\mathcal{R}_{\mathsf{sub}} = \left\{ \begin{array}{l} \mathsf{sub}(z, \mathsf{0}) \to z, \\ \mathsf{sub}(\mathsf{s}(z), \mathsf{s}(y)) \to x \Leftarrow \mathsf{sub}(z, y) \to x \end{array} \right\}.$$

In this paper, we assume that CTRSs are all oriented, i.e., the rules of any CTRS are all oriented because the class of deterministic CTRSs is defined as a subclass of the oriented CTRSs [3, 4].

We concentrate on pure-constructor systems and constructor-based reductions. A *pure-constructor system* [3] is such a system that for any rewrite rule and any condition, the left-hand side is a pattern, and the right-hand side is a constructor term. A *constructor-based reduction* [5] is a reduction such that ranges of matching substitutions are restricted to constructor terms. These restrictions are natural if we consider functional programs with eager evaluation. Moreover these restrictions make entire discussion simple and clear drastically. One may think that the innermost reduction is more general and natural. However, it is known that innermost reduction of conditional systems is not well-defined [6]. In pure-constructor systems right-hand sides of the rules are restricted to patterns. This is however no problem. For example, nested function call of a rule $f(x) \to h(g(x))$ can be represented by decomposing the nesting into conditions like $f(x) \to z \Leftarrow g(x) \to y; h(y) \to z$.

In Section 3, we define a rule-based transformation system $\mathcal{DT}_0$ consisting of rules Nar, Fld_0 and Def, which is inspired by unfold/fold transformations. We give examples of the transformation in order to explain intuitive meaning of each transformation rule. In Section 4, we prove simulation soundness of the transformation system $\mathcal{DT}_0$. On the other hand, it is shown that $\mathcal{DT}_0$ is not simulation complete. We propose three side conditions (C1), (C2) and (C3) for Fld_0 and define $\mathcal{DT}_{123}$ to be the transformation system $\mathcal{DT}_0$ having Fld_1, Fld_2 and Fld_3 instead of Fld_0. Then we prove that $\mathcal{DT}_{123}$ is simulation complete. In Section 5, we discuss syntactic conditions related to inductive theorems used in the folding transformations and the condition, called the level condition, used in (C2). In Section 6, we prove that the transformation system $\mathcal{DT}_3$, which is $\mathcal{DT}_0$ having Fld_3, is complete if no conditional part of rules in $\mathcal{E}$ contains a symbol defined in $\mathcal{E}$ and $\mathcal{R}$ is quasi-decreasing. Here, the completeness means that for any input CTRS there exists a transformation sequence that leads to a deterministic CTRS. In order to prove the completeness we introduce a strategy for $\mathcal{DT}_3$ and show that the strategy is terminating and produces a deterministic CTRS. In Section 7, we present larger examples. In Section 8, we refer to related works on inversion computation and on program generation. In Section 9, we give concluding remarks.

2. Preliminaries

In this section, we introduce notations used later. We assume that the reader is familiar with basic concepts of term rewriting [7, 8].

Let $\mathcal{F}$ be a *signature*, a finite set of *function symbols* accompanied with a mapping *arity* which maps each function symbol f to a natural number $\mathrm{arity}(f)$. $\mathcal{F}$ is assumed to be partitioned into two disjoint sets $\mathcal{D}$ of *defined symbols* and $\mathcal{C}$ of *constructors*, that is, $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$. Let $\mathcal{V}$ be a countably infinite set of *variables* such that $\mathcal{F} \cap \mathcal{V} = \emptyset$. A function symbol $g \in \mathcal{F}$ is called a *constant symbol* if $\mathrm{arity}(g) = 0$.

The set of *terms* over $\mathcal{F}$ and $\mathcal{V}$ is denoted by $\mathcal{T}(\mathcal{F}, \mathcal{V})$, the set of variables occurring in terms $t_1, \ldots, t_n$ by $\mathcal{V}ar(t_1, \ldots, t_n)$, and the set of function symbols occurring in terms $t_1, \ldots, t_n$ by $\mathcal{F}un(t_1, \ldots, t_n)$. We will abuse $\mathcal{V}ar(P_1, \ldots, P_n)$ to denote the set of (free) variables occurring in objects $P_1, \ldots, P_n$ such as terms, equations, conditions, rewrite rules and so on, and similarly $\mathcal{F}un(P_1, \ldots, P_n)$ for function symbols. A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ is called *ground* if $\mathcal{V}ar(t) = \emptyset$. The set of all ground terms is denoted by $\mathcal{T}(\mathcal{F})$. A term $t \in \mathcal{T}(\mathcal{C}, \mathcal{V})$ is called a *constructor term*. A term of the form $f(t_1, \ldots, t_n)$ is called a *pattern* in $\mathcal{T}(\mathcal{F}, \mathcal{V})$ if $f \in \mathcal{D}$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C}, \mathcal{V})$. We denote the *root symbol* of a term t by $\mathrm{root}(t)$.

Let $\Box \notin \mathcal{F} \cup \mathcal{V}$ be a special constant symbol, called a *hole*. A *context* is a term $C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{V})$ with exactly one occurrence of $\Box$. We write $C[t]$ for the term obtained from C by replacing the occurrence of $\Box$ in C with a term t. A term t is a *subterm* of a term s, denoted by $s \trianglerighteq t$, if there exists a context C such that $C[t] = s$. In the case $C \neq \Box$, t is called a *proper subterm* of u, denoted by $s \triangleright t$.

A *substitution* is a function $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $\{x \mid \sigma(x) \neq x\}$ is finite. The set $\{x \mid \sigma(x) \neq x\}$ is denoted by $\mathcal{D}om(\sigma)$ and called the *domain* of σ. A substitution σ with $\mathcal{D}om(\sigma) = \{x_1, \ldots, x_n\}$ is often denoted by $\{x_1 \mapsto \sigma(x_1), \ldots, x_n \mapsto \sigma(x_n)\}$. A substitution σ with $\mathcal{D}om(\sigma) = \emptyset$ is called the *empty substitution*. A substitution σ can be extended to $\sigma : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ in a natural way. We sometimes simply write $t\sigma$ instead of $\sigma(t)$ for the term to which σ maps t.

A substitution σ is *ground* if $x\sigma \in \mathcal{T}(\mathcal{F})$ for all $x \in \mathcal{D}om(\sigma)$. A substitution σ is a *constructor substitution* if $x\sigma \in \mathcal{T}(\mathcal{C}, \mathcal{V})$ for all $x \in \mathcal{D}om(\sigma)$. A term s is an *instance* of a term t if $s = t\sigma$ for some substitution σ. Especially it is a *constructor-instance* if σ is a constructor substitution. The *composition* $\sigma\theta$ of two substitutions σ and θ is defined as $x(\sigma\theta) = (x\sigma)\theta$. The restriction

of a substitution σ to a set V of variables, denoted by $\sigma|_V$, is the substitution obtained from σ by restricting the domain of σ to V. That is, $x(\sigma|_V) = x\sigma$ if $x \in V$ and $x(\sigma|_V) = x$ otherwise. A substitution δ is an *extension* of a substitution θ if $\mathcal{D}om(\delta) \supseteq \mathcal{D}om(\theta)$ and $x\delta = x\theta$ for any $x \in \mathcal{D}om(\theta)$.

An *equation* is a pair of terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, which is denoted by $s \sim t$. A substitution σ is a *unifier* of a set E of equations if $s\sigma = t\sigma$ for every $s \sim t \in E$. E is said to be *unifiable* if there is a unifier of E. If σ is a unifier of $\{s \sim t\}$, we say that σ is a unifier of s and t, and also that s and t are unifiable. A unifier σ of E is a *most general unifier* of E if for any unifier θ of E, there exists a substitution δ such that $\theta = \sigma\delta$. It is known that most general unifiers of E are unique up to renaming. We denote the most general unifier of E by $\mathrm{mgu}(E)$.

A binary relation $\rightharpoonup$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is *monotonic* if for all $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, $s \rightharpoonup t$ implies $f(\ldots, s, \ldots) \rightharpoonup f(\ldots, t, \ldots)$, and $\rightharpoonup$ is *stable under (constructor) substitutions* if for all $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ and for all (constructor) substitution σ, $s \rightharpoonup t$ implies $s\sigma \rightharpoonup t\sigma$. The *reflexive transitive closure* of $\rightharpoonup$ is denoted by $\overset{*}{\rightharpoonup}$. We write $\overset{n}{\rightharpoonup}$ for the n-fold composition of $\rightharpoonup$, i.e., $\overset{0}{\rightharpoonup}$ is the identity relation and $\overset{n+1}{\rightharpoonup} = \overset{n}{\rightharpoonup} \circ \rightharpoonup$, where $\circ$ denotes the composition operator for binary relations.

We introduce a relational logic over $\mathcal{T}(\mathcal{F}, \mathcal{V})$. An atom is a pair of terms s and t, denoted by $s \to t$. Formulas are atoms, existentially quantified formulas, conjunction of formulas and implication of formulas. Satisfaction of a formula φ by a pair of a relation $\rightharpoonup$ on $\mathcal{T}(\mathcal{F}, \mathcal{V})$ and a substitution $\sigma : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$, denoted by $\langle \rightharpoonup, \sigma \rangle \models \varphi$, is inductively defined as follows:

$\langle \rightharpoonup, \sigma \rangle \models u \to v$ iff $u\sigma \rightharpoonup v\sigma$
$\langle \rightharpoonup, \sigma \rangle \models \varphi \wedge \varphi'$ iff $\langle \rightharpoonup, \sigma \rangle \models \varphi$ and $\langle \rightharpoonup, \sigma \rangle \models \varphi'$
$\langle \rightharpoonup, \sigma \rangle \models \varphi \Rightarrow \varphi'$ iff $\langle \rightharpoonup, \sigma \rangle \models \varphi$ implies $\langle \rightharpoonup, \sigma \rangle \models \varphi'$
$\langle \rightharpoonup, \sigma \rangle \models \exists x.\varphi$ iff
  $\langle \rightharpoonup, \sigma' \rangle \models \varphi$ for some ground constructor substitution σ' which
  is an extension of $\sigma|_{\mathcal{D}om(\sigma)\setminus\{x\}}$ such that $\mathcal{D}om(\sigma') = \mathcal{D}om(\sigma) \cup \{x\}$.

A sequence of conditions $u_1 \to v_1; \cdots; u_n \to v_n$ is regarded as conjunction $u_1 \to v_1 \wedge \cdots \wedge u_n \to v_n$.

An *oriented conditional rewrite rule* (*rewrite rule*, for short) is a formula in the form of $l \to r \Leftarrow u_1 \to v_1; \cdots; u_n \to v_n$ $(n \geq 0)$. $l \to r$ and $u_1 \to v_1; \cdots; u_n \to v_n$ are called the *body part* and *conditional part* of the

rule, respectively. Terms l and r are called the *left-hand side* and *right-hand side* of the rule. $u_i \to v_i$ is called a *condition* of the rule. A rewrite rule whose conditional part is empty is called an *unconditional rule*, denoted as $l \to r$ by omitting $\Leftarrow$.

A condition $u \to v$ is called a *pattern condition* if u is a pattern and v is a constructor term. A rewrite rule $l \to r \Leftarrow c$ is called a *pure-constructor rule* if l is a pattern, r is a constructor term, and the conditions of c are all pattern conditions. Note that pure-constructor rules are also *normal* [3]. A rewrite rule $l \to r \Leftarrow c$ is said to be of *type 3* [4] if $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l) \cup \mathcal{V}ar(c)$, and of *type 4* if no restriction. A conditional rewrite rule $l \to r \Leftarrow u_1 \to v_1; \cdots; u_n \to v_n$ of type 3 is called *deterministic* [1] if $\mathcal{V}ar(u_i) \subseteq \mathcal{V}ar(l, v_1, \ldots, v_{i-1})$ for all i $(1 \leq i \leq n)$. A variable occurrence in $u_i \to v_i$ is said to be *known* if it is in $\mathcal{V}ar(l, v_1, \ldots, v_{i-1})$, and *unknown* otherwise. The deterministic rules of type 4 are defined in the same way as type 3. Two different rules $l_1 \to r_1 \Leftarrow c_1$ and $l_2 \to r_2 \Leftarrow c_2$ *overlay* if l_1 and l_2 are unifiable.

An *oriented conditional term rewriting system* (CTRS, for short) is a finite set $\mathcal{R}$ of oriented conditional rewrite rules. For a CTRS $\mathcal{R}$, let $\mathcal{D}_\mathcal{R}$ be the set of all defined symbols occurring in the left-hand sides of the rules of $\mathcal{R}$, i.e., $\mathcal{D}_\mathcal{R} = \{\mathrm{root}(l) \mid l \to r \Leftarrow c \in \mathcal{R}\}$. A *pure-constructor system* is a CTRS $\mathcal{R}$ whose rewrite rules are all pure-constructor rewrite rules[1]. A CTRS $\mathcal{R}$ is *deterministic* if all rewrite rules of $\mathcal{R}$ are deterministic.

The *k-level reduction* $\to_\mathcal{R}^{(k)}$ of $\mathcal{R}$ is inductively defined as follows:

- $\to_\mathcal{R}^{(0)} = \emptyset$, and
- $\to_\mathcal{R}^{(j)} = \to_\mathcal{R}^{(j-1)} \cup$
  $\Big\{(C[l\sigma], C[r\sigma]) \mid l \to r \Leftarrow c \in \mathcal{R},\ C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{V}),$
  σ is a substitution, $\langle \overset{*}{\to}_\mathcal{R}^{(j-1)}, \sigma\rangle \models c\Big\}$ for $j > 0$.

The *reduction* $\to_\mathcal{R}$ is defined as $\bigcup_{k\geq 0} \to_\mathcal{R}^{(k)}$.

The constructor-based reduction $\xrightarrow[\mathsf{c}]{}_\mathcal{R}$ of a CTRS $\mathcal{R}$ [5] can be defined in the same way as the ordinary reduction of a CTRS except that matching substitutions are restricted to constructor substitutions. Note that $\xrightarrow[\mathsf{c}]{}_\mathcal{R} \subseteq \to_\mathcal{R}$.

[1]The class of pure-constructor CTRSs is the same as the class of normalized TRSs in [9]

A deterministic oriented CTRS $\mathcal{R}$ of type 3 is *quasi-decreasing* [10, 2] if there exists a well-founded partial order $\succ$ on $\mathcal{T}(\mathcal{F},\mathcal{V})$ satisfying

- $(\to_{\mathcal{R}} \cup \rhd) \subseteq \succ$, and
- for every rule $l \to r \Leftarrow u_1 \to v_1; \cdots; u_n \to v_n$, substitution σ and i $(1 \leq i < n)$

$$\text{if } \langle \to_{\mathcal{R}}, \sigma \rangle \models u_j \to v_j \text{ for every } j\ (0 \leq j \leq i) \text{ then } l\sigma \succ u_{i+1}\sigma.$$

It is known that the class of operationally terminating CTRSs [2] and that of quasi-decreasing CTRSs are equivalent [2].

3. Transformation system

In this section, we define a transformation system, which is inspired by unfold/fold transformations like [11].

All CTRSs in this paper are oriented and pure-constructor systems. We use $\mathcal{R}$ for a fixed deterministic CTRS and $\mathcal{E}$ for a (possibly) non-deterministic CTRS to be determinized where $\mathcal{F}un(\mathcal{R}) \cap \mathcal{D}_{\mathcal{E}} = \emptyset$.

We give the notion of inductive theorems before defining the transformation system. We say that a formula φ is an *inductive theorem of* $\mathcal{R}$ if $\langle \xrightarrow[c]{}_{\mathcal{R}}, \sigma \rangle \models \varphi$ for every ground constructor substitution σ such that $\mathcal{D}om(\sigma) = \mathcal{V}ar(\varphi)$. For example, $\mathsf{add3}(x_1,x_2,x_3) \to y \Leftrightarrow \exists z.\mathsf{add}(x_1,x_2) \to z; \mathsf{add}(x_3,z) \to y$ is an inductive theorem of $\mathcal{R}_{\mathsf{add3}} \cup \mathcal{R}_{\mathsf{add}}$, where

$$\mathcal{R}_{\mathsf{add3}} = \left\{\ \mathsf{add3}(x_1,x_2,x_3) \to y \Leftarrow \mathsf{add}(x_1,x_2) \to z; \mathsf{add}(x_3,z) \to y\ \right\}.$$

Let $\mathcal{V}ar(c) \setminus \mathcal{V}ar(l,r) = \{y_1,\ldots,y_k\}$. We write $X_{c\setminus l,r}$ for $y_1 \cdots y_k$. This is typically used in the form of $l \to r$ OP $\exists X_{c\setminus l,r}.c$, where OP is either $\Rightarrow$, $\Leftarrow$, or $\Leftrightarrow$. Note that $l \to r \Leftarrow c$ is an inductive theorem if and only if $l \to r \Leftarrow \exists X_{c\setminus l,r}.c$ is an inductive theorem. We have also that $l \to r \Rightarrow \exists X_{c\setminus l,r}.c$ is an inductive theorem if $l \to r \Rightarrow c$ is an inductive theorem, however the reverse does not hold.

Definition 3.1 (Transformation System $\mathcal{DT}_0$). *The transformation system $\mathcal{DT}_0$ consists of the rules given in Figure 1. We write $\mathcal{E} \Rrightarrow_{\mathcal{R}} \mathcal{E}'$ for a one-step transformation from an oriented pure-constructor CTRS $\mathcal{E}$ (on an oriented pure-constructor deterministic CTRS $\mathcal{R}$) into another CTRS $\mathcal{E}'$ by applying downward one of transformation rules in Figure 1. We say that a transformation of the form $\mathcal{E} = \mathcal{E}_1 \Rrightarrow_{\mathcal{R}} \cdots \Rrightarrow_{\mathcal{R}} \mathcal{E}_n$* succeeds *if $\mathcal{E}_n$ is deterministic.*

(Nar: Narrowing)

$$\frac{\mathcal{E}_1 = \mathcal{E} \cup \{\rho : \ l \to r \Leftarrow c; u \to v\}}{\mathcal{E} \cup \{l\sigma \to r\sigma \Leftarrow c\sigma; c'\sigma \mid \rho' : \ l' \to r' \Leftarrow c' \in \mathcal{E}_1 \cup \mathcal{R}, \\ \mathcal{V}ar(\rho) \cap \mathcal{V}ar(\rho') = \emptyset, \ \sigma = \mathrm{mgu}(\{u \sim l', v \sim r'\})\}}.$$

($\mathbf{Fld_0}$: Folding) If $l' \to r' \Leftrightarrow \exists X_{c' \setminus l', r'}.c'$ is an inductive theorem of $\mathcal{E}_1 \cup \mathcal{R}$ such that $\mathcal{V}ar(l, r, c) \cap \mathcal{V}ar(c') \subseteq \mathcal{V}ar(l', r')$, then

$$\frac{\mathcal{E}_1 \ = \ \mathcal{E} \cup \{l \to r \Leftarrow c; c'\}}{\mathcal{E} \cup \{l \to r \Leftarrow c; l' \to r'\}}.$$

(Def: Definition) If $l \to r \Leftarrow c$ is a pure-constructor rewrite rule with $\mathrm{root}(l) \notin \mathcal{D}_{\mathcal{E} \cup \mathcal{R}}$, then

$$\frac{\mathcal{E}}{\mathcal{E} \cup \{l \to r \Leftarrow c\}}.$$

Figure 1: Transformation Rules of $\mathcal{DT}_0$

We give intuitive explanations of transformations by using simple examples. The first example explains Nar and Fld_0.

Example 1. *Consider CTRSs $\mathcal{E}_{\mathsf{sub}}$ and $\mathcal{R}_{\mathsf{add}}$ in the introduction. We have only one non-deterministic rule*

$$\rho_1 : \ \mathsf{sub}(z, y) \to x \Leftarrow \mathsf{add}(x, y) \to z.$$

Now we transform ρ_1 by Nar *to obtain the following rewrite rules:*

$$\rho_2 : \ \mathsf{sub}(x, \mathsf{0}) \to x,$$
$$\rho_3 : \ \mathsf{sub}(\mathsf{s}(z'), \mathsf{s}(y')) \to x \Leftarrow \mathsf{add}(x, y') \to z'.$$

Because this is very complex, we intuitively explain this transformation by decomposing it into several steps.

The first step is case splitting of the conditional part of ρ_1. We have two cases 0 and $\mathsf{s}(y')$ for the second argument y of add*, which is induced by the rules defining* add*. We obtain the following two rules:*

$$\rho_{2'} : \ \mathsf{sub}(z, 0) \to x \Leftarrow \mathsf{add}(x, 0) \to z,$$
$$\rho_{3'} : \ \mathsf{sub}(z, \mathsf{s}(y')) \to x \Leftarrow \mathsf{add}(x, \mathsf{s}(y')) \to z.$$

The second step takes reductions of the conditions. The term $\mathsf{add}(x, 0)$ *in the conditional part of* $\rho_{2'}$ *is reduced to* x *by* $\mathcal{R}_{\mathsf{add}}$*, hence* $\rho_{2'}$ *is transformed to* $\rho_{2''}$ *below. However* $\mathsf{add}(x, \mathsf{s}(y'))$ *is not reducible by* $\mathcal{R}_{\mathsf{add}}$ *because the conditional part of* $\rho_{3'}$ *is not reducible. When the rule* $\rho_{3'}$ *is applied to a term with a substitution* σ*, the term* $\mathsf{add}(x, \mathsf{s}(y'))\sigma$ *is reduced to* $z\sigma$*, where only the rule* $\mathsf{add}(x, \mathsf{s}(y')) \to \mathsf{s}(z') \Leftarrow \mathsf{add}(x, y') \to z'$ *is applicable. Here variables are renamed for ease of understanding. Thus* $\mathsf{add}(x, y')\sigma$ *must be reduced to a term* t *such that* $\mathsf{s}(t) = z\sigma$*. Summarizing this observation, the rule* $\rho_{3'}$ *is transformed to* $\rho_{3''}$*:*

$$\rho_{2''}: \ \mathsf{sub}(z, 0) \to x \Leftarrow x \to z,$$
$$\rho_{3''}: \ \mathsf{sub}(z, \mathsf{s}(y')) \to x \Leftarrow \mathsf{s}(z') \to z; \mathsf{add}(x, y') \to z'.$$

Note that although $\rho_{2''}$ *and* $\rho_{3''}$ *are not pure-constructor rules, these rules are used only for explaining the intuition behind the transformation rules and do not appear in the real transformations. The third step removes conditions that contain no defined symbols. Intuitively the condition* $x \to z$ *means that the values of* z *and* x *are equal. Thus* $\rho_{2''}$ *is simplified to* ρ_2 *by substituting* x *for* z *and by removing* $x \to x$*. By a similar reason,* $\rho_{3''}$ *is simplified to* ρ_3 *by substituting* $\mathsf{s}(z')$ *for* z *and by removing* $\mathsf{s}(z') \to \mathsf{s}(z')$*. The series of splitting, reductions, simplification are formalized as the transformation rule named* Nar*, which transforms* ρ_1 *into* ρ_2 *and* ρ_3 *in one step. We sometime describe this rule like* $\mathrm{Nar}_{\mathsf{add}}$ *with the function used in the reductions attached. If a rule contains an infeasible condition like* $\mathsf{s}(x) \to 0$ *then both sides are not unifiable and it fails the simplification by removing such conditions with no defined symbol in left-hand side. Such a rule, however, is not produced by* Nar *since we take the most general unifier also from right-hand sides.*

Since ρ_3 *is still non-deterministic, we will fold by replacing* $\mathsf{add}(z, y') \to z'$ *with* $\mathsf{sub}(z', y') \to x$ *according to the following rule* $\rho_{1'}$*, which is* ρ_1 *with variables (partially) renamed:*

$$\rho_{1'}: \ \mathsf{sub}(z', y') \to x \Leftarrow \mathsf{add}(x, y') \to z'.$$

By this folding, we obtain the following rule ρ_4*:*

$$\rho_4: \ \mathsf{sub}(\mathsf{s}(z'), \mathsf{s}(y')) \to x \Leftarrow \mathsf{sub}(z', y') \to x.$$

This is formalized as the transformation rule Fld_0*. The formula* $l' \to r' \Leftrightarrow \exists X_{c' \setminus l', r'}.c'$ *in the transformation rule* Fld_0 *are typically selected from rewrite*

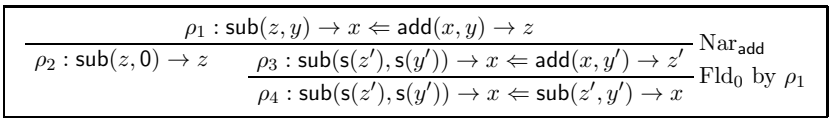

Figure 2: Transformation of sub in Example 1

rules $l' \to r' \Leftarrow c'$ *that already appeared in the transformation sequences. Here we can prove that* $\mathsf{sub}(z',y') \to x \Leftrightarrow \mathsf{add}(x,y') \to z'$ *is an inductive theorem of* $\{\rho_2,\rho_3\} \cup \mathcal{R}_{\mathsf{add}}$ *by using structural induction. This is also proved by using Lemma 5.4 shown later in this paper.*

Now, we have a successful transformation sequence

$$\mathcal{E}_{\mathsf{sub}} = \{\rho_1\} \Rrightarrow^{\mathrm{Nar}}_{\mathcal{R}_{\mathsf{add}}} \{\rho_2,\rho_3\} \Rrightarrow^{\mathrm{Fld}_0}_{\mathcal{R}_{\mathsf{add}}} \{\rho_2,\rho_4\} = \mathcal{R}_{\mathsf{sub}},$$

and stop the transformation because the rules are all deterministic. Note that the name of the rule is attached to each one-step transformation like $\Rrightarrow^{\mathrm{Nar}}_{\mathcal{R}}$. *This transformation sequence is represented a proof-tree-like notation as in Figure 2, which we use in the sequel.*

The side condition on variables in Fld_0 rule is necessary as shown by the next example.

Example 2. *Consider* $\mathcal{E}_{\mathsf{oneThird}}$ *and* $\mathcal{R}_{\mathsf{add}}$*:*

$$\mathcal{E}_{\mathsf{oneThird}} = \left\{ \begin{array}{l} \mathsf{oneThird}(z) \to x \Leftarrow \mathsf{add}(x,x) \to y; \mathsf{add}(x,y) \to z, \\ \mathsf{isTwoThird}(y,z) \to x \Leftarrow \mathsf{add}(x,x) \to y; \mathsf{add}(x,y) \to z \end{array} \right\}.$$

If we ignored the variable condition of Fld_0*, the latter rule could be folded by the former rule, producing* $\mathcal{E}'_{\mathsf{oneThird}}$*:*

$$\mathcal{E}'_{\mathsf{oneThird}} = \left\{ \begin{array}{l} \mathsf{oneThird}(z) \to x \Leftarrow \mathsf{add}(x,x) \to y; \mathsf{add}(x,y) \to z, \\ \mathsf{isTwoThird}(y,z) \to x \Leftarrow \mathsf{oneThird}(z) \to x \end{array} \right\}.$$

It is possible to reduce $\mathsf{isTwoThird}(\mathsf{s}(0),\mathsf{s}(\mathsf{s}(\mathsf{s}(0))))$ *to* $\mathsf{s}(0)$ *by* $\mathcal{E}'_{\mathsf{oneThird}} \cup \mathcal{R}_{\mathsf{add}}$*, but impossible by* $\mathcal{E}_{\mathsf{oneThird}} \cup \mathcal{R}_{\mathsf{add}}$*.*

The third example explains a typical choice of formulas used in Fld_0 transformation.

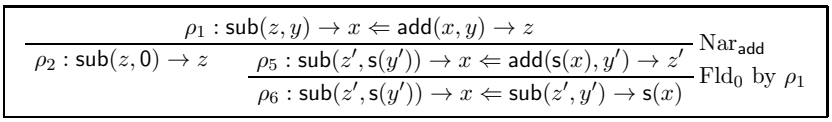

Figure 3: Transformation of sub with $\mathcal{R}'_{\mathsf{add}}$

Example 3. *Consider* $\mathcal{E}_{\mathsf{sub}}$ *and* $\mathcal{R}'_{\mathsf{add}}$*:*

$$\mathcal{R}'_{\mathsf{add}} = \left\{ \begin{array}{l} \mathsf{add}(x,0) \to x, \\ \mathsf{add}(x,\mathsf{s}(y)) \to z \Leftarrow \mathsf{add}(\mathsf{s}(x),y) \to z \end{array} \right\}.$$

Note that the latter rule of $\mathcal{R}'_{\mathsf{add}}$ *is modified a bit from* $\mathcal{R}_{\mathsf{add}}$*. The transformation is shown in Figure 3. We obtain a deterministic CTRS* $\mathcal{E}'_{\mathsf{sub}}$ *slightly different from Example 1. In the folding step,* $\mathsf{add}(\mathsf{s}(x),y') \to z'$ *is replaced with* $\mathsf{sub}(z',y') \to \mathsf{s}(x)$ *according to the following inductive theorem, which is a constructor-instance of* ρ_1*:*

$$\mathsf{sub}(z',y') \to \mathsf{s}(x) \Leftrightarrow \mathsf{add}(\mathsf{s}(x),y') \to z'.$$

Note that constructor substitutions preserve inductive theorems. as shown by Proposition 5.3 in Section 5.

The fourth example explains Def rule.

Example 4. *Consider the following CTRSs* $\mathcal{E}_{\mathsf{div}}$ *and* $\mathcal{R}_{\mathsf{mul}}$*, where* div *is the division on natural numbers:*

$$\mathcal{E}_{\mathsf{div}} = \left\{ \; \mathsf{div}(z,y) \to x \Leftarrow \mathsf{mul}(x,y) \to z; >_{\mathbb{N}}(y,0) \to \mathsf{tt} \; \right\},$$
$$\mathcal{R}_{\mathsf{mul}} = \mathcal{R}_{\mathsf{add}} \cup \left\{ \begin{array}{l} \mathsf{mul}(x,0) \to 0, \\ \mathsf{mul}(0,y) \to 0, \\ \mathsf{mul}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{s}(z) \Leftarrow \mathsf{mul}(x,\mathsf{s}(y)) \to w; \mathsf{add}(w,y) \to z, \\ >_{\mathbb{N}}(0,y) \to \mathsf{ff}, \\ >_{\mathbb{N}}(\mathsf{s}(x),0) \to \mathsf{tt}, \\ >_{\mathbb{N}}(\mathsf{s}(x),\mathsf{s}(y)) \to z \Leftarrow >_{\mathbb{N}}(x,y) \to z \end{array} \right\}.$$

From $\mathcal{E}_{\mathsf{div}}$ *and* $\mathcal{R}_{\mathsf{mul}}$*, we obtain a successful transformation* $\mathcal{E}_{\mathsf{div}} \overset{*}{\Rrightarrow}_{\mathcal{R}_{\mathsf{mul}}} \mathcal{R}_{\mathsf{div}}$ *as shown in Figure 4. The rewrite rule* ρ_{11} *in Figure 4 is still non-deterministic*

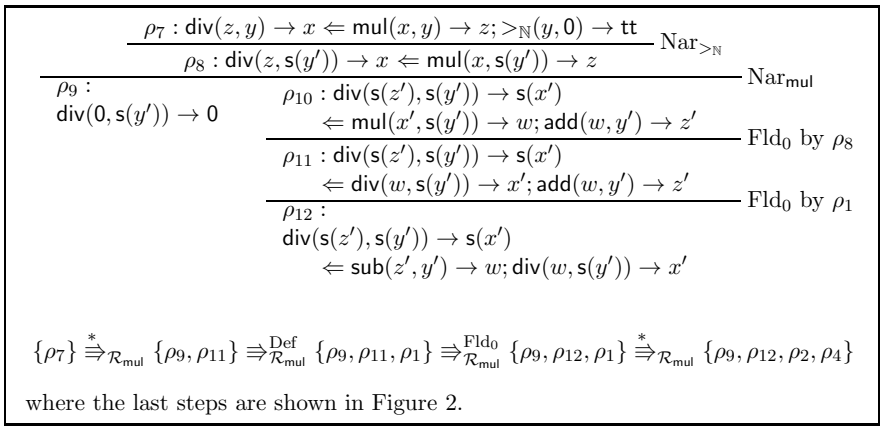

Figure 4: Transformation of div in Example 4

since the occurrence of w is unknown. If we can replace $\mathsf{add}(w,y') \to z'$ in ρ_{11} by an atom $u \to w$ such that the term u contains no occurrence of w, the rule becomes deterministic. Thus we define the rule $\mathsf{sub}(z,y) \to x \Leftarrow \mathsf{add}(x,y) \to z$ and replace $\mathsf{add}(w,y') \to z'$ in ρ_{11} by $\mathsf{sub}(z',y') \to w$. Typically we introduce rules like this example to make left-hand side of a condition deterministic by combination with Fld_0.

As a result of the transformation, we obtained a deterministic CTRS $\mathcal{R}_{\mathsf{div}} \cup \mathcal{R}_{\mathsf{mul}}$:

$$\mathcal{R}_{\mathsf{div}} = \mathcal{R}_{\mathsf{sub}} \cup \left\{ \begin{array}{l} \mathsf{div}(0,\mathsf{s}(y)) \to 0, \\ \mathsf{div}(\mathsf{s}(z),\mathsf{s}(y)) \to \mathsf{s}(x) \Leftarrow \mathsf{sub}(z,y) \to w; \mathsf{div}(w,\mathsf{s}(y)) \to x \end{array} \right\}.$$

Remark that the transformation rule Fld_0 does not preserve constructor-based reduction as shown in the following example.

Example 5. *Consider $\rho_1 : \mathsf{sub}(x,y) \to z \Leftarrow \mathsf{add}(z,y) \to x$ in Example 1. Since $\mathsf{add}(z,y) \to x \Leftrightarrow \mathsf{sub}(x,y) \to z$ is an inductive theorem of $\mathcal{E}_{\mathsf{sub}} \cup \mathcal{R}_{\mathsf{add}}$, we obtain the following rule by* Fld_0*:*

$$\mathsf{sub}(x,y) \to z \Leftarrow \mathsf{sub}(x,y) \to z.$$

However, it is not possible to reduce $\mathsf{sub}(0,0)$ to 0 by the resulting CTRS.

Additional conditions to avoid this problem are discussed in Section 4.3.

The following important syntactic property holds obviously from the construction of the transformation rules.

Proposition 3.2. *Let $\mathcal{R}$ and $\mathcal{E}_1$ be oriented and pure-constructor CTRSs such that $\mathcal{D}_{\mathcal{E}_1} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$. Then $\mathcal{E}_2$ obtained from $\mathcal{E}_1$ by one transformation step is also an oriented and pure-constructor system satisfying $\mathcal{D}_{\mathcal{E}_2} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$ and $\mathcal{D}_{\mathcal{E}_1} \subseteq \mathcal{D}_{\mathcal{E}_2}$.*

4. Soundness

In this section, we first redefine the constructor-based reduction of a CTRS. We then show that the transformation system is simulation sound and also simulation complete, that is, the system is correct.

4.1. Redefinition of the constructor-based reduction

We redefine the constructor-based reduction. The *k-level constructor-based reduction* $\xrightarrow[\mathsf{c}]{}{}^{(k)}_{\mathcal{R}}$ of $\mathcal{R}$ is inductively defined as follows:

- $\xrightarrow[\mathsf{c}]{}{}^{(0)}_{\mathcal{R}} = \emptyset$, and
- $\xrightarrow[\mathsf{c}]{}{}^{(j)}_{\mathcal{R}} = \xrightarrow[\mathsf{c}]{}{}^{(j-1)}_{\mathcal{R}} \cup$
  $\Big\{(l\sigma, r\sigma) \mid l \to r \Leftarrow c \in \mathcal{R},$
  σ is a constructor substitution, $\langle \xrightarrow[\mathsf{c}]{}{}^{(j-1)}_{\mathcal{R}}, \sigma\rangle \models c\Big\}$ for $j > 0$.

The *constructor-based reduction* $\xrightarrow[\mathsf{c}]{}_{\mathcal{R}}$ is the monotonic closure of $\bigcup_{k\geq 0} \xrightarrow[\mathsf{c}]{}{}^{(k)}_{\mathcal{R}}$. This definition is simpler than the ordinary one and makes proofs of this paper easier and clearer. These definitions are equivalent for pure-constructor CTRSs as shown by the following proposition.

Proposition 4.1. *Let $\mathcal{R}$ be a pure-constructor CTRS. Then the redefined simpler definition and the ordinary definition of constructor reductions are equivalent.*

Proof . *In this proof, we distinguish the ordinary constructor-based reduction from the redefined one by attaching a prime to the former, e.g., $\xrightarrow[\mathsf{c}]{}{}'_{\mathcal{R}}$ for the ordinary one and $\xrightarrow[\mathsf{c}]{}_{\mathcal{R}}$ for the redefined one.*

Consider the more interesting direction that $\xrightarrow[\mathsf{c}]{}{}'_{\mathcal{R}} \subseteq \xrightarrow[\mathsf{c}]{}_{\mathcal{R}}$. First we show by induction on k the claim that $u \xrightarrow[\mathsf{c}]{}{}^{*}{}'^{(k)}_{\mathcal{R}} v$ implies $u \xrightarrow[\mathsf{c}]{}{}^{(k)}_{\mathcal{R}} v$ for any pattern u and any constructor term v.

Since $u \neq v$ we have at least one step reduction in $u \overset{*}{\underset{\mathsf{c}}{\rightarrow}}{}'^{(k)}_{\mathcal{R}} v$. Thus the case $k = 0$ is trivial because $\underset{\mathsf{c}}{\rightarrow}{}'^{(0)}_{\mathcal{R}} = \emptyset$. We consider the case that $k > 0$. Let $u \overset{*}{\underset{\mathsf{c}}{\rightarrow}}{}'^{(k)}_{\mathcal{R}} v$. Since u is a pattern, there exists a rule $l \to r \Leftarrow u_1 \to v_1; \cdots; u_n \to v_n$ in $\mathcal{R}$ and a constructor substitution σ such that $u = l\sigma$, $r\sigma \overset{*}{\underset{\mathsf{c}}{\rightarrow}}{}'^{(k)}_{\mathcal{R}} v$ and $u_i\sigma \overset{*}{\underset{\mathsf{c}}{\rightarrow}}{}'^{(k-1)}_{\mathcal{R}} v_i\sigma$ for all i. Since $u_i\sigma$ is a pattern and $v_i\sigma$ is a constructor term, we obtain $u_i\sigma \underset{\mathsf{c}}{\rightarrow}{}^{(k-1)}_{\mathcal{R}} v_i\sigma$ by the induction hypothesis. Thus, $u = l\sigma \underset{\mathsf{c}}{\rightarrow}{}^{(k)}_{\mathcal{R}} r\sigma \overset{*}{\underset{\mathsf{c}}{\rightarrow}}{}'^{(k)}_{\mathcal{R}} v$. Here $r\sigma$ is a constructor term. Therefore, $u = l\sigma \underset{\mathsf{c}}{\rightarrow}{}^{(k)}_{\mathcal{R}} r\sigma = v$.

Now we show that $\underset{\mathsf{c}}{\rightarrow}{}'_{\mathcal{R}} \subseteq \underset{\mathsf{c}}{\rightarrow}_{\mathcal{R}}$. Assume $s \underset{\mathsf{c}}{\rightarrow}{}'_{\mathcal{R}} t$. Then $s \underset{\mathsf{c}}{\rightarrow}{}'^{(k)}_{\mathcal{R}} t$ for some k and hence there exists a rule $l \to r \Leftarrow c$ in $\mathcal{R}$ and a constructor substitution σ such that $s = C[l\sigma]$, $t = C[r\sigma]$ and $u_i\sigma \overset{*}{\underset{\mathsf{c}}{\rightarrow}}{}'^{(k-1)}_{\mathcal{R}} v_i\sigma$ for all i. Since $u_i\sigma$ is a pattern and $v_i\sigma$ is a constructor term, we obtain $u_i\sigma \underset{\mathsf{c}}{\rightarrow}{}^{(k-1)}_{\mathcal{R}} v_i\sigma$ by the claim. Thus we have $l\sigma \underset{\mathsf{c}}{\rightarrow}{}^{(k)}_{\mathcal{R}} r\sigma$. This means that $s = C[l\sigma]$ is reduced to $t = C[r\sigma]$ by the monotonic closure of $\bigcup_{k\geq 0} \underset{\mathsf{c}}{\rightarrow}{}^{(k)}_{\mathcal{R}}$, which concludes the proof. □

The following proposition guarantees that it is enough to consider patterns in order to prove the simulation soundness and simulation completeness.

Proposition 4.2. *Let $\mathcal{R}$ and $\mathcal{E}_1$ be oriented and pure-constructor CTRSs such that $\mathcal{R}$ is deterministic satisfying $\mathcal{D}_{\mathcal{E}_1} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$. Suppose $s \underset{\mathsf{c}}{\rightarrow}_{\mathcal{E}_1 \cup \mathcal{R}} t$ implies $s \underset{\mathsf{c}}{\rightarrow}_{\mathcal{E}_2 \cup \mathcal{R}} t$ for any pattern $s \in \mathcal{T}(\mathcal{D}_{\mathcal{E}_1} \cup \mathcal{C})$ and $t \in \mathcal{T}(\mathcal{C})$. Then $s' \overset{*}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}_1 \cup \mathcal{R}} t'$ implies $s' \overset{*}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}_2 \cup \mathcal{R}} t'$ for any terms $s', t' \in \mathcal{T}(\mathcal{D}_{\mathcal{E}_1} \cup \mathcal{D}_{\mathcal{R}} \cup \mathcal{C})$.*

Proof . *We prove that $s' \overset{n}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}_1 \cup \mathcal{R}} t'$ implies $s' \overset{*}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}_2 \cup \mathcal{R}} t'$ by induction on the number n of the former reduction steps. Since the case $n = 0$ is trivial, we consider the case $n > 0$. Then the former reduction is represented as $s' = C[l\sigma] \underset{\mathsf{c}}{\rightarrow}_{\mathcal{E}_1 \cup \mathcal{R}} C[r\sigma] \overset{n-1}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}_1 \cup \mathcal{R}} t'$. Now $s' = C[l\sigma] \underset{\mathsf{c}}{\rightarrow}_{\mathcal{E}_2 \cup \mathcal{R}} C[r\sigma]$ follows from $\mathcal{D}_{\mathcal{E}_1} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$ if the rule used in the first step is in $\mathcal{R}$, and from the assumption and the fact that $l\sigma$ is a pattern, otherwise. We also have $C[r\sigma] \overset{*}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}_2 \cup \mathcal{R}} t'$ from the induction hypothesis, which concludes the proof.* □

4.2. Simulation soundness

We show the *simulation soundness*, that is,

$$\mathcal{E} \overset{*}{\Rightarrow}_{\mathcal{R}} \mathcal{E}' \text{ and } s \overset{*}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}' \cup \mathcal{R}} t \text{ imply } s \overset{*}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E} \cup \mathcal{R}} t \text{ for any } s, t \in \mathcal{T}(\mathcal{D}_{\mathcal{E}} \cup \mathcal{D}_{\mathcal{R}} \cup \mathcal{C}).$$

It suffices to show the simulation soundness for one step reduction by Proposition 4.2. Instead, we have to be careful with newly defined symbols introduced by the rule Def.

Lemma 4.3. *Let $\mathcal{E}_1 \Rrightarrow_{\mathcal{R}} \mathcal{E}_2$. If $s \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}} t$, then $s \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}} t$ for every pattern $s \in \mathcal{T}(\mathcal{D}_{\mathcal{E}_1} \cup \mathcal{C})$ and $t \in \mathcal{T}(\mathcal{C})$.*

Proof . *We show that $s \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_2 \cup \mathcal{R}} t$ implies $s \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}} t$ by induction on k.*

Consider the case that the reduction $s \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_2 \cup \mathcal{R}} t$ is by a rule $l \to r \Leftarrow c$ in $\mathcal{E}_1 \cap \mathcal{E}_2$. Note that this rule does not contain any newly introduced symbol in $\mathcal{E}_2$. Thus $s = l\theta \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_2 \cup \mathcal{R}} r\theta = t$ and $\langle \xrightarrow[\mathsf{c}]{}^{(k-1)}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c$ for some ground constructor substitution θ. Since $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c$ by the induction hypothesis, we have $s = l\theta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}} r\theta = t$.

Otherwise we consider separately each transformation rule in the transformation.

- Nar*: Let $\mathcal{E}_1 = \mathcal{E} \cup \{l \to r \Leftarrow c; u \to v\}$ and $\mathcal{E}_2 = \mathcal{E} \cup \{l\sigma \to r\sigma \Leftarrow c\sigma; c'\sigma \mid l' \to r' \Leftarrow c' \in \mathcal{E}_1 \cup \mathcal{R}, \sigma = \mathrm{mgu}(\{u \sim l', v \sim r'\})\}$. We can assume that $s = l\sigma\theta$, $t = r\sigma\theta$ and $\langle \xrightarrow[\mathsf{c}]{}^{(k-1)}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c\sigma; c'\sigma$ for a rule $l' \to r' \Leftarrow c' \in \mathcal{E}_1 \cup \mathcal{R}$ and a ground constructor substitution θ. By the induction hypothesis, $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c\sigma; c'\sigma$. Thus $l'\sigma\theta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}} r'\sigma\theta$. Since $\sigma = \mathrm{mgu}(\{u \sim l', v \sim r'\})$, we have $u\sigma\theta = l'\sigma\theta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}} r'\sigma\theta = v\sigma\theta$. Therefore $s = l\sigma\theta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}} r\sigma\theta = t$.*

- Fld_0*: Let $\mathcal{E}_1 = \mathcal{E} \cup \{l \to r \Leftarrow c; c'\}$, $\mathcal{E}_2 = \mathcal{E} \cup \{l \to r \Leftarrow c; l' \to r'\}$ and $s \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_2 \cup \mathcal{R}} t$ by the rule $l \to r \Leftarrow c; l' \to r'$. Thus $s = l\theta$, $t = r\theta$, $\langle \xrightarrow[\mathsf{c}]{}^{(k-1)}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c$ and $l'\theta \xrightarrow[\mathsf{c}]{}^{(k-1)}_{\mathcal{E}_2 \cup \mathcal{R}} r'\theta$ for a ground constructor substitution θ. Here we can assume that $\mathcal{D}om(\theta) = \mathcal{V}ar(l, r, c, l', r')$ without loss of generality. $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c$ and $l'\theta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}} r'\theta$ follow from the induction hypothesis. Since $l' \to r' \Rightarrow \exists X_{c' \setminus l', r'}.c'$ is an inductive theorem of $\mathcal{E}_1 \cup \mathcal{R}$, we have $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta' \rangle \models c'$ for an extension θ' of $\theta|_{\mathcal{V}ar(l', r')}$ where $\mathcal{D}om(\theta') = \mathcal{V}ar(l', r', c')$. We have an extension δ of θ and θ' because $\mathcal{V}ar(l, r, c, l', r') \cap \mathcal{V}ar(l', r', c') \subseteq \mathcal{V}ar(l', r')$ from the variable condition and also $x\theta' = x\theta$ for $x \in \mathcal{V}ar(l', r')$. Therefore $l\theta = l\delta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_1 \cup \mathcal{R}} r\delta = r\theta$.*

- Def*: Since s does not contain the newly introduced symbol $\mathrm{root}(l)$, the term s is reduced by a rule in $\mathcal{E}$.* □

Theorem 4.4 (Simulation Soundness). *Let $\mathcal{R}$ and $\mathcal{E}$ be oriented and pure-constructor CTRSs. If $\mathcal{R}$ is deterministic satisfying $\mathcal{D}_\mathcal{E} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$, then the transformations of $\mathcal{DT}_0$ are simulation sound, i.e. if $\mathcal{E} \overset{*}{\Rrightarrow}_\mathcal{R} \mathcal{E}'$, $s \overset{*}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}'\cup\mathcal{R}} t$ implies $s \overset{*}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}\cup\mathcal{R}} t$ for any $s, t \in \mathcal{T}(\mathcal{D}_\mathcal{E} \cup \mathcal{D}_\mathcal{R} \cup \mathcal{C})$.*

Proof . *The property $\mathcal{D}_\mathcal{E} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$ is preserved by Proposition 3.2. We prove by induction on the number n of the steps of the transformation that $s \overset{}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}'\cup\mathcal{R}} t$ implies $s \overset{}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}\cup\mathcal{R}} t$. Thanks to Proposition 4.2 we can assume that s is a pattern in $\mathcal{T}(\mathcal{D}_\mathcal{E} \cup \mathcal{C})$.*

Since the case $n = 0$ is trivial, consider the case $n > 0$. Let $\mathcal{E} \overset{n-1}{\Rrightarrow}_\mathcal{R} \mathcal{E}'' \Rrightarrow_\mathcal{R} \mathcal{E}'$. Let $s \overset{}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}'\cup\mathcal{R}} t$. Since $\mathcal{D}_\mathcal{E} \subseteq \mathcal{D}_{\mathcal{E}''}$, we have $s \overset{}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}''\cup\mathcal{R}} t$ by Lemma 4.3. Thus $s \overset{}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}\cup\mathcal{R}} t$ follows from the induction hypothesis. □

4.3. Simulation completeness

We show the *simulation completeness*, that is,

> if $\mathcal{E} \overset{*}{\Rrightarrow}_\mathcal{R} \mathcal{E}'$, $s \overset{*}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}\cup\mathcal{R}} t$ implies $s \overset{*}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{E}'\cup\mathcal{R}} t$ for any $s, t \in \mathcal{T}(\mathcal{D}_\mathcal{E} \cup \mathcal{D}_\mathcal{R} \cup \mathcal{C})$.

Since Fld_0 is not simulation complete as shown in Example 5, we must be careful in applying Fld_0. Before giving additional side conditions of Fld_0, we define a necessary definition.

Definition 4.5 (Level condition). *Let $l \to r \Leftarrow c$ be a formula that satisfies the following for every $k \geq 0$ and ground constructor substitution σ:*

$$\mathcal{D}om(\sigma) = \mathcal{V}ar(l, r, c) \text{ and } \langle \overset{}{\underset{\mathsf{c}}{\rightarrow}}_\mathcal{R}^{(k)}, \sigma \rangle \models c \text{ imply } \langle \overset{}{\underset{\mathsf{c}}{\rightarrow}}_\mathcal{R}^{(k+1)}, \sigma \rangle \models l \to r.$$

Then $l \to r \Leftarrow c$ is said to be an inductive theorem of $\mathcal{R}$ satisfying the level condition.

A rule $l \to r \Leftarrow c$ in $\mathcal{R}$ is clearly an inductive theorem of $\mathcal{R}$ satisfying the level condition.

We define three additional side conditions of Fld_0 as follows;

(C1) $l' \to r' \Leftarrow c'$ is a constructor-instance of a rule in $\mathcal{E} \cup \mathcal{R}$.

(C2) $l' \to r' \Leftarrow c'$ is an inductive theorem of $\mathcal{E}_1 \cup \mathcal{R}$ satisfying the level condition, and there exists a well-founded order $\sqsupset$ such that

$(l\sigma, r\sigma) \sqsupset (l'\sigma, r'\sigma)$ for any ground constructor substitution σ such that $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}_1 \cup \mathcal{R}}, \sigma \rangle \models c; c'$.

(C3) There exists a well-founded order $\sqsupset$ such that

- for any $l'' \to r'' \Leftarrow u_1 \to v_1; \cdots; u_n \to v_n$ in $\mathcal{E}_1 \cup \mathcal{R}$
  $(l''\sigma, r''\sigma) \sqsupseteq (u_i\sigma, v_i\sigma)$ for any ground constructor substitution σ such that $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}_1 \cup \mathcal{R}}, \sigma \rangle \models u_1 \to v_1; \cdots; u_n \to v_n$, and
- $(l\sigma, r\sigma) \sqsupset (l'\sigma, r'\sigma)$ for any ground constructor substitution σ such that $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}_1 \cup \mathcal{R}}, \sigma \rangle \models c; c'$.

It is generally undecidable whether a given formula is an inductive theorem (with level condition). In Section 5.1, we discuss syntactic conditions for the inductive theorem used in the side-condition of Fld_0. We also give a syntactic condition for the level condition of (C2) in Section 5.2. We use the condition (C3) to prove a completeness result in Section 6.

We use Fld_i for Fld_0 with (Ci) attached for each i. We use $\mathcal{DT}_{123}$ for denoting the system obtained from $\mathcal{DT}_0$ by replacing Fld_0 with three transformation rules Fld_1, Fld_2 and Fld_3.

The condition (C1) is a simple restriction that takes a constructor-instance of a rule in $\mathcal{E}$ (not $\mathcal{E}_1$) for folding. This is simple but valuable, especially when replacing a conditional part by a function introduced by Def.

Unfortunately Fld_1 is insufficient for the transformation in Example 1, because $\mathsf{sub}(x, y) \to z$ already disappeared in an earlier stage in the transformation sequence. Now consider the condition (C2). By comparing the applications of Fld_0 in Examples 1 and 5, we observe that the resulting rewrite rules constitute a recursive definition of sub. The former application is successful because l is smaller than l' in some sense, which makes sense as a recursive definition. The folding application in Example 5 is not successful because l and l' are equal in some sense, and hence the rule $l \to r \Leftarrow l' \to r'$ is nonsense as a recursive definition. This is why the order constraint is necessary.

In the following example, the order for (C2) is designed by the recursive path order [12].

Example 6. *Consider the transformation sequence in Example 1:*

$$\mathcal{E}_{\mathsf{sub}} = \{\rho_1\} \Rrightarrow^{\mathrm{Nar}}_{\mathcal{R}_{\mathsf{add}}} \{\rho_2, \rho_3\} \Rrightarrow^{\mathrm{Fld}_0}_{\mathcal{R}_{\mathsf{add}}} \{\rho_2, \rho_4\} = \mathcal{R}_{\mathsf{sub}}.$$

The second step $\Rrightarrow^{\mathrm{Fld}_0}_{\mathcal{R}_{\mathsf{add}}}$ *can be replaced by* Fld_2*. The rule* ρ_3 *is folded by a constructor instance* $\mathsf{sub}(z', y') \to x \Leftarrow \mathsf{add}(x, y') \to z$ *of* ρ_1 *and an order* $\sqsupset$ *defined as*

$$(s, t) \sqsupset (s', t') \iff s >_{rpo} s',$$

where $>_{rpo}$ *is the recursive path order [12]. Then* $\sqsupset$ *is well-founded and* $(\mathsf{sub}(\mathsf{s}(z'), \mathsf{s}(y')), x) \sqsupset (\mathsf{sub}(z', y'), x)$*. Since* $\sqsupset$ *is stable under substitution,* $(\mathsf{sub}(\mathsf{s}(z'), \mathsf{s}(y'))\sigma, x\sigma) \sqsupset (\mathsf{sub}(z', y')\sigma, x\sigma)$ *holds for every ground constructor substitution* σ*. We also show that the level condition of* $\mathsf{sub}(z', y') \to x \Leftarrow \mathsf{add}(x, y') \to z$ *holds. This is guaranteed by Lemma 5.5, as discussed in Section 5.*

In general, it is enough to find a well-founded order that is stable under substitution such that $(l, r) \succ (l', r')$, and the monotonicity property is not necessary. This enables us to combine the argument filtering technique [13] and simplification orders [12] like recursive path order in order to design a suitable order.

In the condition (C2), the constraint imposed on the order is strongly localized and hence we have to show only $(l\sigma, r\sigma) \sqsupset (l'\sigma, r'\sigma)$. The level condition for inductive theorems in (C2) is necessary as shown by the following example.

Example 7. *Consider* $\mathcal{E}_{\mathsf{even}}$ *and* $\mathcal{R}_{\mathsf{double}}$*:*

$$\mathcal{E}_{\mathsf{even}} = \left\{ \begin{array}{l} \rho_{13} : \ \mathsf{even}(x) \to \mathsf{tt} \Leftarrow \mathsf{double}(y) \to x, \\ \rho_{14} : \ \mathsf{odd}(x) \to \mathsf{ff} \Leftarrow \mathsf{double}(y) \to x \end{array} \right\},$$

$$\mathcal{R}_{\mathsf{double}} = \left\{ \begin{array}{l} \mathsf{double}(\mathsf{0}) \to \mathsf{0}, \\ \mathsf{double}(\mathsf{s}(y)) \to \mathsf{s}(\mathsf{s}(z)) \Leftarrow \mathsf{double}(y) \to z \end{array} \right\}.$$

We have the following transformation sequence:

$$\mathcal{E}_{\mathsf{even}} = \{\rho_{13}, \rho_{14}\} \Rrightarrow^{\mathrm{Fld}_2}_{\mathcal{R}_{\mathsf{double}}} \{\rho_{15}, \rho_{14}\} \Rrightarrow^{\mathrm{Fld}_0}_{\mathcal{R}_{\mathsf{double}}} \{\rho_{15}, \rho_{16}\},$$

where

$$\begin{array}{l} \rho_{15} : \ \mathsf{even}(x) \to \mathsf{tt} \Leftarrow \mathsf{odd}(x) \to \mathsf{ff}, \\ \rho_{16} : \ \mathsf{odd}(x) \to \mathsf{ff} \Leftarrow \mathsf{even}(x) \to \mathsf{tt}. \end{array}$$

The second folding step satisfies the order condition in (C2), but not the level condition in (C2). It is possible to reduce $\mathsf{odd}(\mathsf{0})$ *to* ff *by* $\{\rho_{15}, \rho_{14}\} \cup \mathcal{R}_{\mathsf{double}}$*, but impossible by* $\{\rho_{15}, \rho_{16}\} \cup \mathcal{R}_{\mathsf{double}}$*.*

The conditions (C1) and (C2) have a benefit that they do not require a kind of termination property of CTRSs. However, they are not sufficient for the completeness result that we discuss in Section 6. In this sense the condition (C3) is more powerful and we use this condition for the completeness proof of transformations. Instead it requires to design a well-founded order such that every rewrite rule in $\mathcal{E} \cup \mathcal{R}$ must be ordered. Note that it is not necessary to use the same order $\sqsupset$ in each application of Fld_2 or Fld_3 during the entire transformation sequence. This greatly eases the design of the orders especially for Fld_2.

In the rest of subsection, we prove the simulation completeness of $\mathcal{DT}_{123}$. The simulation completeness of Fld_i $(1 \leq i \leq 3)$ is shown by the following lemmas, where different induction measures are used.

Lemma 4.6. *The rule* Fld_1 *is simulation complete.*

Proof . *Let* $\mathcal{E}_1 = \mathcal{E} \cup \{l \to r \Leftarrow c; c'\}$ *and* $\mathcal{E}_2 = \mathcal{E} \cup \{l \to r \Leftarrow c; l' \to r'\}$ *by* Fld_1*, where* $l' \to r' \Leftarrow c'$ *is a constructor-instance of a rule in* $\mathcal{E}$*. We show that for a pattern* $s \in \mathcal{T}(\mathcal{D}_{\mathcal{E}_1} \cup \mathcal{C})$ *and ground constructor term* t*,* $s \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ *implies* $s \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}} t$ *by induction on* k*.*

Consider the case that the reduction $s \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ *is by a rule* $l'' \to r'' \Leftarrow c''$ *in* $\mathcal{E}_1 \cap \mathcal{E}_2$*;* $s = l''\theta \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} r''\theta = t$ *and* $\langle \xrightarrow[\mathsf{c}]{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c''$ *for a ground constructor substitution* θ*. Since* $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c''$ *by the induction hypothesis, we have* $s = l''\theta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}} r''\theta = t$*.*

Consider the case that the reduction $s \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ *is by the rule* $l \to r \Leftarrow c; c'$*, that is,* $s = l\theta$*,* $t = r\theta$ *and* $\langle \xrightarrow[\mathsf{c}]{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c; c'$ *for a ground constructor substitution* θ*, where we can assume that* $\mathcal{D}om(\theta) = \mathcal{V}ar(l, r, c, c')$ *without loss of generality. By the induction hypothesis,* $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c; c'$*.*

Since $l' \to r' \Leftarrow c'$ *is a constructor-instance of a rule in* $\mathcal{E} \subseteq \mathcal{E}_2$*, let the rule be* $l'' \to r'' \Leftarrow c''$*. Then there exists a constructor substitution* γ *such that* $l''\gamma = l'$*,* $r''\gamma = r'$ *and* $c''\gamma = c'$*. Since* $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c''\gamma$*, we have* $l'\theta = l''\gamma\theta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}} r''\gamma\theta = r'\theta$*. Therefore* $s = l\theta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}} r\theta = t$ *follows.* □

Lemma 4.7. *The transformation rule* Fld_2 *is simulation complete.*

Proof . *Let* $\mathcal{E}_2 = \mathcal{E} \cup \{l \to r \Leftarrow c; l' \to r'\}$ *be obtained from* $\mathcal{E}_1 = \mathcal{E} \cup \{l \to r \Leftarrow c; c'\}$ *by* Fld_2*. We show for a pattern* $s \in \mathcal{T}(\mathcal{D}_{\mathcal{E}_1} \cup \mathcal{C})$ *and ground constructor term* t *that* $s \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ *implies* $s \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}} t$ *by (well-founded) induction on the* $(k, (s, t))$ *ordered by the lexicographic product* $>_{\mathbb{N}} \times_{lex} \sqsupset$*.*

Consider the case that the reduction $s \xrightarrow{c}{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ *is by a rule* $l'' \to r'' \Leftarrow c''$ *in* $\mathcal{E}$*, that is,* $s = l''\theta \xrightarrow{c}{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} r''\theta = t$ *and* $\langle \xrightarrow{c}{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c''$ *for a ground constructor substitution* θ*. Since* $\langle \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c''$ *by the induction hypothesis, we have* $s = l''\theta \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}} r''\theta = t$*.*

Consider the case that the reduction $s \xrightarrow{c}{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ *is by the rule* $l \to r \Leftarrow c; c'$*, that is,* $s = l\theta \xrightarrow{c}{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} r\theta = t$ *and* $\langle \xrightarrow{c}{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c; c'$ *for a ground constructor substitution* θ*, where we can assume that* $\mathcal{D}om(\theta) = \mathcal{V}ar(l, r, c, c')$ *without loss of generality.*

Since $l' \to r' \Leftarrow c'$ *is an inductive theorem of* $\mathcal{E}_1 \cup \mathcal{R}$ *with level condition, we have* $l'\sigma \xrightarrow{c}{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} r'\sigma$ *for some extension* σ *of* θ*, where* $\mathcal{D}om(\sigma) \supseteq \mathcal{V}ar(l', r', c')$*. Here* $(s, t) = (l\theta, r\theta) = (l\sigma, r\sigma) \sqsupset (l'\sigma, r'\sigma)$ *from (C2). Thus we have* $l'\sigma \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}} r'\sigma$ *by the induction hypothesis. Since* $\langle \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c$ *also holds by the induction hypothesis, it follows that* $s = l\theta = l\sigma \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}} r\sigma = r\theta = t$*.* □

Lemma 4.8. *The rule* Fld_3 *is simulation complete.*

Proof . *Let* $\mathcal{E}_2 = \mathcal{E} \cup \{l \to r \Leftarrow c; l' \to r'\}$ *be obtained from* $\mathcal{E}_1 = \mathcal{E} \cup \{l \to r \Leftarrow c; c'\}$ *by* Fld_3*. We show for a pattern* $s \in \mathcal{T}(\mathcal{D}_{\mathcal{E}_1} \cup \mathcal{C})$ *and constructor term* t *that* $s \xrightarrow{c}{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ *implies* $s \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}} t$ *by (well-founded) induction on the* $((s, t), k)$ *ordered by the lexicographic product* $\sqsupset \times_{lex} >_{\mathbb{N}}$*.*

Consider the case that the reduction $s \xrightarrow{c}{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ *is by a rule* $l'' \to r'' \Leftarrow c''$ *in* $\mathcal{E}$*, that is,* $s = l''\theta \xrightarrow{c}{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} r''\theta = t$ *and* $\langle \xrightarrow{c}{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c''$ *for a ground constructor substitution* θ*. From the first condition of (C3), for any condition* $u \to v$ *in* c'' *we have* $(s, t) = (l''\theta, r''\theta) \sqsupseteq (u\theta, v\theta)$*. It follows from the induction hypothesis that* $\langle \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c''$*, and hence* $s = l''\theta \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}} r''\theta = t$*.*

Consider the case that the reduction $s \xrightarrow{c}{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ *is by the rule* $l \to r \Leftarrow c; c'$*, that is,* $s = l\theta \xrightarrow{c}{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} r\theta = t$ *and* $\langle \xrightarrow{c}{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c; c'$ *where we can assume that* $\mathcal{D}om(\theta) = \mathcal{V}ar(l, r, c, c')$ *without loss of generality.*

Since $l' \to r' \Leftarrow c'$ *is an inductive theorem of* $\mathcal{E}_1 \cup \mathcal{R}$*, we have* $l'\sigma \xrightarrow{c}_{\mathcal{E}_1 \cup \mathcal{R}} r'\sigma$ *for some extension* σ *of* θ*, where* $\mathcal{D}om(\sigma) \supseteq \mathcal{V}ar(l', r', c')$*. Here* $(s, t) = (l\sigma, r\sigma) \sqsupset (l'\sigma, r'\sigma)$*. Thus we have* $l'\sigma \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}} r'\sigma$ *by the induction hypothesis.*

From the first condition of (C3), for any condition $u \to v$ *in* $c; c'$ *we have* $(s, t) = (l\theta, r\theta) \sqsupseteq (u\theta, v\theta) = (u\sigma, v\sigma)$*. It follows from* $\langle \xrightarrow{c}{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}}, \sigma \rangle \models c$ *that* $\langle \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}}, \sigma \rangle \models c$ *by the induction hypothesis. Therefore* $s = l\theta = l\sigma \xrightarrow{c}_{\mathcal{E}_2 \cup \mathcal{R}} r\sigma = r\theta = t$*.* □

The simulation completeness for the remaining transformation rules is shown as follows.

Lemma 4.9. *The rules* Nar *and* Def *are simulation complete.*

Proof . *Let $\mathcal{E}_2$ be a CTRS obtained from $\mathcal{E}_1$ by a rule* Nar *or* Def. *We show that for a pattern s on $\mathcal{T}(\mathcal{D}_{\mathcal{E}_1} \cup \mathcal{C})$ and constructor term t, $s \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ implies $s \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}} t$ by induction on k.*

Consider the case that the reduction $s \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} t$ is by a rule $l'' \to r'' \Leftarrow c''$ in $\mathcal{E}_1 \cap \mathcal{E}_2$, that is, $s = l''\theta \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}_1 \cup \mathcal{R}} r''\theta = t$ and $\langle \xrightarrow[\mathsf{c}]{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c''$ for a ground constructor substitution θ. Since $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c''$ by the induction hypothesis, we have $s = l''\theta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}} r''\theta = t$.

Otherwise we separately consider each transformation rule in the transformation.

- Nar*: Let $\mathcal{E}_1 = \mathcal{E} \cup \{\rho : \ l \to r \Leftarrow c; u \to v\}$ and $\mathcal{E}_2 = \mathcal{E} \cup \{l\sigma \to r\sigma \Leftarrow c\sigma; c'\sigma \mid \rho' : \ l' \to r' \Leftarrow c' \in \mathcal{E}_1 \cup \mathcal{R}, \sigma = \mathrm{mgu}(\{u \sim l', v \sim r'\})\}$. We have $s = l\theta$, $t = r\theta$, $\langle \xrightarrow[\mathsf{c}]{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta \rangle \models c$ and $u\theta \xrightarrow[\mathsf{c}]{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}} v\theta$ for a ground constructor substitution θ, where we can assume such that $\mathcal{D}om(\theta) = \mathcal{V}ar(l, r, c, u, v)$ without loss of generality. By the induction hypothesis, $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta \rangle \models c$.*

  Let $l' \to r' \Leftarrow c'$ be the rule used in $u\theta \xrightarrow[\mathsf{c}]{}^{(k-1)}_{\mathcal{E}_1 \cup \mathcal{R}} v\theta$, that is, $u\theta = l'\theta'$, $v\theta = r'\theta'$ and $\langle \xrightarrow[\mathsf{c}]{}^{(k-2)}_{\mathcal{E}_1 \cup \mathcal{R}}, \theta' \rangle \models c'$ for some θ' such that $\mathcal{D}om(\theta') = \mathcal{V}ar(l', r', c')$ since $\mathcal{V}ar(\rho) \cap \mathcal{V}ar(\rho') = \emptyset$. From the induction hypothesis $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}}, \theta' \rangle \models c'$ follows. Since the domains of θ and θ' are disjoint, there exists an extension θ'' of these two substitution. Since $u\theta = l'\theta'$ and $v\theta = r'\theta'$ we have $u\theta'' = l'\theta''$ and $v\theta'' = r'\theta''$. It follows from $\sigma = \mathrm{mgu}(\{u \sim l', v \sim r'\})$ that there exists δ such that $\theta'' = \sigma\delta$. Here $\sigma\delta$ is an extension of θ and θ'. Thus we have $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}}, \sigma\delta \rangle \models c; c'$. Therefore, $s = l\theta = l\sigma\delta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}_2 \cup \mathcal{R}} r\sigma\delta = r\theta = t$.

- Def*: Trivial since $\mathcal{E}_1 \cup \mathcal{R} \subseteq \mathcal{E}_2 \cup \mathcal{R}$.* □

Now we obtained the simulation completeness of $\mathcal{DT}_{123}$.

Theorem 4.10 (Simulation completeness). *Let $\mathcal{R}$ and $\mathcal{E}$ be oriented and pure-constructor CTRSs. If $\mathcal{R}$ is deterministic satisfying $\mathcal{D}_{\mathcal{E}} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$, then the transformations of $\mathcal{DT}_{123}$ are simulation complete, i.e. $\mathcal{E} \overset{*}{\Rrightarrow}_{\mathcal{R}} \mathcal{E}'$ and $s \xrightarrow[\mathsf{c}]{}^{*}_{\mathcal{E} \cup \mathcal{R}} t$ imply $s \xrightarrow[\mathsf{c}]{}^{*}_{\mathcal{E}' \cup \mathcal{R}} t$ for any $s, t \in \mathcal{T}(\mathcal{D}_{\mathcal{E}} \cup \mathcal{D}_{\mathcal{R}} \cup \mathcal{C})$.*

Proof . *Shown by applying Proposition 3.2 and Lemmas 4.6, 4.7, 4.8 and 4.9 repeatedly.* □

5. Syntactic conditions

This section gives syntactic sufficient conditions for the folding transformations.

5.1. On inductive theorems

Here, we discuss the side condition in Fld_0. Consider a transformation sequence $\mathcal{E}_1 \Rrightarrow_{\mathcal{R}} \cdots \Rrightarrow_{\mathcal{R}} \mathcal{E}_n$, and syntactic conditions that guarantee that $l' \to r' \Leftrightarrow \exists X_{c'\backslash l',r'}.c'$ is an inductive theorem of $\mathcal{E}_n$.

Proposition 5.1. *Let $l \to r \Leftarrow c \in \mathcal{E}$ be a rule such that it has no other overlay rule in $\mathcal{E}$. Then $l \to r \Leftrightarrow \exists X_{c\backslash l,r}.c$ is an inductive theorem of $\mathcal{E} \cup \mathcal{R}$. Moreover $l \to r \Leftarrow \exists X_{c\backslash l,r}.c$ satisfies the level condition.*

Proof . *Suppose $\langle \xrightarrow[\mathsf{c}]{}^{(k)}_{\mathcal{E}\cup\mathcal{R}}, \delta\rangle \models c$. Then $l\delta \xrightarrow[\mathsf{c}]{}^{(k+1)}_{\mathcal{E}\cup\mathcal{R}} r\delta$. Thus $l \to r \Leftarrow \exists X_{c\backslash l,r}.c$ is an inductive theorem with level condition.*

We show that $l \to r \Rightarrow \exists X_{c\backslash l,r}.c$ is an inductive theorem. Assume that $l\theta \xrightarrow[\mathsf{c}]{}_{\mathcal{E}\cup\mathcal{R}} r\theta$ for some ground constructor substitution θ. Since $\mathcal{D}_{\mathcal{E}} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$, only the rule $l \to r \Leftarrow c$ is applicable to the reduction. Thus $\langle \xrightarrow[\mathsf{c}]{}_{\mathcal{E}\cup\mathcal{R}}, \delta\rangle \models c$ for some extension δ of θ. □

From the simulation soundness and simulation completeness results, it immediately follows that the transformations of $\mathcal{DT}_{123}$ preserve inductive theorems.

Lemma 5.2. *The transformations of $\mathcal{DT}_{123}$ preserve inductive theorems, that is, if a formula φ is an inductive theorem of $\mathcal{E}\cup\mathcal{R}$, then φ is an inductive theorem of $\mathcal{E}' \cup \mathcal{R}$ for $\mathcal{E} \overset{*}{\Rrightarrow}_{\mathcal{R}} \mathcal{E}'$.*

Proof . *This is shown by structural induction on φ by using Theorems 4.4 and 4.10.* □

The following proposition is also useful.

Proposition 5.3. *A constructor instance of an inductive theorem of $\mathcal{E} \cup \mathcal{R}$ (with level condition) is also an inductive theorem of $\mathcal{E} \cup \mathcal{R}$ (with level condition).*

By combining Propositions 5.1 and 5.3 and Lemma 5.2, we have the following lemma, which works as a syntactic sufficient condition for Fld_0.

Lemma 5.4. *Let $\mathcal{E} \overset{*}{\Rrightarrow}_{\mathcal{R}} \mathcal{E}'$ be a transformation sequence of $\mathcal{DT}_{123}$. Let $l \to r \Leftarrow c$ be a rule in $\mathcal{E}$ having no other overlay rule in $\mathcal{E}$. Then constructor instances of $l \to r \Leftrightarrow \exists X_{c\setminus l,r}.c$ are inductive theorems of $\mathcal{E}' \cup \mathcal{R}$.*

5.2. On level condition

Here we discuss the level condition in (C2). Consider a transformation sequence $\mathcal{E}_1 \Rrightarrow_{\mathcal{R}} \cdots \Rrightarrow_{\mathcal{R}} \mathcal{E}_n$, and an application of Fld_2 to E_n. From the proof of Proposition 5.1, $l' \to r' \Leftarrow c'$ in $\mathcal{E}_n \cup \mathcal{R}$ is an inductive theorem of $\mathcal{E}_n \cup \mathcal{R}$ that satisfies the level condition. This is not so powerful because we often choose a rule $l' \to r' \Leftarrow c'$ not in $\mathcal{E}_n$ but one that previously appeared in $\mathcal{E}_j$ $(1 \leq j < n)$. Recalling Example 7, ρ_{13} is chosen from $\mathcal{E}_{\mathsf{even}}$ in the second folding step. However, it does not satisfy the level condition in $\{\rho_{15}, \rho_{14}\} \cup \mathcal{R}_{\mathsf{double}}$ after modifying ρ_{13} by the first folding step.

Now we introduce a technical notion of descendants. In the transformation $\mathcal{E} \cup \{\rho\} \Rrightarrow_{\mathcal{R}} \mathcal{E} \cup \mathcal{E}'$, we say a rule ρ is transformed into rules in $\mathcal{E}'$ and say each rule in $\mathcal{E}'$ is a *child* of ρ. We use the term *descendant* as the transitive closure of the child relation.

Observing a transformation $\mathcal{E} = \{\rho_1\} \Rrightarrow^{\mathrm{Nar}}_{\mathcal{R}_{\mathsf{add}}} \{\rho_2, \rho_3\} = \mathcal{E}'$ in Example 1, descendants ρ_2 and ρ_3 of ρ_1 are only by Nar. Roughly speaking, ρ_2 is an instance of ρ_1 substituted $\mathsf{0}$ for y, and ρ_3 is an instance of ρ_1 substituted $\mathsf{s}(y')$ for y. We can say that ρ_1 is also an inductive theorem on $\mathcal{E}' \cup \mathcal{R}$ with level condition.

This is summarized as the following lemma.

Lemma 5.5. *Let $\mathcal{E} \overset{*}{\Rrightarrow}_{\mathcal{R}} \mathcal{E}'$ be a transformation sequence of $\mathcal{DT}_{123}$. Let $\rho : l \to r \Leftarrow c$ be in $\mathcal{E}$ such that any descendants of ρ are produced by* Nar *in the sequence. Then ρ is an inductive theorem of $\mathcal{E}' \cup \mathcal{R}$ with level condition.*

Proof . *First we define a notion of proof reductions. As in [2], a reduction of a CTRS is captured as a proof tree. A set of* proof reductions *of $\langle \overset{}{\to}_{\mathsf{c}}, \mathcal{R}, \sigma \rangle \models s \to t$ is the set of all reductions of a proof tree with $s \to t$ as the goal. Note that they are not uniquely determined.*

Secondly we prove the following claim by induction on the number n of the transformation steps.

Let σ be a ground constructor substitution such that $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}'\cup\mathcal{R}}, \sigma\rangle \models c$. Let PR be a set of all proof reductions of $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}'\cup\mathcal{R}}, \sigma\rangle \models c$. Then there exist ground constructor substitutions θ, δ and a rule ρ': $l\theta \to r\theta \Leftarrow c' \in \mathcal{E}'$ such that ρ' is a descendant of ρ, $l\theta\delta = l\sigma$, $r\theta\delta = r\sigma$, $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}'\cup\mathcal{R}}, \delta\rangle \models c'$ and there exists a set of proof reductions of $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}'\cup\mathcal{R}}, \delta\rangle \models c'$ which is a subset of PR.

In the case of $n = 0$ this is trivial by taking empty substitution as θ, c as c', and σ as δ.

We consider the case that $n > 0$. Let $\mathcal{E} \overset{n-1}{\Rrightarrow}_{\mathcal{R}} \mathcal{E}_1 \Rrightarrow_{\mathcal{R}} \mathcal{E}'$. From the simulation soundness of the transformation, we have $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}_1\cup\mathcal{R}}, \sigma\rangle \models c$. Let PR_1 be one of its sets of proof reductions. Then by the induction hypothesis there exist ground constructor substitutions θ_1, δ_1 and a rule ρ_1: $l\theta_1 \to r\theta_1 \Leftarrow c_1 \in \mathcal{E}_1$ such that ρ_1 is a descendant of ρ, $l\theta_1\delta_1 = l\sigma$, $r\theta_1\delta_1 = r\sigma$, $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}_1\cup\mathcal{R}}, \delta\rangle \models c_1$ and there exists a set of proof reductions PR'_1 of $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}_1\cup\mathcal{R}}, \delta_1\rangle \models c_1$ which is a subset of PR_1.

If $\mathcal{E}_1 \Rrightarrow_{\mathcal{R}} \mathcal{E}'$ does not transform ρ_1, the rule ρ_1 is still in $\mathcal{E}'$. We can take θ_1 as θ, δ_1 as δ and ρ_1 as ρ'. We can construct a set of proof reductions of $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}'\cup\mathcal{R}}, \delta_1\rangle \models c_1$ which is a subset of PR by modifying PR'_1 according to the difference between PR_1 and PR.

Suppose $\mathcal{E}_1 \Rrightarrow_{\mathcal{R}} \mathcal{E}'$ transforms ρ_1 by Nar. Then c_1 is represented as $d_1, u \to v$. Since $u\delta_1 \overrightarrow{\mathsf{c}}_{\mathcal{E}_1\cup\mathcal{R}} v\delta_1$, there exists a rule $l' \to r' \Leftarrow d' \in \mathcal{E}_1 \cup \mathcal{R}$ and δ' such that $u\delta_1 = l'\delta'$, $v\delta_1 = r'\delta'$ and $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}_1\cup\mathcal{R}}, \delta'\rangle \models d'$, where $\mathcal{V}ar(\rho_1) \cap \mathcal{V}ar(l', r', d') = \emptyset$ and hence there exists an extension δ'' of δ_1 and δ', without loss of generality. Thus Nar produces a rule $l\theta_1\theta' \to r\theta_1\theta' \Leftarrow d_1\theta'$ where $\theta' = \mathrm{mgu}(\{u \sim l', v \sim r'\})$. Since there exists δ such that $\delta'' = \theta'\delta$, by taking $\theta_1\theta'$ as θ, we have $\theta\delta = \theta_1\theta'\delta = \theta_1\delta''$. Thus $l\theta\delta = l\theta_1\delta'' = l\theta_1\delta_1 = l\sigma$ and similarly $r\theta\delta = r\sigma$. Since $d'\delta' = \theta'\delta$, by taking $d'\theta'$ as c' we have $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}_1\cup\mathcal{R}}, \delta\rangle \models c'$ whose reduction set is a subset of PR_1. We can construct a set of proof reductions of $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}'\cup\mathcal{R}}, \delta\rangle \models c'$ which is a subset of PR.

Thirdly we show that $l \to r \Leftarrow c$ is an inductive theorem of $\mathcal{E}'$ with level condition. Let σ be a ground constructor substitution such that $\langle \overrightarrow{\mathsf{c}}^{(k)}_{\mathcal{E}'\cup\mathcal{R}}, \sigma\rangle \models c$. By the claim, there exist ground constructor substitutions θ, δ and a rule $l\theta \to r\theta \Leftarrow c' \in \mathcal{E}'$ such that $l\theta\delta = l\sigma$, $r\theta\delta = r\sigma$ and $\langle \overrightarrow{\mathsf{c}}^{(k)}_{\mathcal{E}'\cup\mathcal{R}}, \delta\rangle \models c'$. Therefore $l\sigma \overrightarrow{\mathsf{c}}^{(k+1)}_{\mathcal{E}'\cup\mathcal{R}} r\sigma$. □

Example 8. *Consider the transformation sequence in Example 1:*

$$\mathcal{E}_{\mathsf{sub}} = \{\rho_1\} \Rrightarrow^{\mathrm{Nar}}_{\mathcal{R}_{\mathsf{add}}} \{\rho_2, \rho_3\} \Rrightarrow^{\mathrm{Fld}_0}_{\mathcal{R}_{\mathsf{add}}} \{\rho_2, \rho_4\} = \mathcal{R}_{\mathsf{sub}},$$

The second step $\Rightarrow^{Fld_0}_{\mathcal{R}_{\mathsf{add}}}$ *satisfies the condition (C2), which follows from the fact that the descendants* ρ_2 *and* ρ_3 *of* ρ_1 *are produced by* Nar *by Lemma 5.5.*

6. Completeness: strategy and termination

The transformations of $\mathcal{DT}_{123}$ are not terminating in general, because infinitely many applications of Nar and Def are possible.

This section proposes a simple strategy $\mathcal{SS}$ for $\mathcal{DT}_3$. We show that $\mathcal{SS}$ is complete, i.e., the transformations according to $\mathcal{SS}$ always terminate and produce deterministic CTRSs if no conditional part of the rules in $\mathcal{E}$ contains a symbol in $\mathcal{D}_{\mathcal{E}}$ and $\mathcal{R}$ is quasi-decreasing [10, 2].

The first step of $\mathcal{SS}$ selects a non-deterministic rule ρ from $\mathcal{E}$ if one exists, introduces new rewrite rules by Def, and transforms ρ to a deterministic one by applying Fld_3. The second step applies Nar to the new rewrite rules introduced in the first step, and goes back to the first step. We design $\mathcal{SS}$ so that the number of new rewrite rules introduced by Def is finite, which guarantees the termination of the strategy. In the strategy, we assume the set $\mathcal{C}$ of constructors contains tp_i $(0 \leq i \leq m)$ such that $\mathrm{arity}(\mathsf{tp}_i) = i$, where m is the maximum number of $\mathrm{arity}(f)$ in $f \in \mathcal{F}$. By using these constructors we present a tuple $(t_1, \ldots, t_i)$ of terms as $\mathsf{tp}_i(t_1, \ldots, t_i)$ [14].

We explain $\mathcal{SS}$ intuitively by using an example. Consider $\mathcal{E}_{\mathsf{div}}$ and $\mathcal{R}_{\mathsf{mul}}$ in Example 4.

In the first step, we choose a non-deterministic rule from $\mathcal{E}$, if one exists. We have the following non-deterministic rule ρ_7:

$$\rho_7 : \ \mathsf{div}(x, y) \to z \Leftarrow \mathsf{mul}(z, y) \to x; >_{\mathbb{N}}(y, \mathsf{0}) \to \mathsf{tt}.$$

The first condition $\mathsf{mul}(z, y) \to x$ of ρ_7 has one unknown occurrence of z. If we have a function that computes the first argument of mul from the second argument and the result of mul, it is possible to make ρ_7 deterministic by folding. Thus we introduce the following new rewrite rule by Def:

$$\rho_{17} : \ \mathsf{mul}_{[02][1]}(x_0, x_2) \to \mathsf{tp}_1(x_1) \Leftarrow \mathsf{mul}(x_1, x_2) \to x_0,$$

where each number i in the suffix of $\mathsf{mul}_{[02][1]}$ corresponds to x_i in $\mathsf{mul}(x_1, x_2) \to x_0$. The first part of the suffix [02] of $\mathsf{mul}_{[02][1]}$ means that this function takes the result and the second argument of the original function mul, and the second part [1] of the suffix means that the result of $\mathsf{mul}_{[02][1]}$ is the first argument of mul. Note that we don't introduce new rewrite rules if they are already introduced in the past steps.

We can now erase the unknown occurrence in ρ_7 by Fld_0 with ρ_{17}:

$$\{\rho_7\} \Rrightarrow^{\mathrm{Def}}_{\mathcal{R}_{\mathsf{mul}}} \{\rho_7, \rho_{17}\} \Rrightarrow^{\mathrm{Fld}_0}_{\mathcal{R}_{\mathsf{mul}}} \{\rho_{18}, \rho_{17}\}$$

where

$$\rho_{18}: \ \mathsf{div}(x, y) \to z \Leftarrow \mathsf{mul}_{[02][1]}(x, y) \to \mathsf{tp}_1(z); >_{\mathbb{N}}(y, \mathsf{0}) \to \mathsf{tt},$$

and the side condition of Fld_0 is satisfied by Lemma 5.4.

The second step applies Nar to the rewrite rules introduced by Def in the first step. We apply Nar to ρ_{17}, which produces the following three rules:

$$\begin{aligned}
&\rho_{19}: \mathsf{mul}_{[02][1]}(\mathsf{0}, \mathsf{0}) \to \mathsf{tp}_1(x)\\
&\rho_{20}: \mathsf{mul}_{[02][1]}(\mathsf{0}, y) \to \mathsf{tp}_1(\mathsf{0})\\
&\rho_{21}: \mathsf{mul}_{[02][1]}(\mathsf{s}(z), \mathsf{s}(y)) \to \mathsf{tp}_1(\mathsf{s}(x)) \Leftarrow \mathsf{mul}(x, \mathsf{s}(y)) \to w; \mathsf{add}(w, y) \to z.
\end{aligned}$$

Then we go back to the first step. Here rules produced by Nar contain no symbols in $\mathcal{D}_{\mathcal{E}}$.

We repeatedly execute the first and the second steps and finally obtain a deterministic CTRS. Note that this strategy eventually terminates because the function symbols possibly introduced are finitely many.

The $\mathcal{SS}$ is defined as follows:

Step 1: Choose a non-deterministic rule $\rho : l \to r \Leftarrow u_1 \to v_1, \ldots, u_n \to v_n$ from $\mathcal{E}$ if one exists; otherwise stop. Repeat the following until ρ becomes deterministic:

1-1: Let the leftmost unknown variable occurrence in $u_1, \ldots, u_n$ be in u_i. Suppose $u_i \to v_i$ be $f(s_1, \ldots, s_n) \to s_0$. Let w be the sorted sequence of suffixes of s_i's such that s_i has no unknown occurrence of variable. Let w' be the sorted sequence of suffixes of the remaining s_i's. Introduce the following rewrite rule by Def if it has not already been introduced:

$$\rho': \ f_{[w][w']}(\vec{x}_w) \to \mathsf{tp}_{|w'|}(\vec{x}_{w'}) \Leftarrow f(x_1, \ldots, x_n) \to x_0,$$

where $\vec{x}_{w_1 w_2 \cdots w_j}$ represents the list of variables $x_{w_1}, x_{w_2}, \ldots, x_{w_j}$.

1-2: Apply Fld_0 to ρ with a constructor-instance of ρ'.

Step 2: Apply Nar to all ρ's introduced in the last execution of Step 1, and goes back to Step 1.

From the definition of $\mathcal{SS}$, the following properties are obtained.

Proposition 6.1. *Let $\mathcal{E}_0 \Rrightarrow_{\mathcal{R}} \cdots \Rrightarrow_{\mathcal{R}} \mathcal{E}_n$ be a transformation sequence according to the strategy $\mathcal{SS}$. If it holds for $i = 0$ that no conditional part of rules in $\mathcal{E}_i$ contains a symbol in $\mathcal{D}_{\mathcal{E}_i}$, then it holds also for $1 \leq i \leq n$.*

Proposition 6.2. *Transformations according to the strategy $\mathcal{SS}$ eventually terminate.*

Proof . *The application of* Fld_0 *in Step 1-2 erases an unknown variable occurrences in u_i, and hence i increases at least by 1. Thus the loop inside Step 1 terminates.*

Rewrite rules introduced by Def *in Step 1 are non-deterministic, but their number is bounded. One execution of Step 1 makes one rule deterministic. Thus the loop consisting of Step 1 and 2 also terminates.* □

In the rest of this section, we prove the completeness of $\mathcal{DT}_3$ by showing that Fld_0 steps of the strategy satisfy the condition (C3).

Since this section assumes that $\mathcal{R}$ is quasi-decreasing, there exists a well-founded order $\succ$ on $\mathcal{T}(\mathcal{F}un(\mathcal{R}), \mathcal{V})$ that satisfies the conditions of the quasi-decreasing property of $\mathcal{R}$. We design an order $\sqsupset$ on pairs of terms for the condition (C3) based on $\succ$. We regard a pair of terms like $(\mathsf{mul}_{[02][1]}(s_0, s_2), \mathsf{tp}_1(s_1))$ as the term $\mathsf{mul}(s_1, s_2)$ and compare the resulting term by $\succ$. Thus we prepare a function ϕ, which represents such a translation, that is,

$$\phi(\mathsf{mul}_{[02][1]}(s_0, s_2), \mathsf{tp}_1(s_1)) = \mathsf{mul}(s_1, s_2).$$

This is defined as follows.

Definition 6.3. *For each $f \in \mathcal{D}_{\mathcal{R}}$,*

$$\phi(f(s_1, \ldots, s_n), s_0) = f(s_1, \ldots, s_n),$$
$$\phi(f_{[w_1 \cdots w_i][w'_1 \cdots w'_j]}(s_{w_1}, \ldots, s_{w_i}), \mathsf{tp}_j(s_{w'_1}, \ldots, s_{w'_j})) = f(s_1, \ldots, s_n).$$

where $\mathrm{arity}(f) = n$, *$i + j = n + 1$ and $w_1 \cdots w_i w'_1 \cdots w'_j$ is a permutation of $0 \cdots n$.*

Let $\mathcal{E} \overset{*}{\Rrightarrow}_{\mathcal{R}} \mathcal{E}'$ be a transformation according to $\mathcal{SS}$. The partial order $\sqsupset$ on pairs of terms in $\mathcal{F}un(\mathcal{E}' \cup \mathcal{R})$ is the strict part of $\sqsupseteq$, which is defined by $(u, v) \sqsupseteq (u', v')$ if and only if

$$\begin{array}{l} \mathrm{root}(u) \in \mathcal{D}_{\mathcal{E}} \text{ and } \mathrm{root}(u') \notin \mathcal{D}_{\mathcal{E}}, \text{ or} \\ \mathrm{root}(u), \mathrm{root}(u') \notin \mathcal{D}_{\mathcal{E}} \text{ and } \phi(u, v) \succeq \phi(u', v') \end{array}$$

We prepare two propositions related to $\sqsupset$. The former shows that the body part and the conditional part of the rule introduced by Def in Step 1-1 are equal with respect to $\sqsupseteq$. The latter shows that $\sqsupset$ satisfies the conditions of the quasi-decreasing property of $\mathcal{R}$.

Proposition 6.4. *Let $l \to r \Leftarrow u \to v$ be a constructor instance of a rewrite rule introduced by Def in the strategy. Then $(l\theta, r\theta) \sqsupseteq (u\theta, v\theta)$ and $(u\theta, v\theta) \sqsupseteq (l\theta, r\theta)$ for any ground constructor substitution θ.*

Proof . *Let $l \to r \Leftarrow u \to v$ be a constructor instance of $l' \to r' \Leftarrow u' \to v'$. Then $l = l'\sigma$, $r = r'\sigma$, $u = u'\sigma$ and $v = v'\sigma$ for some ground constructor substitution. From the construction of rewrite rules introduced by Def in the strategy, we have $\phi(l', r') = \phi(u', v') = u'$, and hence $\phi(l'\sigma\theta, r'\sigma\theta) = \phi(u'\sigma\theta, v'\sigma\theta) = u\sigma\theta$. Therefore the proposition holds.* □

Proposition 6.5. *Let $\mathcal{R}$ be a quasi-decreasing CTRS, which satisfies $\mathcal{D}_{\mathcal{E}} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$. Let $l \to r \Leftarrow c$ be a rewrite rule in $\mathcal{R}$. Then $(l\theta, r\theta) \sqsupset (u\theta, v\theta)$ for any condition $u \to v$ in c and any ground constructor substitution θ such that $\langle \overset{c}{\to}_{\mathcal{E}\cup\mathcal{R}}, \theta \rangle \models c$.*

Proof . *Let $\langle \overset{c}{\to}_{\mathcal{E}\cup\mathcal{R}}, \theta \rangle \models c$. Since $\mathcal{D}_{\mathcal{E}} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$, we have $\langle \overset{c}{\to}_{\mathcal{R}}, \theta \rangle \models c$. Since $\phi(l\theta, r\theta) = l\theta$ and $\phi(u\theta, v\theta) = u\theta$, and $\mathcal{R}$ is quasi-decreasing, it follows that $\phi(l\theta, r\theta) = l\theta \succ u\theta = \phi(u\theta, v\theta)$. Thus $(l\theta, r\theta) \sqsupset (u\theta, v\theta)$.* □

Now we give the key lemma of this section which asserts that Fld_0 steps in $\mathcal{SS}$ are also Fld_3 steps.

Lemma 6.6. *Let $\mathcal{R}$ and $\mathcal{E}_0$ be oriented and pure-constructor CTRSs such that $\mathcal{D}_{\mathcal{E}_0} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$ and no conditional part of rules in $\mathcal{E}_0$ contains a symbol in $\mathcal{D}_{\mathcal{E}_0}$. Let $\mathcal{R}$ be deterministic and quasi-decreasing. Then the transformation sequence $\mathcal{E}_0 \Rrightarrow_{\mathcal{R}} \cdots \Rrightarrow_{\mathcal{R}} \mathcal{E}_n$ according to $\mathcal{SS}$ is also a $\mathcal{DT}_3$ sequence.*

Proof . *We show the following claims by induction on n:*

1. *The sequence $\mathcal{E}_0 \overset{*}{\Rrightarrow}_{\mathcal{R}} \mathcal{E}_n$ is a $\mathcal{DT}_3$ sequence.*
2. *For any rewrite rule $l \to r \Leftarrow c$ in $\mathcal{E}_i$ $(0 \leq i \leq n)$, if it is not introduced by* Def *in Step 1-1 then*

   $$(l\theta, r\theta) \sqsupset (u\theta, v\theta)$$ *for any condition $u \to v$ in c and any ground constructor substitution θ such that $\langle \overset{\to}{\mathsf{c}}_{\mathcal{E}_n \cup \mathcal{R}}, \theta \rangle \models c$.*

In the case of $n = 0$, the claim 1 trivially holds, hence we will show the claim 2. Let $l \to r \Leftarrow c$ be a rewrite rule in $\mathcal{E}_0$ such that $\langle \overset{\to}{\mathsf{c}}_{\mathcal{E}_0 \cup \mathcal{R}}, \theta \rangle \models c$. We have $\mathrm{root}(l\theta) \in \mathcal{D}_{\mathcal{E}_0}$ *and* $\mathrm{root}(u\theta) \notin \mathcal{D}_{\mathcal{E}_0}$*. Thus $(l\theta, r\theta) \sqsupset (u\theta, v\theta)$ follows.*

Consider the case $n > 0$. We can write the sequence as $\mathcal{E}_0 \overset{n-1}{\Rrightarrow}_{\mathcal{R}} \mathcal{E}_{n-1} \Rrightarrow_{\mathcal{R}} \mathcal{E}_n$. For the claim 1, it is enough to consider the case that Fld_0 *is applied in the last step. Let $\mathcal{E}_1 = \mathcal{E} \cup \{l \to r \Leftarrow (c, u \to v)\}$ and $\mathcal{E}_2 = \mathcal{E} \cup \{l \to r \Leftarrow (c, l' \to r')\}$, where $l' \to r' \Leftarrow u \to v$ is a constructor instance of a previously introduced rule by* Def*. Suppose this rule is introduced in $\mathcal{E}_i$. Since the rule has no other overlay rule in $\mathcal{E}_i \cup \mathcal{R}$, it is shown by Lemma 5.4 and the induction hypothesis on the claim 1 that $l' \to r' \Leftrightarrow \exists X_{c' \setminus l', r'}.c'$ is an inductive theorem of $\mathcal{E}'' \cup \mathcal{R}$. The condition (C3) holds from Propositions 6.4, 6.5 and the induction hypothesis of the claim 2. Therefore the claim 1 holds.*

For the claim 2, we have three cases according to the transformation rule used in the last step. Here, from the soundness of the transformation (Theorems 4.4 and 4.10) and the claim 1, we have $\overset{\to}{\mathsf{c}}_{\mathcal{E}_{n-1} \cup \mathcal{R}} = \overset{\to}{\mathsf{c}}_{\mathcal{E}_n \cup \mathcal{R}}$.

- Nar*: In the strategy,* Nar *is used in Step 1-2 and is applied only to rules introduced by* Def*. Let $\mathcal{E}_{n-1} = \mathcal{E} \cup \{l \to r \Leftarrow u \to v\}$ and $\mathcal{E}_n = \mathcal{E} \cup \{l\sigma \to r\sigma \Leftarrow c'\sigma \mid l' \to r' \Leftarrow c' \in \mathcal{R}, \sigma = \mathrm{mgu}(\{u \sim l', v \sim r'\})\}$. Let $l\sigma \to r\sigma \Leftarrow c'\sigma$ be a rule produced in the last transformation step, and θ be a ground constructor substitution such that $\langle \overset{\to}{\mathsf{c}}_{\mathcal{E}_n \cup \mathcal{R}}, \theta \rangle \models c'\sigma$. Let $u' \to v'$ be a condition in c'. For the claim 2, it suffices to show that $(l\sigma\theta, r\sigma\theta) \sqsupset (u'\sigma\theta, v'\sigma\theta)$.*

  By Proposition 6.4, $(l\sigma\theta, r\sigma\theta) \sqsupseteq (u\sigma\theta, v\sigma\theta)$. Since σ is the most general unifier, we have $u\sigma = l'\sigma$ and $v\sigma = r'\sigma$. Thus $(u\sigma\theta, v\sigma\theta) = (l'\sigma\theta, r'\sigma\theta)$. Since $\langle \overset{\to}{\mathsf{c}}_{\mathcal{E}_n \cup \mathcal{R}}, \sigma\theta \rangle \models c'$, we have $(l'\sigma\theta, r'\sigma\theta) \sqsupset (u'\sigma\theta, v'\sigma\theta)$ by Proposition 6.5. Therefore $(l\sigma\theta, r\sigma\theta) \sqsupset (u'\sigma\theta, v'\sigma\theta)$.

- Fld_0*: In the strategy,* Fld_0 *is used in Step 1-1. Let $\mathcal{E}_1 = \mathcal{E} \cup \{l \to r \Leftarrow c; u \to v\}$ and $\mathcal{E}_2 = \mathcal{E} \cup \{l \to r \Leftarrow c; l' \to r'\}$, where $l' \to r' \Leftarrow u \to v$ is a constructor instance of previously introduced rule by* Def*. The rule*

$l \to r \Leftarrow c; u \to v$ *is not the one introduced by* Def*, because rules introduced by* Def *disappear in Step 1-2.*

Suppose $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}_n \cup \mathcal{R}}, \sigma \rangle \models c; u \to v$*, then* $\langle \overrightarrow{\mathsf{c}}_{\mathcal{E}_{n-1} \cup \mathcal{R}}, \sigma \rangle \models c; u \to v$ *since* $\overrightarrow{\mathsf{c}}_{\mathcal{E}_{n-1} \cup \mathcal{R}} = \overrightarrow{\mathsf{c}}_{\mathcal{E}_n \cup \mathcal{R}}$*. By the induction hypothesis of the claim 2, we have* $(l\sigma, r\sigma) \sqsupseteq (u\sigma, v\sigma)$ *and* $(l\sigma, r\sigma) \sqsupseteq (u'\sigma, v'\sigma)$ *for any* $u' \to v'$ *in* c*. By Proposition 6.4,* $(u\sigma, v\sigma) \sqsupseteq (l'\sigma, r'\sigma)$*. Thus* $(l\sigma, r\sigma) \sqsupseteq (u\sigma, v\sigma) \sqsupseteq (l'\sigma, r'\sigma)$ *follows.*

For rules in $\mathcal{E}$*, the claim 2 follows from* $\overrightarrow{\mathsf{c}}_{\mathcal{E}_{n-1} \cup \mathcal{R}} = \overrightarrow{\mathsf{c}}_{\mathcal{E}_n \cup \mathcal{R}}$ *and the induction hypothesis of the claim 2.*

- Def*: In this case, we have nothing to show for the claim 2.* □

Summarizing this section, we obtain the following theorem.

Theorem 6.7 (Completeness). *Let* $\mathcal{R}$ *and* $\mathcal{E}$ *be oriented and pure-constructor CTRSs such that* $\mathcal{D}_{\mathcal{E}} \cap \mathcal{F}un(\mathcal{R}) = \emptyset$ *and no conditional part of any rule in* $\mathcal{E}$ *contains a symbol in* $\mathcal{D}_{\mathcal{E}}$*. If* $\mathcal{R}$ *is deterministic and quasi-decreasing, then the transformation according to the strategy* $\mathcal{SS}$ *eventually stops with producing a deterministic CTRS.*

In this section, we developed the strategy $\mathcal{SS}$, which is terminating and always produces deterministic CTRSs if no conditional part of the rules in $\mathcal{E}$ contains a symbol in $\mathcal{D}_{\mathcal{E}}$ and $\mathcal{R}$ is quasi-decreasing. This means that $\mathcal{DT}_{123}$ is a complete determinization transformation system for non-deterministic CTRSs satisfying the restrictions. The former restriction is not a problem from the viewpoint of program generation of rewriting systems. A specification of f is usually given as properties c that must be satisfied, which is a relationship between inputs $x_1, \ldots, x_n$ of f and its output y. This is representable in the form of $f(x_1, \ldots, x_n) \to y \Leftarrow c$. The latter restriction is currently problematic since quasi-decreasingness, which is equivalent to the operational termination, is difficult to show. The deterministic CTRSs presented in this paper including the resulting CTRSs by the determinization are, nevertheless, quasi-decreasing except for $\mathcal{R}_{\mathsf{div}}$, $\mathcal{R}'_{\mathsf{lib}}$, $\mathcal{R}'_{\mathsf{ssp}}$ and $\mathcal{R}''_{\mathsf{ssp}}$. These quasi-decreasing CTRSs have equivalent simply terminating TRSs, where a simply terminating TRS [15] is a TRS whose termination is proved by a simplification order [12]. It is not difficult to show that a pure-constructor CTRS transformed from a simply terminating TRS is quasi-decreasing because a simplification order has the subterm property, i.e. $l \succ C[t]$ implies $l \succ t$.

7. Examples

In this section, we present larger examples.

Example 9. *In this example, we generate a deterministic CTRS* $\mathcal{R}_{\mathsf{ssp}}$*, which solves the sub-list summation problem. The inputs and outputs of* ssp *are described below:*

Input: *a list* xs *of natural numbers and a natural number* v*.*

Output: *a sub-list* ys *of* xs *satisfying* $\sum_{y \in ys} y = v$*.*

CTRSs $\mathcal{E}_{\mathsf{ssp}}$ *and* $\mathcal{R}_{\mathsf{lib}}$ *are given as follows:*

$$\mathcal{E}_{\mathsf{ssp}} = \{\, \mathsf{ssp}(xs, v) \to ys \Leftarrow \mathsf{subL}(ys, xs) \to \mathsf{tt}; \mathsf{sum}(ys) \to v \,\},$$

$$\mathcal{R}_{\mathsf{lib}} = \left\{ \begin{array}{l} \mathsf{subL}(\mathsf{nil}, \mathsf{nil}) \to \mathsf{tt}, \\ \mathsf{subL}(xs, y :: ys) \to z \Leftarrow \mathsf{subL}(xs, ys) \to z, \\ \mathsf{subL}(y :: xs, y :: ys) \to z \Leftarrow \mathsf{subL}(xs, ys) \to z, \\ \mathsf{sum}(\mathsf{nil}) \to \mathsf{0}, \\ \mathsf{sum}(x :: xs) \to z \Leftarrow \mathsf{sum}(xs) \to w; \mathsf{add}(w, x) \to z, \\ \mathsf{add}(x, \mathsf{0}) \to x, \\ \mathsf{add}(x, \mathsf{s}(y)) \to \mathsf{s}(z) \Leftarrow \mathsf{add}(x, y) \to z \end{array} \right\}.$$

As a result of the transformation shown in Figure 5, we obtained a CTRS :

$$\mathcal{R}_{\mathsf{ssp}} = \left\{ \begin{array}{l} \mathsf{ssp}(\mathsf{nil}, \mathsf{0}) \to \mathsf{nil}, \\ \mathsf{ssp}(y :: ys', v) \to xs \Leftarrow \mathsf{ssp}(ys', v) \to xs, \\ \mathsf{ssp}(y :: ys', v) \to y :: xs' \\ \quad \Leftarrow \mathsf{sub}(v, y) \to w; \mathsf{ssp}(ys', w) \to xs', \\ \mathsf{sub}(z, \mathsf{0}) \to z, \\ \mathsf{sub}(\mathsf{s}(v), \mathsf{s}(w)) \to z \Leftarrow \mathsf{sub}(v, w) \to z, \end{array} \right\},$$

which can solve the sub-list summation problem. For example, $\mathsf{ssp}($`[1,2,3]`$, 3)$ *is reducible to terms* `[1,2]` *and* `[3]`*.*

Example 10. *This is another example, we generate a deterministic CTRS* $\mathcal{R}'_{\mathsf{ssp}}$*, which is the multi-set version of the subset summation problem. CTRSs* $\mathcal{E}'_{\mathsf{ssp}}$ *and* $\mathcal{R}'_{\mathsf{lib}}$ *are given as follows:*

$$\mathcal{E}'_{\mathsf{ssp}} = \{\, \mathsf{ssp}'(xs, v) \to ys \Leftarrow \subseteq_m (ys, xs) \to \mathsf{tt}; \mathsf{sum}(ys) \to v \,\},$$

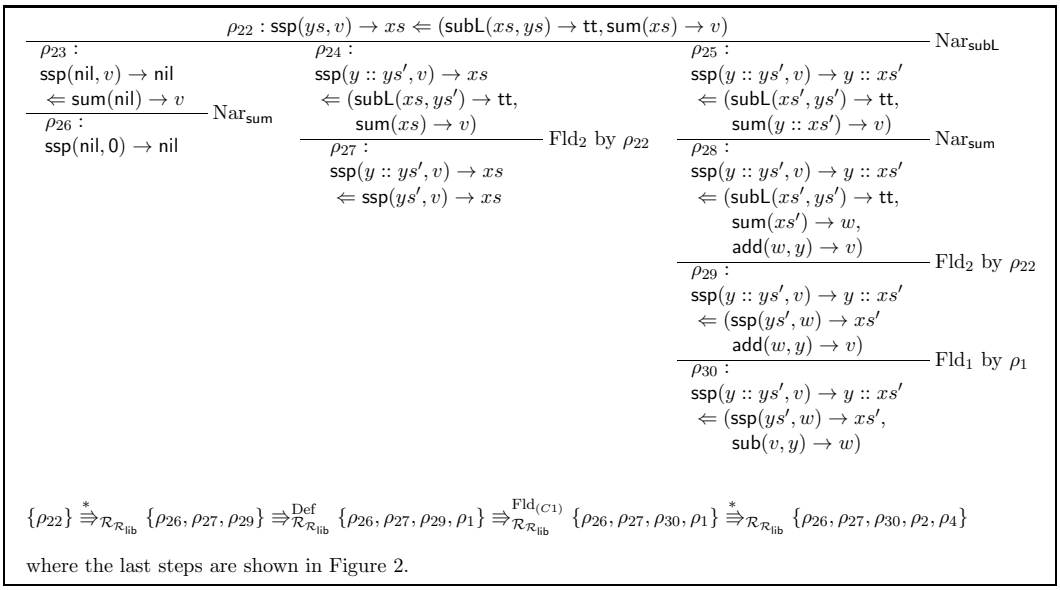

Figure 5: Transformation of ssp in Example 9

$$\mathcal{R}'_{\mathsf{lib}} = \left\{ \begin{array}{l} \subseteq_m(\mathsf{nil}, ys) \to \mathsf{tt}, \\ \subseteq_m(x :: xs, ys) \to z \\ \quad \Leftarrow \mathsf{del}(x, ys) \to \mathsf{some}(ws); \subseteq_m(xs, ws) \to z, \\ \mathsf{del}(x, \mathsf{nil}) \to \mathsf{none}, \\ \mathsf{del}(x, x :: ys) \to \mathsf{some}(ys), \\ \mathsf{del}(x, y :: ys) \to \mathsf{some}(y :: zs) \Leftarrow \mathsf{del}(x, ys) \to \mathsf{some}(zs), \\ \mathsf{sum}(\mathsf{nil}) \to \mathsf{0}, \\ \mathsf{sum}(x :: xs) \to z \Leftarrow \mathsf{sum}(xs) \to w; \mathsf{add}(w, x) \to z, \\ \mathsf{add}(x, \mathsf{0}) \to x, \\ \mathsf{add}(x, \mathsf{s}(y)) \to \mathsf{s}(z) \Leftarrow \mathsf{add}(x, y) \to z \end{array} \right\}.$$

We omit to present the transformation because it is rather complex. We

obtain the following CTRS:

$$\mathcal{R}'_{\mathsf{ssp}} = \left\{ \begin{array}{l} \mathsf{ssp}'(xs, \mathsf{0}) \to \mathsf{nil}, \\ \mathsf{ssp}'(y' :: ws, v) \to y' :: ys' \\ \quad \Leftarrow \mathsf{sub}(v, y') \to w; \mathsf{ssp}'(ws, w) \to ys', \\ \mathsf{ssp}'(x' :: xs', v) \to y' :: ys' \\ \quad \Leftarrow \mathsf{get}(xs') \to \mathsf{tp}_2(y', zs); \mathsf{sub}(v, y') \to w; \\ \qquad \mathsf{ssp}'(x' :: zs, w) \to ys', \\ \mathsf{sub}(z, \mathsf{0}) \to z, \\ \mathsf{sub}(\mathsf{s}(v), \mathsf{s}(w)) \to z \Leftarrow \mathsf{sub}(v, w) \to z, \\ \mathsf{get}(y :: ys) \to \mathsf{tp}_2(y, ys), \\ \mathsf{get}(x' :: xs') \to \mathsf{tp}_2(y, x' :: zs) \Leftarrow \mathsf{get}(xs') \to \mathsf{tp}_2(y, zs) \end{array} \right\},$$

which can solve the subset summation problem. The term $\mathsf{ssp}'(\texttt{[1,2,3]}, \mathsf{3})$ *is reducible to terms* `[1,2]`, `[2,1]` *and* `[3]`.

8. Related works

To the best of our knowledge, there has been no existing work on determinization of CTRSs. However, in the course of our work, we have noticed that the determinization is closely related to partial program inversion [16, 9].

Partial inversion of $g(x, y)$, where the value of the second argument is given, is a function $\overline{g}_{\{2\}}(z, y)$ satisfying $g(x, y) = z \iff \overline{g}_{\{2\}}(z, y) = x$, where the set $\{2\}$ contains the positions of the known arguments. The function $\overline{g}_{\{2\}}$ directly corresponds to $g_{[02][1]}$ which discussed in Section 6. This means that if we can compute partial inversion functions, non-deterministic CTRSs can be determinized in the same way as in Section 6.

There are a lot of studies on program inversion [17, 18, 19, 20, 21, 14, 22, 16, 23, 24, 9, 5]. Most of them are devoted to produce a full-inversion function, that is, a function $\bar{g}_{\{\}}(z)$ satisfying $g(x, y) = z \iff \bar{g}_{\{\}}(z) = (x, y)$. Unfortunately full-inversion is insufficient for the determinization.

Nishida et al. presented a partial program-inversion algorithm for term rewriting systems [16]. Their algorithm consists of two stages. The first stage transforms a constructor TRS, which is a CTRS having no conditions, into a CTRS that defines partial inversion functions. The second stage transforms the CTRS obtained from the first stage into a TRS by using the unraveling technique. It is shown that the first stage preserves innermost reduction. The relationship between the method of [16] and our transformation according to the strategy $\mathcal{SS}$ is summarized as follows:

- The first stage of [16] allows only TRSs as input. Transforming a constructor CTRS into an equivalent TRS is difficult even for deterministic CTRSs [25, 26, 27].
- The transformation according to $\mathcal{SS}$ may possibly simulate the first stage of [16], if we combine the transformation according to $\mathcal{SS}$ with the transformation from a constructor TRS into an equivalent pure-constructor CTRS. (This is a conjecture.)

A non-mechanical transformation of $\mathcal{DT}_{123}$ can produce results different from the first stage of [16]. For instance, no program inversion method obtains $\mathcal{R}_{\mathsf{ssp}}$ in Example 9, which works efficiently. The inversion technique, instead, produces the following CTRS $\mathcal{R}''_{\mathsf{ssp}}$, which is less efficient than $\mathcal{R}_{\mathsf{ssp}}$:

$$\mathcal{R}''_{\mathsf{ssp}} = \mathcal{R}_{\mathsf{add}} \cup \mathcal{R}_{\mathsf{sub}} \cup \left\{ \begin{array}{l} \mathsf{ssp}(xs, v) \to ys \Leftarrow \overline{\mathsf{subL}}_{\{2\}}(\mathsf{tt}, xs) \to ys; \mathsf{sum}(ys) \to v, \\ \overline{\mathsf{subL}}_{\{2\}}(\mathsf{tt}, \mathsf{nil}) \to \mathsf{nil}, \\ \overline{\mathsf{subL}}_{\{2\}}(z, y :: ys) \to xs \Leftarrow \overline{\mathsf{subL}}_{\{2\}}(z, ys) \to xs, \\ \overline{\mathsf{subL}}_{\{2\}}(z, y :: ys) \to y :: xs \Leftarrow \overline{\mathsf{subL}}_{\{2\}}(z, ys) \to xs, \\ \mathsf{sum}(\mathsf{nil}) \to 0, \\ \mathsf{sum}(x :: xs) \to z \Leftarrow \mathsf{sum}(xs) \to w; \mathsf{add}(w, x) \to z \end{array} \right\}.$$

This $\mathcal{R}''_{\mathsf{ssp}}$ can also be produced by our transformation $\mathcal{DT}_3$ with the strategy $\mathcal{SS}$.

Almendros-Jiménez and Vidal proposed a simple method for partial program inversion [9]. Their method is applicable for a subclass of constructor and left-linear TRSs, which means that their method can be applied no wider than the method of [16].

On the other hand, determinization can be regarded as a program generation, because a non-deterministic CTRS is also regarded as a non-executable specification. The study of program generation has a long history. A well-known method is unfold/fold transformation, firstly introduced for functional programs [28] and formulated for logic programs [11] (see the paper [29] for a survey). This method has also been studied extensively for logic [30, 31, 32, 33, 34, 35], constraint logic [36, 37, 38, 39], functional [40] and functional-logic [41] programs.

There is a one-to-one correspondence between the constructor-based reduction of a pure-constructor CTRS and the execution of the CTRS as a

definite logic program, where a definite logic program is a set of clauses without negative literals in the premise. For example, $\mathsf{add}(\mathsf{0},\mathsf{s}(\mathsf{0})) \overset{(2)}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{R}_{\mathsf{add}}} \mathsf{s}(\mathsf{0})$ because $\mathsf{add}(\mathsf{0},\mathsf{0}) \overset{(1)}{\underset{\mathsf{c}}{\rightarrow}}_{\mathcal{R}_{\mathsf{add}}} \mathsf{0}$. On the other hand, a definite logic program is obtained from $\mathcal{R}_{\mathsf{add}}$ by regarding each condition $f(t_1,\ldots,t_n) \to t_0$ as an atom $F(t_1,\ldots,t_n,t_0)$:

$$\left\{ \begin{array}{l} \mathsf{ADD}(x,\mathsf{0},x) \\ \mathsf{ADD}(x,\mathsf{s}(y),\mathsf{s}(z)) \Leftarrow \mathsf{ADD}(x,y,z) \end{array} \right\}.$$

Under the program, $\mathsf{ADD}(\mathsf{0},\mathsf{s}(\mathsf{0}),\mathsf{s}(\mathsf{0}))$ holds because $\mathsf{ADD}(\mathsf{0},\mathsf{0},\mathsf{0})$ holds. In this sense, our transformation system is deeply related to fold/unfolding transformations of definite logic programs [30, 31, 32, 33, 34, 35]. Actually, the unfolding rule corresponds to Nar. The goal replacement rule corresponds to Fld_1. The logical soundness of transformations of logic programming is defined as follows: an atomic formula is true under the initial program if it is true under the final program. The logical soundness corresponds to the simulation soundness. Similarly the logical completeness corresponds to the simulation completeness.

Since the logical completeness does not hold in general and depends on the termination property of final program [28], the history of studies on fold/unfold transformation can be rephrased as a history of fights with termination. Roychoudhury et al. proposed a general treatment of termination for the logical completeness [34]. They introduced a measure for atoms and two measures for clauses on a single domain, and defined measure-preserving folding/unfolding. They proved that the system is logically complete if the initial program is measure-consistent. The main differences between the method [34] and the method of this paper are as follows:

- The method in this paper does not define simultaneous folding with multiple rules.
- Measures are fixed in advance in the method [34], while in our approach orders are allowed to be different in each folding application.

Pettorossi et al. proposed a method to attach integer constraints to a fold/unfold transformation sequence and to check the logical completeness by the satisfiability of the constraints [35]. This approach has a benefit that a relevant measure is found by the satisfiability check. The measure found by satisfiability check is for the entire transformation, which is the main difference between [35] and this paper.

9. Concluding remarks

9.1. Garbage

Infeasible rewrite rules, whose conditions are never satisfied, may be produced. For example, both the conditions $=_{\mathbb{N}}(y, x) \to \mathsf{tt}$ and $=_{\mathbb{N}}(y, x) \to \mathsf{ff}$ may appear in a rule. If we know that $=_{\mathbb{N}}$ has the confluence property, we can remove the infeasible rule. However, it is generally difficult to remove infeasible rules in a systematic way.

9.2. Obtaining efficient CTRSs

The transformations of $\mathcal{DT}_3$ are mechanized by the strategy $\mathcal{SS}$ proposed in Section 6. However, the resulting CTRS is not efficient in general. For example, we can generate $\mathcal{R}_{\mathsf{ssp}}$, which works efficiently, by $\mathcal{DT}_{123}$ as shown in Example 9, but the strategy $\mathcal{SS}$ generates the less efficient $\mathcal{R}''_{\mathsf{ssp}}$ as shown in Section 8. Thus we should develop another strategy that generates more efficient CTRSs mechanically.

Acknowledgments

We thank anonymous referees for their valuable comments, Naoki Nishida for discussions related to inverse computation, and René Thiemann for contributing the initial idea of similarity between evaluation mechanism of logic programs and constructor-based reduction on pure-constructor CTRS. This work was partially supported by Grant-in-Aid for Scientific Research #20300010.

References

[1] H. Ganzinger, Order-sorted completion: The many-sorted way, Theoretical Computer Science 89 (1) (1991) 3–32.

[2] S. Lucas, C. Marché, J. Meseguer, Operational termination of conditional term rewriting systems, Information Processing Letters 95 (2005) 446–453.

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

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

[5] N. Nishida, G. Vidal, Program inversion for tail recursive functions, in: M. Schmidt-Schauß (Ed.), Proceedings of the 22nd International Conference on Rewriting Techniques and Applications, Vol. 10 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011, pp. 283–298.

[6] B. Gramlich, On the (non-)existence of least fixed points in conditional equational logic and conditionl rewriting, in: I. Guessarian (Ed.), Proceedings of the 2nd International Workshop on Fixed Points in Computer Science (FICS 2000) – Extended Abstracts, 2000, pp. 38–40.

[7] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.

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

[9] J. M. Almendros-Jiménez, G. Vidal, Automatic partial inversion of inductively sequential functions, in: Z. Horváth, V. Zsók, A. Butterfield (Eds.), Proceedings of 18th International Symposium on Implementation and Application of Functional Languages, Vol. 4449 of Lecture Notes in Computer Science, 2007, pp. 253–270.

[10] J. Giesl, T. Arts, Verification of Erlang processes by dependency pairs, Applicable Algebra in Engineering, Communication and Computing 12 (1/2) (2001) 39–72.

[11] H. Tamaki, T. Sato, Unfold/fold transformation of logic programs, in: S.-Å. Tärnlund (Ed.), Proceedings of the 2nd International Conference on Logic Programming, 1984, pp. 127–138.

[12] N. Dershowitz, Orderings for term-rewriting systems, Theoretical Computer Science 17 (1982) 279–301.

[13] T. Arts, J. Giesl, Termination of term rewriting using dependency pairs, Theoretical Computer Science 236 (1-2) (2000) 133–178.

[14] N. Nishida, M. Sakai, T. Sakabe, Generation of inverse term rewriting systems for pure treeless functions, in: Y. Toyama (Ed.), Proceedings of the International Workshop on Rewriting in Proof and Computation, 2001, pp. 188–198.

[15] M. Kurihara, A. Ohuchi, Modularity of simple termination of term rewriting systems with shared constructors, Theoretical Computer Science 103 (1992) 273–282.

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

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

[18] P. G. Harrison, Function inversion, in: Proceedings of IFIP TC2 Workshop on Partial Evaluation and Mixed Computation, North-Holland, 1988, pp. 153–166.

[19] A. Romanenko, The generation of inverse functions in Refal, in: Proceedings of IFIP TC2 Workshop on Partial Evaluation and Mixed Computation, North-Holland, 1988, pp. 427–444.

[20] H. Khoshnevisan, K. M. Sephton, Invx: An automatic function inverter, in: N. Dershowitz (Ed.), Proceedings of the 3rd International Conference on Rewriting Techniques and Applications, Vol. 355 of Lecture Notes in Computer Science, 1989, pp. 564–568.

[21] A. Romanenko, Inversion and metacomputation, in: Proceedings of the Symposium on Partial Evaluation and Semantics-Based Program Manipulation, ACM, 1991, pp. 12–22.

[22] R. Glück, M. Kawabe, A program inverter for a functional language with equality and constructors, in: A. Ohori (Ed.), Programming Languages and Systems, First Asian Symposium, Vol. 2895 of Lecture Notes in Computer Science, Springer, 2003, pp. 246–264.

[23] R. Glück, M. Kawabe, A method for automatic program inversion based on LR(0) parsing, Fundamenta Informmaticae 66 (4) (2005) 367–395.

[24] M. Kawabe, R. Glück, The program inverter lrinv and its structure, in: M. V. Hermenegildo, D. Cabeza (Eds.), Practical Aspects of Declarative Languages, 7th International Symposium, Vol. 3350 of Lecture Notes in Computer Science, Springer, 2005, pp. 219–234.

[25] M. Marchiori, Unravelings and ultra-properties, in: M. Hanus, M. Rodríguez-Artalejo (Eds.), Algebraic and Logic Programming, 5th International Conference, Vol. 1139 of Lecture Notes in Computer Science, Springer, 1996, pp. 107–121.

[26] N. Nishida, M. Sakai, T. Sakabe, Soundness of unravelings for conditional term rewriting systems via ultra-properties related to linearity, Logical Methods in Computer Science 8 (3) (2012) 1–49.

[27] K. Gmeiner, B. Gramlich, F. Schernhammer, On soundness conditions for unraveling deterministic conditional rewrite systems, in: A. Tiwari (Ed.), 23rd International Conference on Rewriting Techniques and Applications, Vol. 15 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012, pp. 193–208.

[28] R. M. Burstall, J. Darlington, A transformation system for developing recursive programs, Journal of the ACM 24 (1) (1977) 44–67.

[29] A. Pettorossi, M. Proietti, Rules and strategies for transforming functional and logic programs, ACM Computing Surveys 28 (1996) 360–414.

[30] T. Kanamori, K. Horiuchi, Construction of logic programs based on generalized unfold/fold rules, in: J.-L. Lassez (Ed.), Proceedings of the 4th International Conference on Logic Programming, 1987, pp. 744–768.

[31] T. Sato, Equivalence-preserving first-order unfold/fold transformation systems, Theoretical Computer Science 105 (1992) 57–84.

[32] H. Seki, Unfold/fold transformation of general logic programs for the well-founded semantics, Journal of Logic Programming 16 (1) (1993) 5–23.

[33] A. Roychoudhury, K. N. Kumar, C. R. Ramakrishnan, I. V. Ramakrishnan, Beyond Tamaki-Sato style unfold/fold transformations for normal logic programs, International Journal of Foundations of Computer Science 13 (3) (2002) 387–403.

[34] A. Roychoudhury, K. N. Kumar, C. R. Ramakrishnan, I. V. Ramakrishnan, An unfold/fold transformation framework for definite logic programs, ACM Transactions on Programming Languages and Systems 26 (3) (2004) 464–509.

[35] A. Pettorossi, M. Proietti, V. Senni, Constraint-based correctness proofs for logic program transformations, Formal Aspects of Computing 24 (4-6) (2012) 569–594.

[36] M. J. Maher, A transformation system for deductive database modules with perfect model semantics, Theoretical Computer Science 110 (1993) 377–403.

[37] S. Etalle, M. Gabbrielli, Transformations of CLP modules, Theoretical Computer Science 166 (1-2) (1996) 101–146.

[38] N. Bensaou, I. Guessarian, Transforming constraint logic programs, Theoretical Computer Science 206 (1-2) (1998) 81–125.

[39] F. Fioravanti, A. Pettorossi, M. Proietti, Transformation rules for locally stratified constraint logic programs, in: M. Bruynooghe, K.-K. Lau (Eds.), Program Development in Computational Logic, Vol. 3049 of Lecture Notes in Computer Science, Springer, 2004, pp. 77–89.

[40] D. Sands, Total correctness by local improvement in the transformation of functional programs, ACM Transactions on Programming Languages and Systems 18 (2) (1996) 175–234.

[41] M. Alpuente, M. Falaschi, G. Moreno, G. Vidal, Rules + strategies for transforming lazy functional logic programs, Theoretical Computer Science 311 (2004) 479–525.