PROVING AND DISPROVING TERMINATION OF CONSTRAINED TERM REWRITING SYSTEMS

NAOKI NISHIDA1 AND TSUBASA SAKATA1 AND MASAKI SAKAI1 AND TATSUYA HATTORI1

1 Furo-cho, Chikusa-ku, Nagoya 464-8603, Japan
Graduate School of Information Science, Nagoya University

ABSTRACT. In this paper, we propose methods for proving and disproving termination of constrained term rewriting systems, where constraints are interpreted by built-in semantics given by users, and rewrite rules are sound for the interpretation, i.e., the interpretation of terms is preserved under the reduction. The method for proving termination is a straightforward extension of the dependency pair framework for unconstrained term rewriting systems, and is applicable if the semantics is over linear integer arithmetics. On the other hand, the non-termination proof has a gap from the case of unconstrained systems because constrained rewrite rules such that the right-hand sides are encompassments of the left-hand sides do not always cause non-termination. For such constrained rewrite rules, we characterize extra constraints that ensure non-termination caused by the constrained rewrite rules. We also show that such extra constraints sometimes can be obtained from the constraints of the constrained rewrite rules, by weakening the constraints. The method for disproving termination is based on this approach, and tries to make the constraints of constrained rewrite rules weaker in order to ensure non-termination.

1. Introduction

Recently, theorem proving on constrained term rewriting systems (constrained TRSs, for short) is investigated in several ways [Bou95, Arm02, Bou05, Fal06, Fur08, Bou08, Sak09]. In theorem proving, termination of initial constrained systems has to be proved in advance. In the case of constrained TRSs obtained from imperative programs with “while” loops, it is hard to prove termination by using reduction orders because any path-based order (such as LPO and RPO) usually employed is hardly helpful in proving termination of the systems. Moreover, reduction orders used in the expansion operation of theorem proving must be consistent with the reduction of the initial systems that should be terminating 1, and must be specified by users at the beginning of theorem proving processes. Since any

1998 ACM Subject Classification: F. Theory of Computation.
Key words and phrases: termination, non-termination, constrained rewrite system.
This work is partly supported by MEXT. KAKENHI #18500011, #20300010 and #20500008, and Kayamori Foundation of Informational Science Advancement.

1For the initial system R, the reduction orders \( \succ \) must satisfies \( \xrightarrow{R} \subseteq \succ \).
of local variables. For a positive integer \( n \), \( f(n) \) computes \( 2^{2^n} \). As \( f(3) \) returns 65536 (= \( 2^{2^2} \)), so the term \( f(s^3(0)) \) is reduced by \( R_t \) to \( s^{65536}(0) \). The C program is terminating and the corresponding \( R_t \) is also terminating. On the other hand, the C program obtained by replacing \( x > i \) by \( x \neq i \) is still terminating but the corresponding TRS \( R_{\text{sn}} \) obtained from \( R_t \) by replacing \( x > i \) by \( x > i \lor i > x \) is not terminating; e.g., \( u(0, s(0), 0) \) is not terminating w.r.t. the new TRS \( R_{\text{sn}} \). Note that termination of either the C program in Figure 1 or the corresponding integer rewriting system [Fuh09] of \( R_t \) could not be proved by AProVE [Gie06].
As a termination proof technique for unconstrained term rewriting systems (TRSs, for short), the dependency pair framework (the DP framework, for short) is well studied [Art00, Hir03, Hir07, Gie04, Gie06, Gie07, Fuh08, Fuh09]. The DP framework has been extended for the class of constrained equational systems (CESs, for short) that are rewrite systems with built-in numbers and semantic data structures [Fal08]. The extended DP framework has been adapted to the class of PA-based TRSs, that is a special subclass of CESs [Fal09].

In this paper, we propose methods for proving and disproving termination of constrained TRSs. The aim of this paper is to provide a termination prover for constrained TRSs, that works efficiently in theorem provers for constrained systems.

The method for proving termination of constrained TRSs is straightforwardly based on the DP framework [Art00, Gie04] for unconstrained TRSs. The method is applicable to constrained TRSs such that the semantics is over linear integer arithmetics, so-called Presburger arithmetics. We first show a termination criterion for constrained TRSs, following [Art00, Fal08]. Then, we propose the DP framework for constrained TRSs, following the extended DP framework [Fal08, Fal09] for CESs. Especially, we propose only a DP processor based on polynomial interpretations, adapting the corresponding processor in the extended DP framework for CESs to constrained TRSs. At the adaption, we introduce the mechanism for the lower bound detection [Gie07, Fuh08, Fuh09] into the DP processor while a simple variant of the mechanism is used in [Fal08, Fal09]. On the other hand, the DP processor proposed in this paper is partially restricted in contrast to the original one: any argument of marked symbols is filtered out by polynomial interpretations if the argument contains a non-interpreted symbol. Due to this restriction, unlike [Fal08, Fal09], the DP processor does not consider rewrite rules at all but instead the processor employs the semantics given for some function symbols. The reason of this restriction is to implement an efficient prover easily. Thus, the method in this paper is a restricted but simplified combination of the processors in [Gie07, Fuh08, Fuh09] and [Fal08, Fal09]. Nevertheless, we will show that the method can prove termination of the leading example $R_f$ in this section, while the other existing methods cannot prove the termination.

To propose methods for disproving termination, we first show a non-termination criterion of constrained TRSs. In the case of unconstrained TRSs, the non-termination criterion $t \xrightarrow{+} C[t\theta]$ is well known, and thus rewrite rules such that the right-hand sides are encompassments of the left-hand sides (i.e., rules of the form $t \rightarrow C[t\theta]$) cause non-termination. In contrast, constrained rewrite rules such that the right-hand sides are encompassments of the left-hand sides do not always cause non-termination. For example, the second rule of $R_f$ in Figure 1 does not cause non-termination. The approach to non-termination proofs is to extend the non-termination criterion $t \xrightarrow{+} C[t\theta]$ to the one for the constrained reduction, and to detect a constraint that ensures non-termination caused by $t \xrightarrow{\phi} C[t\theta]$. The constrained reduction under a constraint $\phi$ is a reduction where a constrained rule $l \rightarrow r \Leftarrow \psi$ can be applied to a term $l\sigma$ if $\phi \Rightarrow \sigma(\psi)$ is valid. We also formalize the criterion as a rule-based criterion. More precisely, for rewrite rules of the form $t \rightarrow C[t\theta] \Leftarrow \phi$, we characterize extra constraints that ensure non-termination caused by the rewrite rules. Furthermore, we show that such extra constraints sometimes can be obtained from the constraints $\phi$ of the rewrite rules, by weakening the constraints $\phi$. Finally, we propose a method to detect the extra constraints, that tries to make the constraints $\phi$ weaker. If the method succeeds in weakening, then it is proved that the rewrite rules cause non-termination. We will show that the method succeeds in disproving termination of the TRS $R_{sn}$ in this section.
Constrained systems in this class of this paper consist of left-hand terms, right-hand terms, and constraints, and the class can be regarded as a strict subclass of CESs [Fal08] (and PA-based TRSs [Fal09] if the semantics is over linear integer arithmetics). Nevertheless, it is worth of dealing with the class for the following reason; this class is often used in theorem proving [Bou05, Fur08, Bou08, Sak09]; methods for directly proving (and disproving) termination of the systems in this class are desired in order to implement theorem provers with efficient termination provers; for the sake of readability, we can discuss non-termination on a simple class of constrained systems. One of the differences from the class of CESs is that the signature for linear arithmetics over integers is not fixed while all the rules over the signature must be consistent with the semantics. We believe that the discussion in this paper holds for more complicated classes, such as CESs.

This paper is organized as follows. Section 2 prepares notations of term rewriting and constraints. Section 3 recalls constrained TRSs treated in this paper. Section 4 proposes a termination criterion for constrained TRSs and shows a DP processor based on polynomial interpretations. Section 5 proposes a non-termination criterion for constrained TRSs, that is a sufficient condition to disprove termination. Non-trivial proofs and experimental results are described in Appendix A and F, respectively.

2. Preliminaries

Here, we will review the following basic notations of term rewriting [Baa98, Ohl02], and first-order predicate logic [Hut00].

Throughout this paper, we use \( \mathcal{V} \) as a countably infinite set of variables. The set of terms over a signature \( \mathcal{F} \) and \( \mathcal{V} \) is denoted by \( T(\mathcal{F}, \mathcal{V}) \). The set of all variables appearing in terms \( t_1, \ldots, t_n \) is denoted by \( \text{Var}(t_1, \ldots, t_n) \). If \( f \) is a unary function symbol, then \( f^n(t) \) abbreviates the term \( f(f(\cdots f(t)\cdots)) \), the \( n \)-fold application of \( f \) to \( t \). The identity of terms \( s \) and \( t \) is denoted by \( s \equiv t \). For a term \( t \) and a position \( p \) of \( t \), the notation \( t|_p \) represents the subterm of \( t \) at \( p \). If position \( p \) is a proper prefix of a position \( q \), then we write \( p < q \). The function symbol at the root position \( \varepsilon \) of \( t \) is denoted by root\( (t) \). The notation \( C[t]_p \) represents the term obtained by replacing \( \square \) at position \( p \) of a context \( C[\square] \) with term \( t \). The domain and range of a substitution \( \sigma \) are denoted by \( \text{Dom}(\sigma) \) and \( \text{Ran}(\sigma) \), respectively. The application \( \sigma(t) \) of \( \sigma \) to \( t \) is abbreviated by \( t\sigma \). If \( \text{Dom}(\sigma) = \{ x_1, \ldots, x_n \} \), then we may write \( \{ x_i \mapsto \sigma(x_i) \mid 1 \leq i \leq n \} \) instead of \( \sigma \). The restriction \( \sigma|_X \) of \( \sigma \) to a set \( X \subseteq \mathcal{V} \) is defined as \( \sigma|_X = \{ x \mapsto \sigma(x) \mid x \in \text{Dom}(\sigma) \cap X \} \). Let \( T \subseteq T(\mathcal{F}, \mathcal{V}) \) and \( X \subseteq \mathcal{V} \). Then, a substitution \( \sigma \) is said to be a \( T \)-substitution for \( X \) if \( X \subseteq \text{Dom}(\sigma) \) and \( \text{Ran}(\sigma|_X) \subseteq T \).

Let \( \mathcal{G} \) be a signature such that \( \mathcal{G} \) has at least a constant (i.e., \( T(\mathcal{G}) \neq \emptyset \)), and \( \mathcal{P} \) be a finite set of predicate symbols. Formulas over \( (\mathcal{G}, \mathcal{P}, \mathcal{V}) \) have the following syntax given in Backus Naur form: \( \phi ::= P(t_1, \ldots, t_n) \mid T \mid \bot \mid (\neg \phi) \mid (\phi \land \phi) \mid (\phi \lor \phi) \mid (\forall x. \phi) \mid (\exists x. \phi) \) where \( P \) is an \( n \)-ary predicate in \( \mathcal{P} \), \( t_1, \ldots, t_n \in T(\mathcal{G}, \mathcal{V}) \), and \( x \in \mathcal{V} \). We sometimes denote \( (\neg \phi_1 \lor \phi_2) \) by \( (\phi_1 \Rightarrow \phi_2) \) as usual. We often abbreviate brackets in formulas as usual. The set of free variables in a formula \( \phi \) is denoted by \( \text{fv}(\phi) \). A formula \( \phi \) is called closed if \( \text{fv}(\phi) = \emptyset \). Substitutions from \( \mathcal{V} \) to \( T(\mathcal{G}, \mathcal{V}) \) are applied to formulas as usual.

A structure \( \mathcal{M} \) for (formulas over) \( (\mathcal{G}, \mathcal{P}) \) is a triple \( (A, \mathcal{G}^\mathcal{M}, \mathcal{P}^\mathcal{M}) \) such that a universe \( A \) is a non-empty set of concrete values, \( \mathcal{G}^\mathcal{M} \) is a set of mappings where \( g^\mathcal{M} : A^n \to A \) for

\[ \text{om} \]
each $n$-ary function symbol $g \in \mathcal{G}$ is included in $\mathcal{G}^M$, and $\mathcal{P}^M$ is a family of sets where $P^M \subseteq A^n$ for each $n$-ary predicate symbol $P \in \mathcal{P}$ is included in $\mathcal{P}^M$. The interpretation of closed formulas by $M$ is defined as follows: $(g(t_1, \ldots, t_n))^M = g^M(t_1)^M, \ldots, (t_n)^M$ where $g \in \mathcal{G}$, and a closed formula $\phi$ computes to $\top$ in the structure $M$, written as $M \models \phi$, if $\phi$ holds w.r.t. $M$ as usual. Function symbols in $\mathcal{G}$ are called interpretable (or predefined) by $M$. We suppose that for every element $a \in A$, there exists a term $t \in T(\mathcal{G})$ such that $(t)^M = a$. A formula $\phi$ is called valid (satisfiable, resp.) w.r.t. $M$ if $M \models \theta(\phi)$ for every (some, resp.) $T(\mathcal{G})$-substitution $\theta$ for $\text{fv}(\phi)$. When $\mathcal{G}$, $\mathcal{P}$, and $M$ are fixed in context, we may call formulas over $(\mathcal{G}, \mathcal{P}, \mathcal{V})$ constraints (w.r.t. $M$). We suppose that the binary predicate symbol $\simeq$ is included in $\mathcal{P}$ and its interpretation $\simeq^M$ is the equality $= \overline{\text{over } A}$.

Example 2.1. Let a signature $\mathcal{G}_{PA} = \{0, s(), p(), \text{plus}(), \text{minus}()\}$, a predicate set $\mathcal{P}_{PA} = \{\succ, \simeq\}$, and a structure $M_{PA} = (\mathbb{Z}, \{0^M_{PA}, s^M_{PA}, p^M_{PA}, \text{plus}^M_{PA}, \text{minus}^M_{PA}\}, \{\succ^M_{PA}, \simeq^M_{PA}\})$ for $(\mathcal{G}_{PA}, \mathcal{P}_{PA})$ where $\mathbb{Z}$ is the set of integers, $0^M_{PA} = 0$, $s^M_{PA}(x) = x + 1$, $p^M_{PA}(x) = x - 1$, $\text{plus}^M_{PA}(x_1, x_2) = x_1 + x_2$, $\text{minus}^M_{PA}(x_1, x_2) = x_1 - x_2$, and $\simeq^M_{PA} = \{(n, m) \mid n > m\}$. Constraints w.r.t. $M_{PA}$ corresponding to linear arithmetics over integers, so-called Presburger arithmetics.

3. Constrained Term Rewriting Systems

In this section, we recall the definition of constrained term rewriting systems [Bou08, Sak09], and properties shown in [Sak09].

Let $\mathcal{G}$ and $\mathcal{F}$ be signatures such that $\mathcal{F} \cap \mathcal{G} = \emptyset$ and $\mathcal{G}$ contains at least a constant, $\mathcal{P}$ be a finite set of predicate symbols, and $M$ be a structure for $(\mathcal{G}, \mathcal{P})$. A constrained rewrite rule over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, M)$ is a triple $(l, r, \phi)$, written as $l \rightarrow r \leftarrow \phi$, such that $l$ and $r$ are terms in $T(\mathcal{F} \cup \mathcal{G}, \mathcal{V})$, $l$ is not a variable, $\phi$ is a quantifier-free constraint w.r.t. $M$ (i.e., a formula over $(\mathcal{G}, \mathcal{P}, \mathcal{V})$), and $\text{Var}(l) \supseteq \text{Var}(r) \cup \text{fv}(\phi)$, and $\phi$ is satisfiable w.r.t. $M$. We may write $l \rightarrow r$ instead of $l \rightarrow r \leftarrow \top$.

Let $R$ be a finite set of constrained rewrite rules over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, M)$. The rewrite relation $\rightarrow_R$ is defined as follows: $\rightarrow_R = \{(C[l[\sigma]_p, C[r[\sigma]_p]) \mid l \rightarrow r \leftarrow \phi \in R, \sigma \text{ is a } T(\mathcal{G})\text{-substitution for } \text{fv}(\phi), M \models \phi(\sigma)\}$. To specify explicitly positions $p$ of redexes, we may write $\rightarrow_R^p$ instead of $\rightarrow_R$. We may write $\rightarrow_R^\leq$ if $\varepsilon < p$. A constrained term rewriting system (constrained TRS) over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, M)$ is an abstract reduction system $(T(\mathcal{F} \cup \mathcal{G}, \mathcal{V}), \rightarrow_R)$ and we simply write $R$ instead. Note that $R$ is a term rewriting system (TRS) if the constraints of all rewrite rules in $R$ are $\top$.

Example 3.1. The constrained TRS $R_f$ in Figure 1 is over $(\{\text{f}(), \text{u}(), \text{times}()\}, \mathcal{G}_{PA}, \mathcal{P}_{PA}, \mathcal{V}, M_{PA})$ where $\mathcal{G}_{PA}, \mathcal{P}_{PA}$ and $M_{PA}$ are shown in Example 2.1.

Next, we introduce a property between reduction and interpretation.

Definition 3.2 ([Sak09]). Let $R$ be a constrained TRS over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, M)$. We say that $R$ is locally sound for $M$ if for every ground terms $s \in T(\mathcal{G})$ and $t \in T(\mathcal{F} \cup \mathcal{G})$, $s \rightarrow_R t$ implies that $t \in T(\mathcal{G})$ and $M \models s \simeq t$ (i.e., $(s)^M = (t)^M$).

For example, $R_f$ in Figure 1 is locally sound for $M_{PA}$. Note that $R$ is locally sound for $M$ if $r \in T(\mathcal{G}, \mathcal{V})$ and $\phi \Rightarrow l \simeq r$ is valid w.r.t. $M$ for every rule $l \rightarrow r \leftarrow \phi \in R$ such that $l \in T(\mathcal{G}, \mathcal{V})$ [Sak09].
The local soundness for structures is one of the most important properties for the theorem proving framework in [Fur08, Sak09], and it means that for any redex which can be interpreted to a concrete value, all terms reducible from the redex can be interpreted to the same value of the redex. In other words, local soundness guarantees that the interpretation is preserved by every rewrite step. Assuming constrained TRSs to have this property is not restrictive because this property holds in many cases.

There are two prominent types of constrained systems, from the viewpoint of semantics for constraints. One is a framework of conditional TRSs whose conditional parts representing constraints are solved by equational theories specified by the framework or users [Bou95, Arm02]. The other is a framework such that built-in semantics for predicates and certain function symbols is given and constraint solvers for checking validity or satisfiability are maintained by the framework or by users [Toy87, Dur08, Bou05, Fal08, Fur08, Bou08, Sak09, Fal09] 3. The former can be used even when semantics of constraints is not fixed. The latter is used when the semantics is fixed. Note that in the latter framework one can use theories as semantics.

4. Proving Termination of Constrained TRSs over Integer Arithmetics

In this section, we extend the DP framework [Art00, Gie04] to the one for constrained TRSs, by adapting the extended DP framework [Fal08, Fal09] for CESs to constrained TRSs, and combining with the mechanism [Gie07, Fuh08, Fuh09] for the lower bound detection of integer sequences.

4.1. Termination Criterion for Constrained TRSs

We first characterize the termination criterion for constrained TRSs, following [Art00, Gie04, Fal08, Fal09]. Let R be a constrained TRS over \((F, G, P, V, M)\). The set of defined symbols of R is denoted by \(\mathcal{D}\), i.e., \(\mathcal{D} = \{\text{root}(l) \mid l \rightarrow r \Leftarrow \phi \in R\}\). The marked (tuple) symbol of a defined symbol \(f \in \mathcal{D}\) is denoted by \(f^\sharp\). The set of marked symbols is denoted by \(\mathcal{D}^\sharp\). For a term \(f(t_1, \ldots, t_n)\) with \(f \in \mathcal{D}\), \(f(t_1, \ldots, t_n)^\sharp\) denotes the term \(f^\sharp(t_1, \ldots, t_n)\).

**Definition 4.1** (dependency pair). Let R be a constrained TRS over \((F, G, P, V, M)\). A (constrained) dependency pair for \(l \rightarrow r \Leftarrow \phi \in R\) is a triple \((l^\sharp, t^\sharp, \phi)\) where \(t\) is a subterm of \(r\) such that \(\text{root}(t) \in \mathcal{D}\) and \(t\) is not a proper subterm of \(l\). The set of dependency pairs of \(R\) is denoted by \(\mathcal{DP}(R)\).

**Example 4.2.** The dependency pairs of \(R_f\) in Figure 1 are illustrated in Figure 2.

To facilitate visualization of chains, we may write \(s \xrightarrow{\phi} t\) instead of \(s \rightarrow t \Leftarrow \phi\).

**Definition 4.3** (chain). Let \(R\) and \(S\) be constrained TRSs over \((F, G, P, V, M)\) and \((F', G', P, V, M)\), respectively. A (possibly infinite) sequence \(s_1 \xrightarrow{\sigma_1} t_1 \ xrightarrow{\sigma_2} t_2 \ xrightarrow{\sigma_3} \cdots\) of constrained rules in \(S\) is called an \((S, R)\)-chain if there exist substitutions \(\sigma_1, \sigma_2, \ldots\) such that \(t_i^\sigma_i \xrightarrow{r \Leftarrow} s_{i+1}^\phi\) and \(M \models \phi_i(\sigma_i)\) for all \(i \geq 1\). Note that \(M \models \phi_k(\sigma_k)\) is necessary when the length of the sequence is \(k\). The chain is called minimal if every \(t_i^\sigma_i\) is terminating w.r.t. \(R\), and called strongly infinite if every element in \(S\) appears infinitely many times in the chain.

3The framework restricts term sets appearing in membership conditional systems [Toy87] to sets independent on the notion of normal forms.
in DG \((E \text{TRS})\) estimation methods for unconstrained systems, i.e., any estimated graph of the graphs of unconstrained TRSs. For this reason, we usually use over-approximated use all techniques for constructing estimated dependency graphs of unconstrained TRSs.

\[
\text{Proof. Another approach to the proof is found in Appendix A.}
\]

Like in the case of unconstrained TRSs, the non-existence of infinite chains guarantees termination of constrained TRSs.

**Theorem 4.4.** Let \(R\) be a constrained TRS over \((F, G, P, V, M)\). \(R\) is terminating iff there exists no infinite minimal \((DP(R), R)\)-chain.

\[
\text{Proof. The proof of this theorem follows from the proof of the original DP theorem [Art00]. Another approach to the proof is found in Appendix A.}
\]

Next, we extend the notion of dependency graphs for constrained TRSs.

**Definition 4.5** (dependency graph). The dependency graph of a constrained TRS \(R\) over \((F, G, P, V, M)\), written as \(DG(R)\), is a directed graph \((V, E)\) such that \(V = DP(R)\), and \(E = \{(v, u) \mid v, u \in V, \text{ the sequence } "v \ u" \text{ is a } (DP(R), R)\)-chain\}.

The node sets of strongly connected subgraphs in \(DG(R)\) are called (dependency) cycles in \(DG(R)\) [Art00]. To prove termination, it suffices to prove non-“strong-infinity” for each cycle.

The dependency graphs of constrained TRSs are not computable in general as well as the graphs of unconstrained TRSs. For this reason, we usually use over-approximated graphs instead of the complete graphs. Since the TRS \(R_u\) obtained from a constrained TRS \(R\) by removing constraints is an approximation of \(R\) (i.e., \(\rightarrow_R \subseteq \rightarrow_{R_u}\)), we can use all techniques for constructing estimated dependency graphs of unconstrained TRSs. Therefore, the estimated dependency graphs of constrained TRSs can be obtained by the estimation methods for unconstrained systems, i.e., any estimated graph of \(DG(R_u)\) is an estimated graph of \(DG(R)\). An improvement for computing the estimated graph of \(DG(R)\) is found in Appendix E.
The non-existence of infinite minimal chains is shown via the non-existence of strongly infinite minimal chains on cycles.

**Theorem 4.6.** Let $R$ be a constrained TRS over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{M})$.

1. There is no infinite minimal $(\text{DP}(R), R)$-chain iff there is no strongly infinite minimal $(S, R)$-chain for any cycle $S$ in $\text{DG}(R)$.
2. For any $S \subseteq \text{DP}(R)$, there is no infinite minimal $(S, R)$-chain if and only if there is no strongly infinite minimal $(S, R)$-chain and there is no infinite minimal $(S', R)$-chain for any proper subset $S' \subset S$.

**Proof.** This theorem follows from Theorem 4.4 and the definition of chains and $\text{DG}(R)$. ■

**Example 4.7.** The dependency graph of $R_f$ in Figure 1 is illustrated in Figure 3. The SCCs in $\text{DG}(R_f)$ are $\{(1), (4), (7)\}$, $\{(9)\}$, $\{(11)\}$, $\{(13)\}$, $\{(15)\}$, $\{(19)\}$. To prove termination of $R_f$, it is enough to prove the non-existence of infinite minimal chains for all of them.

The DP framework in [Gie04] is an efficient approach to termination proofs of rewrite systems, that is extended in several ways and extended for several classes of rewrite systems. In the framework, DP processors play an important role for efficient performance.

We adapt the DP framework to constrained TRSs. Let $R$ be a constrained TRS, and $S \subseteq \text{DP}(R)$. A pair $(S,R)$ is called a DP problem. Note that this paper only considers minimal chains. A DP problem $(S,R)$ is called finite if there is no infinite minimal $(S,R)$-chain, and called infinite if either it is not finite or $R$ is not terminating. A DP processor is a function $\text{Proc}$ that takes a DP problem as input and returns either a finite set of DP problems or “no”. A DP processor $\text{Proc}$ is called sound if for any DP problem $(S,R)$, the problem $(S,R)$ is finite whenever all DP problems in $\text{Proc}((S,R))$ are finite. To prove termination of a constrained TRS $R$, sound DP processors are recursively applied to the initial DP problem $(\text{DP}(R), R)$. If all the resulting problems have been reduced to DP problems of the form $(\emptyset, R')$, then the termination has been proved.

The DP processor based on dependency graphs can be used directly for constrained TRSs as well as other extensions, because the processor divides each set of dependency pairs into SCCs of the graph. In addition, the DP processor based on the subterm criterion [Hir07] can be used for constrained TRSs, ignoring constraints of dependency pairs. These two DP processors are sound for constrained TRSs. The detail of the processor based on the subterm criterion is found in Appendix C.

**Example 4.8.** Consider the dependency graph $\text{DG}(R_f)$ in Figure 3. The initial DP problem $(\text{DP}(R_f), R_f)$ is decomposed by the DP processor based on dependency graphs into the set of the DP problems $\{(1), (4), (7), R_f\}$, $\{(9), R_f\}$, $\{(11), R_f\}$, $\{(13), R_f\}$, $\{(14), R_f\}$, $\{(17), R_f\}$, and $\{(19), R_f\}$. Moreover, the DP processor based on the subterm criterion transforms all the problems $\{(9), R_f\}$, $\{(11), R_f\}$, $\{(13), R_f\}$, $\{(14), R_f\}$, $\{(17), R_f\}$, $\{(19), R_f\}$.

![Figure 3: the dependency graph of $R_f$.](image-url)
$\{(19), R_t\}$ into $\{(\emptyset, R_t)\}$. Therefore, the remaining problem to prove termination of $R_t$ is $\{(1), (4), (7), R_t\}$.

### 4.2. DP Processor based on Polynomial Interpretations over Integers

It is difficult to show the non-existence of strongly infinite minimal chains on cycles that contain some kind of pairs representing “while” loops, such as the SCC $\{(1), (4), (7)\}$ in $DG(R_t)$. This is because the size of terms is increasing while a kind of distance between such terms and the point to exit from the loop is decreasing. For this reason, we have to use constraint information in the dependency pairs. Such a use of information is achieved by the DP processor for CESs [Fal08, Fal09]. Moreover, to use polynomial interpretations over integers, the idea of introducing the lower bound detection has been proposed and used [Gei07, Fuh08, Fuh09].

In this subsection, combining the ideas in [Gei07, Fuh08, Fuh09] and [Fal08, Fal09], we propose a restricted but efficient DP processor for constrained TRSs, that is based on polynomial interpretations over integers. Any argument of marked symbols is filtered out by polynomial interpretations if the argument contains a non-interpretable symbol. Due to this restriction, the processor needs not to check decreasingness of rewrite rules, and hence efficient. Nevertheless, this processor is sometimes useful in proving termination of constrained TRSs obtained from imperative programs, whose termination cannot be proved by the original techniques. Detailed differences from the existing techniques will be shown after the mechanism is described.

Throughout this subsection, we restrict constrained TRSs to be over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{M})$ such that the universe of $\mathcal{M}$ is the set $\mathbb{Z}$ of integers and the set of constraints corresponds to linear arithmetics over integers, such as constraints w.r.t. $\mathcal{M}_{\mathbb{Z}_{PA}}$. We denote such a structure $\mathcal{M}$ by $\mathcal{M}_{\mathbb{Z}}$, and the corresponding sets $\mathcal{G}$ and $\mathcal{P}$ of function symbols and predicate symbols by $\mathcal{G}_{\mathbb{Z}}$ and $\mathcal{P}_{\mathbb{Z}}$, respectively.

The basic idea of the DP processor is to reduce an infinite derivation obtained from a strongly infinite minimal chain into an infinite sequence of terms in $T(\mathcal{G}_{\mathbb{Z}})$, and then to transform the sequence into an infinite strictly-decreasing sequence of integers, that has a lower bound (i.e., causes a contradiction). The first transformation is done by restricted polynomial interpretations via tree homomorphisms, and the second one is done by the semantics that the constrained TRSs already have. The reason why the polynomial interpretations are performed by tree homomorphisms to $T(\mathcal{G}_{\mathbb{Z}}, \mathcal{V})$ is that constraints in dependency pairs are combined with the constraints obtained from inequalities of the left-hand and right-hand sides. For example, given some interpretation $I$ of terms and an order $\succ$, the constrained pair $s \rightarrow t \iff \phi$ provides the condition “$\phi$ implies $I(l) \succ I(r)$” to be valid in order to prove termination. If $I(l)$ and $I(r)$ are terms over the same signature of $\phi$, then the validity of the condition can be reduced to the validity of the constraint $\phi \Rightarrow I(l) \succ I(r)$.

**Definition 4.9** (tree homomorphism [Com07]). Let $\mathcal{F}_1$ and $\mathcal{F}_2$ be signatures, and $F \subseteq \mathcal{F}_1$ such that $\mathcal{F}_2 \supseteq \mathcal{F}_1 \setminus F$. A tree homomorphism $\mathcal{H}$ for $F$ is a mapping from $F$ to $T(\mathcal{F}_2, \mathcal{V})$ such that $\mathcal{H}(f) \in T(\mathcal{F}_2, \{x_1, \ldots, x_n\})$ for any $n$-ary function symbol $f \in F$. The application of $\mathcal{H}$ to terms in $T(\mathcal{F}_1, \mathcal{V})$ is defined as follows: $\mathcal{H}(x) = x$ for $x \in \mathcal{V}$, $\mathcal{H}(g(t_1, \ldots, t_n)) = g(\mathcal{H}(t_1), \ldots, \mathcal{H}(t_n))$ for $g \notin F$ (i.e., $g \in \mathcal{F}_1 \setminus F$), and $\mathcal{H}(f(t_1, \ldots, t_n)) = \mathcal{H}(f)\{x_i \rightarrow \mathcal{H}(t_i) \mid 1 \leq i \leq n\}$ for $f \in \mathcal{F}_1$. 
Note that for a tree homomorphism $H$ for $D^x$, we have $H(f^2(t_1, \ldots, t_n)) = H(f^2)\{x_i \mapsto t_i \mid 1 \leq i \leq n\}$.

**Example 4.10.** Consider the term $u^2(x, s(i), \text{times}(z, f(i)))$ in Figure 2. Let $H$ be a tree homomorphism for $\{f^2, u^2\}$ such that $H(f^2) = x_1$ and $H(u^2) = \text{minus}(x_1, x_2)$. Then, we have $H(u^2(x, s(i), \text{times}(z, f(i)))) = \text{minus}(x_1, x_2)\{x_1 \mapsto H(x), x_2 \mapsto H(s(i))\} \equiv \text{minus}(x_1, x_2)\{x_1 \mapsto x, x_2 \mapsto s(i)\} \equiv \text{minus}(x, s(i))$.

We do not have to give any tree homomorphism for symbols in $G$ in order to keep the preservation of the interpretation for terms in $T(G)$ due to local soundness, i.e., to keep $(H(s))^M = (H(t))^M$ for all $s$ and $t \in T(G)$ such that $s \xrightarrow{R} t$. The polynomial interpretation can filter out some arguments as well as argument filterings [Art00, Kus99]. For example, the $i$-th argument of $f^2$ is filtered out by $H$ if $H(f^2)$ does not contain $x_i$.

For an $n$-ary marked symbol $f^2$, we only use linear arithmetic expressions $k_0 + k_1 x_1 + k_2 x_2 + \cdots + k_n x_n$ over integers as $H(f^2)$, that can be encoded as terms in $T(G, \{x_1, \ldots, x_n\})$. In this setting, $H$ can be considered as restricted polynomial interpretations [Man70, Baa98, Hir07, Gie07] over integers because $H$ interprets $t^2$ into a term in $T(G, x)$ that represents linear polynomials over integers. Such tree homomorphisms are not monotone in general, but we do not restrict them to be monotone because we force tree homomorphisms on a condition described later. Thus, we have no restriction on the use of negative integers as coefficients in polynomials, as well as [Hir07, Gie07, Fal08, Fuh09, Fal09].

Next, we give a sufficient condition under which chains are interpreted into integer sequences that have lower bounds. Let $R$ be a constrained TRS over $(F, G, \mathcal{P}_Z, V, M_Z)$, $S \subseteq DP(R)$, $\succ \in \mathcal{P}_Z$, and $\succ_{M_Z}$ be $\succ$ over integers. The condition is the existence of a tree homomorphism $H$ for $D^x$ satisfying all of the following conditions:

**A1.** $H(s) \in T(G_Z, V)$, $H(t) \in T(G_Z, V)$ and $\text{Var}(H(t)) \subseteq \text{Var}(H(s)) \cup \text{fv}(\phi)$ for all pairs $s \rightarrow t \iff \phi \in S$,

**A2.** $\text{Var}(H(t)) \subseteq \text{fv}(\phi)$ for some pair $s \rightarrow t \iff \phi \in S$,

**B1.** $\phi \Rightarrow H(s) \succeq H(t)$ is valid w.r.t. $M_Z$ for all pairs $s \rightarrow t \iff \phi \in S$,

**B2.** $\phi \Rightarrow H(s) \succeq H(t)$ is valid w.r.t. $M_Z$ for some pair $s \rightarrow t \iff \phi \in S$, and

**C.** there exist a term $m \in T(G_z)$ and a pair $u \mapsto f^2(t_1, \ldots, t_n) \iff \phi \in S$ such that for all renamed pairs $f^2(s_1, \ldots, s_n) \mapsto v \iff \psi \in S$ with $\text{Var}(u) \cap \text{Var}(s_1, \ldots, s_n) = 0$, 

$$\left(\phi \land \psi \land \left(\land_{i \in \{j \mid s_j, t_j \in T(G_z, V), \text{Var}(t_j) \subseteq \text{Var}(H(u)), \text{fv}(\phi)\}} t_i \simeq s_i\right)\right) \Rightarrow H(u) \succeq m$$

is valid w.r.t. $M_Z$.

We explain what roles the conditions A1, A2, B1, B2, and C play.

- **A1** and **A2**, original ones of this paper, are used to transform derivations obtained from chains into sequences of terms in $T(G_Z)$. **A1** guarantees that $H(s\sigma) \in T(G_Z)$ implies $H(t\sigma) \in T(G_Z)$ for any $s \rightarrow t \iff \phi \in S$ and any substitution $\sigma$. **A2** guarantees the existence of pairs $s \rightarrow t \iff \phi \in S$ among the derivations such that $H(t\sigma) \in T(G_Z)$ for any substitution $\sigma$ such that $M_Z \models \sigma(\phi)$. In other words, **A1** and **A2** make $H$ filter out any argument of marked symbols if the argument contains a symbol in $F$. For this reason, unlike the usual processors based on reduction pairs or polynomial interpretations, $R$ is not considered. Thus, usable rules are not discussed in this paper.

- **B1** and **B2** are following the usual conditions with considering constraints [Gie07, Fuh08, Fuh09, Fal08, Fal09].

\[4a \succeq b \text{ denotes } a \succ b \lor a \simeq b.\]
Consider the following infinite derivation obtained from a strongly infinite minimal chain:

\[ s_1 \sigma_1 \rightarrow^* s_1 \rightarrow^* s_1 \rightarrow^* s_1 \rightarrow^* \cdots \]

Suppose that \( s_j \rightarrow t_j \Leftarrow \phi_j \) satisfies \( A_2 \), i.e., \( \text{Var}(\mathcal{H}(t_j)) \subseteq \text{fv}(\phi_j) \). Then, the above derivation is transformed by the satisfaction of \( A_1 \) into the following infinite sequence of terms in \( T(\mathcal{G}_Z) \):

\[ \mathcal{H}(t_j \sigma_j) \rightarrow^* \mathcal{H}(s_j \sigma_{j+1}), \mathcal{H}(t_{j+1} \sigma_{j+1}) \rightarrow^* \mathcal{H}(s_{j+2} \sigma_{j+2}), \mathcal{H}(t_{j+2} \sigma_{j+2}) \rightarrow^* \cdots \]

where possibly \( \mathcal{H}(t_{j-1} \sigma_{j-1}) \notin T(\mathcal{G}_Z) \). Then, due to the semantics and the satisfaction of \( B_1 \), we obtain the following infinite sequence of integers:

\[ n_j (> \cup =) n_{j+1} (> \cup =) n_{j+1} (> \cup =) \cdots \]

where \( > \) appears infinitely many times by satisfying \( B_2 \). Due to the satisfaction of \( C \), there exists a lower bound \( m \) such that \( n_i > m \) for all \( i \geq j \). This contradicts the fact that there is no integer sequence \( m_1 > m_2 > \cdots \) such that \( m_i > m \) for all \( i \geq 0 \), i.e., the existence of strongly infinite minimal chains. This observation is summarized as the following theorem.

**Theorem 4.11.** Let \( R \) be a constrained TRS over \((\mathcal{F}, \mathcal{G}_Z, \mathcal{P}_Z, \mathcal{V}, \mathcal{M}_Z)\) that is locally sound for \( \mathcal{M}_Z \), \( S \subseteq DP(R), > \in \mathcal{P}_Z \), and \( >^\mathcal{M}_Z \) be \( > \) over integers. Let \( \mathcal{H} \) be a tree homomorphism for \( \mathcal{D}^\mathbb{Z} \) such that \( A_1, A_2, B_1, B_2 \) and \( C \) are satisfied. Then, there is no strongly infinite minimal \((S, R)\)-chain.

**Example 4.12.** Consider the SCC \{\(1\), \(4\), \(7\)\} in Figure 3. Let \( \mathcal{H} \) be a tree homomorphism such that \( \mathcal{H}(f^2) = \mathcal{H}(u^2) = x_1 \). Then, \( \mathcal{H} \) satisfies \( A_1, A_2, B_1 \) and \( B_2 \). The pair \( 1 \) has the direct successor nodes \( 4 \) and \( 7 \) of the form \( u(x', i', z') \rightarrow v \Leftarrow x' > i' \). In addition, \( (\top \land x' > i' \land x \simeq x' \land i' \simeq 0 \land z' \simeq s(0)) \Rightarrow x > m \) is valid w.r.t. \( \mathcal{M}_P \) if \( m \equiv 0 \). Therefore, \( \mathcal{H} \) satisfies \( C \) and then there is no strongly infinite minimal \{\(1\), \(4\), \(7\)\}, \( R_f \)-chain.

In contrast to [Gie07, Fuh08, Fuh09], we can use constraint solvers for linear arithmetics over integers in order to solve the constraints obtained from either \( B_1, B_2 \) and \( C \), as well as [Fal08, Fal09].

By removing dependency pairs satisfying \( A_2, B_2 \) and \( C \) separately from the DP problem \((S, R)\), Theorem 4.11 can be adapted to the DP processor based on polynomial interpretations over integers, as well as [Gie07, Fuh08, Fal09, Fal08, Fal09].

**Definition 4.13** (DP processor based on polynomial interpretations over integers). Let \( R \) be a constrained TRS over \((\mathcal{F}, \mathcal{G}_Z, \mathcal{P}_Z, \mathcal{V}, \mathcal{M}_Z)\) that is locally sound for \( \mathcal{M}_Z \), \( S \subseteq DP(R) \), \( > \in \mathcal{P} \), and \( >^\mathcal{M}_Z \) be \( > \) over integers. Let \( \mathcal{H} \) be a tree homomorphism for \( \mathcal{D}^\mathbb{Z} \) such that \( A_1, A_2, B_1, B_2 \) and \( C \) are satisfied. Let

- \( S_{A_2} = \{ s \rightarrow t \Leftarrow \phi \in S \mid \text{Var}(\mathcal{H}(t)) \subseteq \text{fv}(\phi) \} \),
- \( S_{B_2} = \{ s \rightarrow t \Leftarrow \phi \in S \mid \phi \Rightarrow \mathcal{H}(s) > \mathcal{H}(t) \) is valid for \( \mathcal{M}_Z \} \), and
- \( S_C = \{ u \rightarrow f^2(t_1, \ldots, t_n) \Leftarrow \phi \in S \mid u \rightarrow f^2(t_1, \ldots, t_n) \Leftarrow \phi \) is a rule in \( C \} \).

A DP processor \( \text{Proc} \) is defined as follows:

\[
\text{Proc}(\langle S, R \rangle) = \begin{cases} 
\{ (S \setminus S_{A_2}, R), (S \setminus S_{B_2}, R), (S \setminus S_C, R) \} & \text{if either } S_{A_2}, S_{B_2} \text{ or } S_C \text{ is not empty} \\
\{ (S, R) \} & \text{otherwise}
\end{cases}
\]
Theorem 4.14. The DP processor in Definition 4.13 is sound.

Example 4.15. Consider the remaining DP problem \( \{ (1), (4), (7) \}, R_f \) in Example 4.7 again. Following Example 4.12, given this problem, the DP processor in Definition 4.13 returns \( \{ ((4), R_f), ((1), (4), R_f), ((4), (7), R_f) \} \). Applying the DP processor based on dependency graphs to \( \{ ((1), (4)), R_f \} \) and \( \{ ((4), (7)), R_f \} \), these problems are reduced to \( (\emptyset, R_f) \). Given \( ((4), R_f) \), the DP processor in Definition 4.13 returns \( (\emptyset, R_f) \), using a tree homomorphism \( \mathcal{H} \) such that \( \mathcal{H}(u^2) = x_1 - x_2 \). Therefore, all the resulting DP problems for the initial DP problem \( DP(R_f, R_f) \) are reduced to \( (\emptyset, R_f) \) and then \( R_f \) is terminating.

The method in this paper works efficiently for proving termination of our target constrained TRSs obtained from imperative programs over integers, while it is limited in scope of target. Therefore, the implementation based on the method in will be useful for theorem proving based on [Sak09].

Since any non-interpretable argument of marked symbols is filtered out, the method in this paper cannot prove termination of the constrained TRS obtained from \( C \) for constrained TRSs because variables in \( R \) then \( R \) is not terminating. Therefore, we need to develop a non-termination criterion in which constraints are valid. Therefore, we need to develop a non-termination criterion in which constraints are considered.

To consider constraints with reductions, we review the reduction under constraints. Let \( R \) be a constrained TRS. If there exists a reduction \( t \vdash_{\mathcal{M}} C[t\theta] \), then \( R \) is not terminating.

A simple technique to detect the reduction \( t \vdash C[t\theta] \) is to find a rewrite rule (or a dependency pair) of the form \( t \to C[t\sigma] \). Thus, in the process of the DP framework for unconstrained TRSs, we can disprove termination if such a dependency pair is detected. In contrast, this approach is not applicable to the case of constrained TRSs. For example, \( R_f \) in Figure 1 is terminating but it has a rule of the form \( t \to C[t\sigma] \), \( \phi \) such that \( \phi \) is not valid. Therefore, we need to develop a non-termination criterion in which constraints are considered.

To consider constraints with reductions, we review the reduction under constraints. Let \( R \) be a constrained TRS over \( \mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{M} \) and \( \phi \) be a constraint that is satisfiable w.r.t. \( \mathcal{M} \). The constrained reduction \( \vdash_{\phi} \) under \( \phi \) is defined as follows [Fur08]:

\[
\vdash_{\phi} R = \{ (C[l\sigma], C[r\sigma]) \mid l \to r \rightleftharpoons \psi \in R, \ \phi \Rightarrow \sigma(\psi) \text{ is valid w.r.t. } \mathcal{M} \}.
\]

It is clear that \( s \vdash_R \) implies \( s\theta \vdash_{\phi} t\theta \) for any substitution \( \theta \) such that \( \mathcal{M} \models \theta(\phi) \). Thus, for non-termination of \( \vdash_R \), it is enough to prove non-termination of \( \vdash_{\phi} R \) for some \( \phi \). More precisely, if there is a reduction \( t \vdash_{\phi} R C[t\theta] \) such that \( t\theta \vdash_{\theta(\phi)} R C[t\theta][\theta] \), \( t\theta \vdash_{\theta(\phi)} R (C[t\theta][\theta]) \theta \), \( \cdots \), then \( R \) is not terminating. Thus, the validity of \( \phi \Rightarrow \theta(\phi) \) is a sufficient condition of non-termination caused by \( t \vdash_{\phi} R C[t\theta] \).
Example 5.2. Consider the constrained TRS $R_{\text{an}}$ obtained from $R_f$ by replacing $x \times i$ by $x \rightarrow i \vee i \rightarrow x$ again: $R_{\text{an}} = \{ \ldots, u(x, i, z) \rightarrow u(x, s(i), \text{times}(z, f(i))) \iff x \rightarrow i \vee i \rightarrow x, \ldots \}$. Let $t$ be $u(x, i, z)$, $C[\!]$ be $\square$, $\phi$ be $i \rightarrow x$, and $\theta$ be $\{ i \mapsto s(i), z \mapsto \text{times}(z, f(i)) \}$. Then, we have the reduction $t \equiv u(x, i, z) \rightarrow_{i \rightarrow x} R_{\text{an}} u(x, s(i), \text{times}(z, f(i))) \equiv C[\theta]$. This reduction provides the following infinite reduction sequence:

$$
\begin{align*}
u(0, s(0), 0) &\rightarrow_{R_{\text{an}}} u(0, s(s(0)), \text{times}(0, f(s(0)))) \\
&\rightarrow_{R_{\text{an}}} u(0, s(s(s(0))), \text{times}(\text{times}(0, f(s(0))), f(s(s(0)))) \\
&\rightarrow_{R_{\text{an}}} \cdots
\end{align*}
$$

This observation is characterized as the following non-termination criterion.

Theorem 5.3. Let $R$ be a constrained TRS over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{M})$, and $\phi$ be a constraint that is satisfiable w.r.t. $\mathcal{M}$. Then, there exists an infinite derivation of $\vdash_R$ (i.e., $R$ is not terminating) if there exist a term $t$, a context $C[\!]$, and a $T(\mathcal{G}, \mathcal{V})$-substitution $\theta$ for $\text{fv}(\phi)$ such that $t \vdash_{\phi \circ R} C[\theta]$ and $\phi \Rightarrow \theta(\psi \land \eta)$ is valid w.r.t. $\mathcal{M}$.

Instead of detecting the reduction in Theorem 5.3, we may detect a rewrite rules (or a dependency pair) and a constraint such that the conditions in Theorem 5.3 are satisfied.

Corollary 5.4. Let $R$ be a constrained TRS over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{M})$. $R$ is not terminating if there exist a constrained rewrite rule $l \rightarrow r \Leftarrow \psi \in R$ (or $\in \text{DP}(R)$), a constraint $\eta$, and a $T(\mathcal{G}, \mathcal{V})$-substitution $\theta$ for $\text{fv}(\psi \land \eta)$ such that $r \equiv C[\theta]$ for some context $C[\!]$, $\psi \land \eta$ is satisfiable w.r.t. $\mathcal{M}$, and $\psi \land \eta \Rightarrow \theta(\psi \land \eta)$ is valid w.r.t. $\mathcal{M}$.

A constraint $\phi$ is called weaker than a constraint $\psi$ if $\phi \Rightarrow \psi$ is valid w.r.t. the structure. The role of $\eta$ in Corollary 5.4 is to make the constraint $\phi$ weaker.

A difficulty of automating the method based on Corollary 5.4 is how to detect a constraint $\phi$, while a rule is easily detected. In Example 5.2, we used the constraint $i \rightarrow x$ for the rule $u(x, i, z) \rightarrow u(x, s(i), \text{times}(z, f(i))) \iff x \rightarrow i \vee i \rightarrow x$. This constraint $i \rightarrow x$ is part of the constraint $x \rightarrow i \vee i \rightarrow x$ of this rule. Thus, the constraints of the constrained rewrite rules $l \rightarrow C[\theta] \Leftarrow \phi$ have a possibility to provide extra constraints $\eta$ that can be used to disprove termination via Corollary 5.4.

Next, we show a method to obtain from a constraint $\phi$ and a substitution $\theta$ an extra constraint $\eta$ in Corollary 5.4.

Theorem 5.5. Let $\mathcal{M}$ be a structure for $(\mathcal{G}, \mathcal{P})$, $\phi$ be a satisfiable constraint w.r.t. $\mathcal{M}$, and $\theta$ be a $T(\mathcal{G}, \mathcal{V})$-substitution for $\text{fv}(\phi)$. Let $\eta_1 \lor \cdots \lor \eta_k$ be a DNF of $\phi$, and $(V, E)$ be a directed graph such that

- $\bullet$ $V = \{ \eta_i \mid \eta_i$ is satisfiable w.r.t. $\mathcal{M} \}$, and
- $\bullet$ $E = \{ (\eta_i, \eta_j) \mid \eta_i, \eta_j \in V, \eta_i \Rightarrow \theta(\eta_j) \}$ is valid w.r.t. $\mathcal{M}$.

Then, every strongly connected subgraph (SCS) of $(V, E)$, whose node set $\{ \eta_1', \ldots, \eta_k' \}$ is not empty (i.e., $k > 0$), satisfies all of the following:

- $\bullet$ $\theta$ is a $T(\mathcal{G}, \mathcal{V})$-substitution for $\text{fv}(\phi \land (\eta_1' \lor \cdots \lor \eta_k'))$,
- $\bullet$ $\phi \land (\eta_1' \lor \cdots \lor \eta_k')$ is satisfiable w.r.t. $\mathcal{M}$, and
- $\bullet$ $\phi \land (\eta_1' \lor \cdots \lor \eta_k') \Rightarrow \theta(\phi \land (\eta_1' \lor \cdots \lor \eta_k'))$ is valid w.r.t. $\mathcal{M}$.

Since each of $\eta_i$ is weaker than $\phi$, the constraint $\eta_1' \lor \cdots \lor \eta_k'$ obtained from the SCSs can make $\phi$ weaker, i.e., it provides the weaker constraint $\phi \land (\eta_1' \lor \cdots \lor \eta_k')$. This is the reason why DNFs are used.
Example 5.6. Consider the constraint \( x \triangleright i \lor i \triangleright x \) again. Let \( \phi \) be \( x \triangleright i \lor i \triangleright x \) and \( \theta \) be a \( T(G_{PA}, V) \)-substitution for \( \{i\} \) such that \( \{i \mapsto s(i)\} \). Then, either \( x \triangleright i \Rightarrow x \triangleright s(i) \), \( x \triangleright i \Rightarrow s(i) \triangleright x \), or \( i \triangleright x \Rightarrow x \triangleright s(i) \) is not valid but \( i \triangleright x \Rightarrow s(i) \triangleright x \) is valid. Thus, we have the graph \( \{(i \triangleright x), \{(i \triangleright x, i \triangleright x)\}\} \). This graph is an SCS of itself.

Finally, we obtain the following sufficient condition for non-termination.

**Theorem 5.7.** A constrained TRS \( R \) over \((F, G, P, V, M)\) is not terminating if there exist a rule \( l \rightarrow C[l \theta] \equiv \phi \in R \) (or \( \in DP(R) \)) and a DNF \( \eta_1 \lor \cdots \lor \eta_n \) of \( \phi \) such that

- \( \theta \) is a \( T(G, V) \)-substitution for \( \text{fv}(\phi) \), and
- the directed graph \((V, E)\) has an SCS where
  - \( V = \{\eta_i \mid \eta_i \text{ is satisfiable w.r.t. } M\} \), and
  - \( E = \{(\eta_i, \eta_j) \mid \eta_i, \eta_j \in V, \eta_i \Rightarrow \theta(\eta_j) \text{ is valid w.r.t. } M\} \).

**Proof.** This theorem follows from Theorem 5.5. \( \square \)

Note that the sufficient condition in Theorem 5.7 is decidable if the validity of constraints over \((G, P, V)\) is decidable and if the standard transformation of formulas into DNFs is used. When deciding the satisfaction of this condition, it is enough and efficient to detect an SCC because the computation of SCCs is more efficient than that of SCSs. When \( k = 1 \) in Theorem 5.7, it is enough to check the validity of \( \eta_1 \Rightarrow \theta(\eta_1) \).

**Example 5.8.** Consider \( R_{\text{sn}} \) again. \( R_{\text{sn}} \) is not terminating because the graph constructed from the constrained rule \( u(x, i, z) \rightarrow u(x, s(i), \text{times}(z, f(i))) \equiv x \triangleright i \lor i \triangleright x \) has an SCS (see Example 5.6). Therefore, \( R_{\text{sn}} \) is not terminating.

For linear arithmetics over integers, this paper uses only the predicates \( \simeq \) and \( \triangleright \). On the other hand, other predicates such as \( \triangleleft \) and \( \trianglerighteq \) are often used with \( \simeq \) and \( \triangleright \). An \( n \)-ary predicate \( p \) is called *weaker than* an \( n \)-ary predicate \( q \) if \( p(x_1, \ldots, x_n) \Rightarrow q(x_1, \ldots, x_n) \) is valid w.r.t. the structure. For example, \( \triangleright \) is strictly weaker than \( \triangleleft \) and \( \trianglerighteq \). Thus, as DNFs of linear arithmetic formulas over \((G_Z, \{\simeq, \triangleleft, \trianglerighteq\})\), DNFs over \((G_Z, \{\simeq, \triangleright\})\) are adequate, in the sense to obtain DNFs \( \eta_1 \lor \cdots \lor \eta_n \) such that each \( \eta_i \) is as weak as possible.

The method in this section to disprove termination analyzes constrained rules of the form \( l \rightarrow C[l \theta] \equiv \phi \). For this reason, the method is not so strong while the non-termination criterion in Theorem 5.3 is a general one. To make the method more practical, we need to improve it. To this end, we will extend Corollary 5.4 for rewrite rules \( t_1 \rightarrow C_1[t_2 \theta_1] \equiv \phi_1, t_2 \rightarrow C_2[t_3 \theta_2] \equiv \phi_2, \ldots, t_n \rightarrow C_n[t_1 \theta_n] \equiv \phi_n \). Moreover, we have to experiment for many non-terminating constrained TRSs in order to verify the usefulness of the method.

**References**


Appendix A. Proofs

A.1. Proof of Theorem 4.4

Before proving Theorem 4.4, we describe a remark.

Let \( s_1 \overset{\sigma_1}{\rightarrow} t_1 \overset{\sigma_2}{\rightarrow} t_2 \cdots \) construct a \((S,R)\)-chain using substitutions \( \sigma_1, \sigma_2, \ldots \). Then, the chain implies the following derivation sequence:

\[
s_1 \sigma_1 \xrightarrow{\varepsilon} t_1 \sigma_1 \xrightarrow{\varepsilon} t_2 \sigma_2 \xrightarrow{\varepsilon} \cdots.
\]

Moreover, \( \sigma_1, \sigma_2, \ldots \) are terminating substitutions. The proof of this theorem follows from the proof of the original DP theorem [Art00].

Next, we show the proof of Theorem 4.4.

Proof. The proof of this theorem follows from the proof of the original DP theorem [Art00]. Here, we show another approach to the proof.

Let \( R' \) be the (possibly infinite) set \( \{(l, r, \sigma) \mid l \rightarrow r \Leftarrow \phi \in R, \text{fv}(\phi) \subseteq \text{Dom}(\sigma), \text{Ran}(\sigma|_{\text{fv}(\phi)}) \subseteq T(G), M \models \sigma(\phi)\} \). Then, \( R' \) can be considered as a (possibly infinite) set of unconstrained rewrite rules. Let \( S \) be the (possibly infinite) set \( \{(s, \ell, \sigma) \mid s \rightarrow t \Leftarrow \psi \in DP(R), \text{fv}(\psi) \subseteq \text{Dom}(\sigma), \text{Ran}(\sigma|_{\text{fv}(\psi)}) \subseteq T(G), M \models \sigma(\psi)\} \). It follows from the definition of dependency pairs that \( DP(R') = S \). Thus, \( DP(R') \) can be considered as a (possibly infinite) set of term pairs. The original DP theorem [Art00] is applicable to the pair \((DP(R'), R')\) since finiteness of the sets \( DP(R') \) and \( R' \) is not necessary for the DP theorem to hold. Thus, we have that \( R' \) is terminating iff there is no infinite minimal \((DP(R'), R')\)-chain.

It follows from the constructions of \( R' \) and \( S \) that \( \rightarrow_R = \rightarrow_{R'} \) and \( \rightarrow^{\varepsilon}_{DP(R)} = \rightarrow^{\varepsilon}_{DP(R')} \).

It is obvious that \( R \) is terminating iff so is \( R' \). Moreover, it is obvious that every derivation sequence w.r.t. \( \rightarrow^{\varepsilon}_{DP(R)} \), \( \varepsilon \) is a derivation sequence w.r.t. \( \rightarrow^{\varepsilon}_{DP(R')} \), and vice versa. Thus, we have that there is no infinite minimal \((DP(R), R)\)-chain iff there is no infinite minimal \((DP(R'), R')\)-chain. Therefore, this theorem holds.

A.2. Proof of Theorem 4.11

The local soundness for structures yields a necessary condition for pairs of dependency pairs that construct chains.

Proposition A.1. Let \( R \) be a constrained TRS over \((F, G, P, V, M)\) that is locally sound for \( M \), and \( S \subseteq DP(R) \). Let a sequence \( u \overset{\sigma}{\rightarrow} f^2(t_1, \ldots, t_n) f^2(s_1, \ldots, s_n) \overset{\psi}{\rightarrow} v \) be an \((S,R)\)-chain with substitutions \( \sigma \) and \( \theta \) such that \( M \models \sigma(\phi), M \models \theta(\psi) \) and \( f^2(t_1, \ldots, t_n) \sigma \xrightarrow{\varepsilon} f^2(s_1, \ldots, s_n) \theta \) where \( \text{Var}(u) \cap \text{Var}(s_1, \ldots, s_n) = \emptyset \). Then, we have

\[
M \models (\phi \land \psi) \land \left( \bigwedge_{i \in \{j \mid t_j, s_j \in T(G,V), \text{Var}(t_j) \subseteq \text{fv}(\phi)\}} t_i \simeq s_i \right)(\sigma \cup \theta),
\]

i.e., \( \phi \land \psi \land \left( \bigwedge_{i \in \{j \mid t_j, s_j \in T(G,V), \text{Var}(t_j) \subseteq \text{fv}(\phi)\}} t_i \simeq s_i \right) \) is satisfiable w.r.t. \( M \).

---

5A substitution \( \sigma \) is terminating w.r.t. \( \rightarrow \) if \( x \sigma \) is terminating w.r.t. \( \rightarrow \) for every \( x \in \text{Dom}(\sigma) \).
Theorem 4.11.

\[
\begin{array}{c}
\text{chain } \cdots \xrightarrow{c} s_k \xrightarrow{c} t_k \xrightarrow{c} s_{k+1} \xrightarrow{c} t_{k+1} \xrightarrow{c} \cdots \\
\text{\textit{T(G)}-seq.} \\
\mathcal{B}_1 \quad \mathcal{B}_2 \\
\text{Z-seq.} \\
\end{array}
\]

Figure 4: overview of the proof of Theorem 4.11.

Proof. Since \(\text{Var}(u) \cap \text{Var}(s_1, \ldots, s_n) = \emptyset\), we can assume w.o.l.g. that \(\text{Dom}(\sigma) \cap \text{Dom}(\theta) = \emptyset\). Then, \(\sigma \cup \theta\) is a substitution. If \(\text{Var}(t_j) \subseteq \text{fv}(\phi)\) and \(t_j \in \mathcal{T}(\mathcal{G}, \mathcal{V})\), then \(t_j \sigma \in \mathcal{T}(\mathcal{G})\) by definition. It follows from local soundness of \(R\) that \(s_j \theta \in \mathcal{T}(\mathcal{G})\) and \((t_j \sigma)^M = (s_j \theta)^M\) if \(\text{Var}(t_j) \subseteq \text{fv}(\phi)\). Thus, \(\mathcal{M} = (\bigwedge_{i \in \{i \mid t_j \in \mathcal{T}(\mathcal{G}, \mathcal{V}), \text{Var}(t_j) \subseteq \text{fv}(\phi)\}} t_i \approx s_i\)(\(\sigma \cup \theta\)). Therefore, we have \(\mathcal{M} = (\phi \land \psi \land (\bigwedge_{i \in \{i \mid t_j \in \mathcal{T}(\mathcal{G}, \mathcal{V}), \text{Var}(t_j) \subseteq \text{fv}(\phi)\}} t_i \approx s_i)))(\sigma \cup \theta)\).

By definition, tree homomorphisms have the following property.

**Proposition A.2.** Let \(R\) be a constrained TRS over \((\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{M})\) that is locally sound for \(\mathcal{M}\), \(\mathcal{H}\) be a tree homomorphism for \(\mathcal{D}^2\), \(s\) and \(t\) be terms in \(\mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{V})\) with \(\text{root}(s) \in \mathcal{D}_R\), and \(\sigma\) and \(\theta\) be \(\mathcal{T}(\mathcal{G})\)-substitutions such that \(\text{Dom}(\sigma) \subseteq \text{Var}(s)\), \(\text{Dom}(\theta) \subseteq \text{Var}(t)\) and \(s^\sigma \xrightarrow{c} R t^\theta\). If \(\mathcal{H}(s^2) \in \mathcal{T}(\mathcal{G}, \mathcal{V})\) and \(\text{Var}(\mathcal{H}(s^2)) \subseteq \text{Dom}(\sigma)\), then \(\mathcal{H}(t^2) \in \mathcal{T}(\mathcal{G}, \mathcal{V})\) and \((\mathcal{H}(s^2))^M = (\mathcal{H}(t^2))^M\).

Proof. Let \(s = f(s_1, \ldots, s_n)\). Suppose that \(s^\sigma \xrightarrow{c} R t^\theta\). Since \(R\) cannot reduce any term rooted by \(f^2\), we have that \(\text{root}(t) = f\). Let \(t = f(t_1, \ldots, t_n)\). Then, we have \(s^\sigma \xrightarrow{c} R t^\theta\) for \(1 \leq i \leq n\). It follows from \(\mathcal{H}(s^2) \in \mathcal{T}(\mathcal{G}, \mathcal{V})\), \(\text{Dom}(\sigma) \subseteq \text{Var}(s)\), and \(\text{Ran}(\sigma) \subseteq \mathcal{T}(\mathcal{G})\) that \(s_i \sigma \in \mathcal{T}(\mathcal{G})\). It follows from local soundness of \(R\) that \(t_i \theta \in \mathcal{T}(\mathcal{G})\) and \((s_i \sigma)^M = (t_i \theta)^M\). By the definition of \(\mathcal{H}\), we have that \(\mathcal{H}(s_i) \equiv s_i\), \(\mathcal{H}(t_i) \equiv t_i\), \(\mathcal{H}(s_i \sigma) \equiv s_i \sigma\) and \(\mathcal{H}(t_i \theta) \equiv t_i \theta\). Thus we have that \(\mathcal{H}(s^2) \equiv \mathcal{H}(f^2 \{x_i \mapsto s_i \sigma \mid 1 \leq i \leq n\}) \text{ and } \mathcal{H}(t^2) \equiv \mathcal{H}(f^2 \{x_i \mapsto t_i \theta \mid 1 \leq i \leq n\})\). Therefore, we have \(\mathcal{H}(t^2) \in \mathcal{T}(\mathcal{G}, \mathcal{V})\) and \((\mathcal{H}(s^2))^M = (\mathcal{H}(t^2))^M = (\mathcal{H}(f^2 \{x_i \mapsto s_i \sigma \mid 1 \leq i \leq n\}))^M = (\mathcal{H}(f^2))^M \{x_i \mapsto (s_i \sigma)^M \mid 1 \leq i \leq n\} = (\mathcal{H}(f^2))^M \{x_i \mapsto (t_i \theta)^M \mid 1 \leq i \leq n\} = (\mathcal{H}(f^2 \{x_i \mapsto t_i \theta \mid 1 \leq i \leq n\}))^M = (\mathcal{H}(t^2))^M\).

Next, we show a proof of Theorem 4.11. The overview of the proof is illustrated in Figure 4.

Proof. We proceed by contradiction. Let \(s_1 \stackrel{\phi_1}{\rightarrow} t_1 s_2 \stackrel{\phi_2}{\rightarrow} t_2 \cdots\) be a strongly infinite minimal \((S, R)\)-chain such that \(t_i \sigma \xrightarrow{c} R s_{i+1} \sigma_{i+1}\), \(\mathcal{M}_2 \models \phi_i \sigma_i\) and \(\text{Ran}(\sigma_i|_{\text{fv}(\phi_i)}) \subseteq \mathcal{T}(\mathcal{G})\) for all \(i \geq 1\). It follows from \(\mathcal{A}_2\) that there is some \(k\) such that \(\mathcal{H}(t_k \sigma_k) \in \mathcal{T}(\mathcal{G})\). Thus, it follows from \(\mathcal{A}_1\) and \(\text{Ran}(\sigma_i|_{\text{fv}(\phi_i)}) \subseteq \mathcal{T}(\mathcal{G})\) that \(\mathcal{H}(s_i \sigma_j), \mathcal{H}(t_j \sigma_j) \in \mathcal{T}(\mathcal{G})\) for every \(j > k\). It follows from Proposition A.2 that \((\mathcal{H}(t_j \sigma_j))^M \geq (\mathcal{H}(s_j \sigma_j))^M\) for all \(j \geq k\). Moreover, it follows from \(\mathcal{B}_1\) that \((\mathcal{H}(s_j \sigma_j))^M \geq (\mathcal{H}(t_j \sigma_j))^M\) for all \(j \geq k\). Thus, we have the following
infinite sequence on integers:
\[
(H(t_k\sigma_k))^{M_Z} = (H(s_{k+1}\sigma_{k+1}))^{M_Z} \geq (H(t_{k+1}\sigma_{k+1}))^{M_Z} = (H(s_{k+2}\sigma_{k+2}))^{M_Z} \\
\geq (H(t_{k+2}\sigma_{k+2}))^{M_Z} = (H(s_{k+3}\sigma_{k+3}))^{M_Z} \geq \ldots .
\]
It follows from B2 that \( > \) appears infinitely many times in the above sequence.

It follows from Proposition A.1 that for every pairs \( u \rightarrow f^2(u_1, \ldots, u_n) \equiv \phi \) and \( f^2(v_1, \ldots, v_n) \rightarrow v \Leftarrow \psi \) in \( S \), if they are connected by \( \mathcal{T}(\mathcal{G}) \)-substitutions \( \sigma \) and \( \theta \) with \( \text{fv}(\sigma) \subseteq \text{Dom}(\psi) \), then \( M_Z \models (\phi \land \psi \land (\wedge_{i \in \{j|u_j, v_j \in \mathcal{T}(\mathcal{G}), \text{Var}(u_j) \subseteq \text{Var}(\phi)\}} u_i \approx v_i)) (\sigma \cup \theta) \). Since \( H(t_i\sigma_i) \in \mathcal{T}(\mathcal{G}) \) for \( i \geq k \) and since \( u_j\sigma \xrightarrow{\mathcal{T}(\mathcal{G})} v_j\theta \) for all \( j \) with \( \text{Var}(u_j) \subseteq \text{Var}(H(u)) \cup \text{fv}(\phi) \), we have \( u_j\sigma \in \mathcal{T}(\mathcal{G}) \). Since \( R \) is locally sound for \( M_Z \), we have \( v_j\theta \in \mathcal{T}(\mathcal{G}) \) and \( (u_j\sigma)^{M_Z} = (v_j\theta)^{M_Z} \) for all \( j \) such that \( \text{Var}(u_j) \subseteq \text{Var}(H(u)) \cup \text{fv}(\phi) \), and hence \( v_j \in \mathcal{T}(\mathcal{G}, V) \). Thus, it follows from C that there exists an integer \((m)^{M_Z} \) and \( k' \geq k \) such that \((H(s_{k'}\sigma_{k'}))^M_Z > (m)^{M_Z}\). Since each element in \( S \) appears infinitely many times, the pair at \( k' \)th step appears infinitely often after \( k \). Thus, we have \( (H(s_i\sigma_i))^M_Z > (m)^{M_Z} \) for all \( i \geq k' \). This contradicts the fact that for any integer \( m' \), there is no infinite sequence \( n_1 > n_2 > \cdots \) of integers such that \( n_i > m' \).

### A.3. Proof of Theorem 4.14

To simplify proofs for soundness of DP processors, we characterize sound DP processors.

**Proposition A.3.** Let \((S, R)\) be a finite DP problem. Then, for any \( S' \subseteq S \), the DP problem \((S', R)\) is finite.

**Proof.** By definition, this proposition holds clearly. ■

**Lemma A.4.** Let \((S, R)\) be a DP problem, and \( S_1, \ldots, S_n \subseteq S \) \( (n > 0) \). Suppose that for any \( S' \subseteq S \), if \( S_i \cap S' \neq \emptyset \) for all \( i \in \{1, \ldots, n\} \), then there is no strongly infinite minimal \((S', R)\)-chain. If the DP problem \((S, R)\) is not finite, then there exists some \( i \in \{1, \ldots, n\} \) such that the DP problem \((S \setminus S_i, R)\) is not finite.

**Proof.** We prove by induction on \(|S|\) the contraposition, i.e., if \((S \setminus S_i, R)\) is finite for \( 1 \leq i \leq n \), then \((S, R)\) is finite. Since \( \bigcup_{i=1}^n S_i \subseteq S \), we have that \( S_i \cap S \neq \emptyset \) for \( 1 \leq i \leq n \). It follows from the assumption that there is no strongly infinite minimal \((S, R)\)-chain. Thus, from Theorem 4.6 (2), it is enough to show that for any \( S' \subseteq S \), \((S', R)\) is finite.

- Consider the case that \( S' \subseteq S \setminus S_i \) for some \( i \). It follows from Proposition A.3 and the assumption that \((S', R)\) is finite.
- Consider the remaining case that \( S' \not\subseteq S \setminus S_i \) for \( 1 \leq i \leq n \). It follows from \( S' \subseteq S \) that \( S' \cap S_i \neq \emptyset \) for \( 1 \leq i \leq n \). Then, it follows from the assumption that for any \( S'' \subseteq S' \), there is no strictly infinite \((S'', R)\)-chain. Since \( S' \subseteq S \), we have that \( S' \setminus S_i \subseteq S \setminus S_i \). It follows from Proposition A.3 that \((S' \setminus S_i, R)\) is finite for \( 1 \leq i \leq n \). Since \( S' \subseteq S \), \((S', R)\) is finite by the induction hypothesis. Therefore, the DP problem \((S, R)\) is finite. ■

**Theorem A.5.** Let \((S, R)\) be a DP problem, and \( S_1, \ldots, S_n \subseteq S \) \( (n > 0) \). Suppose that for any \( S' \subseteq S \), if \( S_i \cap S' \neq \emptyset \) for all \( i \in \{1, \ldots, n\} \), then there is no strongly infinite minimal \((S', R)\)-chain. Let \( \text{Proc} \) be a DP processor such that \( \text{Proc}(\langle S, R \rangle) \) is either \( \{(S, R)\} \) or \( \{(S \setminus S_i, R) | 1 \leq i \leq n \} \). Then, \( \text{Proc} \) is sound.

**Proof.** This theorem follows from Lemma A.4. ■
Next, we show the proof of Theorem 4.14.

**Proof.** It follows from Theorem 4.11 that for any \( S' \subseteq S \) such that \( S_{A2} \cap S' \neq \emptyset, S_{B2} \cap S' \neq \emptyset \) and \( S_C \cap S' \neq \emptyset \), there is no strongly infinite minimal \((S', R)\)-chain. Therefore, due to Theorem A.5, the DP processor in Definition 4.13 is sound. \( \blacksquare \)

### A.4. Proof of Theorem 5.3

**Proof.** We first show that if \( s \overset{n}{\rightarrow}_R t \), then \( s\theta \overset{n}{\rightarrow}_R t\theta \) for any \( T(\mathcal{G}, \mathcal{V}) \)-substitution \( \theta \) for \( \text{fv}(\phi) \) such that \( \theta(\phi) \) is satisfiable w.r.t. \( \mathcal{M} \) and \( \phi \Rightarrow \theta(\phi) \) is valid w.r.t. \( \mathcal{M} \). We prove this claim by induction on \( n \). Since the case that \( n = 0 \) is clear, we only consider the remaining case that \( n > 1 \). Suppose that \( s \equiv C[l]\sigma \overset{l}{\rightarrow}_R C[r]\sigma \overset{n-1}{\rightarrow}_R t \) where \( l \rightarrow r \rightleftharpoons \psi \in R \) and \( \phi \Rightarrow \sigma(\psi) \) is valid w.r.t. \( \mathcal{M} \). Then, \( \theta(\phi) \Rightarrow \theta(\sigma(\psi)) \) is also valid because of the validity of \( \phi \Rightarrow \sigma(\psi) \). It follows from the validity of \( \phi \Rightarrow \theta(\phi) \) and \( \theta(\phi) \Rightarrow \theta(\sigma(\phi)) \) that \( \phi \Rightarrow \theta(\sigma(\phi)) \) is also valid. Thus, we have \( (C[l]\sigma)\theta \overset{n-1}{\rightarrow}_R (C[r]\sigma)\theta \). By the induction hypothesis, we have \( (C[r]\sigma)\theta \overset{n-1}{\rightarrow}_R t\theta \). Therefore, we have \( s\theta \overset{n}{\rightarrow}_R t\theta \).

Next, we prove the theorem. Due to the above claim, we have that \( t\theta \overset{+}{\rightarrow}_R (C[t\theta])\theta, t\theta \overset{+}{\rightarrow}_R (C[t\theta])\theta, \ldots \). Thus, we have an infinite derivation \( t \overset{+}{\rightarrow}_R C[t\theta] + \overset{+}{\rightarrow}_R C[(C[t\theta])\theta] + \overset{+}{\rightarrow}_R C[(C[t\theta])\theta] \cdots \). Since \( \phi \) is satisfiable, there exists a \( T(\mathcal{G}) \)-substitution \( \delta \) for \( \text{fv}(\phi) \) such that \( \mathcal{M} \models \delta(\phi) \). Thus, we have an infinite derivation \( t\delta \overset{+}{\rightarrow}_R (C[t\theta])\delta \overset{+}{\rightarrow}_R (C[C[t\theta]]\theta)\delta \overset{+}{\rightarrow}_R (C[C[t\theta]]\theta)\delta \overset{+}{\rightarrow}_R \cdots \). Therefore, \( R \) is not terminating. \( \blacksquare \)

### A.5. Proof of Theorem 5.5

**Proof.** Since \( \eta_1 \lor \cdots \lor \eta_n \) is a DNF of \( \phi \) and \( \{\eta'_1, \cdots, \eta'_k\} \subseteq \{\eta_1, \cdots, \eta_n\} \), we have \( \text{fv}(\phi) = \text{fv}(\eta_1 \lor \cdots \lor \eta_n) \). Thus, it follows from the assumption that \( \theta \) is a \( T(\mathcal{G}, \mathcal{V}) \)-substitution for \( \text{fv}(\psi \lor (\eta'_1 \lor \cdots \lor \eta'_k)) \) and \( \phi \lor (\eta'_1 \lor \cdots \lor \eta'_k) \) is satisfiable. Since \( \eta_1 \lor \cdots \lor \eta_n \) is a DNF of \( \phi \) and \( \{\eta'_1, \cdots, \eta'_k\} \subseteq \{\eta_1, \cdots, \eta_n\} \), we have that \( (\eta'_1 \lor \cdots \lor \eta'_k) \Rightarrow \phi \) is valid and hence \( \phi \lor (\eta'_1 \lor \cdots \lor \eta'_k) \) is equivalent to \( \eta'_1 \lor \cdots \lor \eta'_k \). Thus, we only show the validity of \( (\eta'_1 \lor \cdots \lor \eta'_k) \Rightarrow \theta(\eta'_i) \) for \( 1 \leq i \leq k \). It follows from the construction of \( E \) and SCSs that there are some \( i \) and \( j \) such that \( \eta'_i \Rightarrow \theta(\eta'_j) \) is valid for \( 1 \leq i \leq k \). Thus, it follows from the validity of \( \eta'_i \Rightarrow \theta(\eta'_j) \) for \( 1 \leq i \leq k \) that \( (\eta'_1 \lor \cdots \lor \eta'_k) \Rightarrow \theta(\eta'_1 \lor \cdots \lor \eta'_k) \) is also valid. Therefore, this theorem was proved. \( \blacksquare \)

### Appendix B. Remark of Theorem 4.11

Consider the DP problems \( \{(1), (4), (7), \} \}, R_t \) and \( \{(4), R_t \} \) again. To show the non-existence of strongly infinite minimal \( \{(4), R_t \} \)-chains, the following condition is sufficient instead of \( C \) in Theorem 4.11:

- There exists a term \( m \in T(\mathcal{G}) \) and a pair \( u \rightarrow f^2(t_1, \ldots, t_n) \iff \phi \in S \) such that \( \phi \Rightarrow \mathcal{M}(u) \rightarrow m \) is valid w.r.t. \( \mathcal{M}_z \).
Note that $C'$ implies $C$. Here we explain the reason why we use the complicated condition $C$. Consider the DP problem $(\{(1),(4),(7)\},R_I)$. When $B_1$ is satisfied, there is no pair satisfying $C'$, i.e., the lower bound cannot be detected. However, every pair appears infinitely many times because of strong infinity of chains. Thus, to detect lower bounds, we can use constraints in the next pairs (direct successor nodes): $\psi$ is one of the constraints and $\bigwedge_{i \in \{j|s_j,t_j \in T(\emptyset,\emptyset), \text{Var}(t_j) \subseteq \text{Var}(u) \land \text{Var}(\emptyset)\}} t_i \simeq s_i$ means that $f^i(s_1,\ldots,s_n) \rightarrow v = \psi$ follows after $u \rightarrow f^1(t_1,\ldots,t_n) = \phi$.

In Theorem 4.11 $C$, to detect lower bounds $m$, we use information in the direct successor nodes. Given a constant $k$, it is easy to extend this approach for the use of information $k$-continuous successor nodes. There exist examples where such an extension succeeds in proving termination. Moreover, $B_2$ can be extended similarly to the extension of $C'$ to $C$, by replacing $\mathcal{H}(u) \succ m$ in $C$ with $\mathcal{H}(u) \succ \mathcal{H}(v)$.

Appendix C. DP Processor based on the Subterm Criterion

Here, we adapt the DP processor based on the subterm criterion to the one for constrained TRSs.

We define the function $\text{rmc}$ that eliminates constraints from constrained pairs as follows: $\text{rmc}(S) = \{l \rightarrow r \mid l \rightarrow r \Leftarrow \phi \in S\}$. It is obvious that $\rightarrow_S \subseteq \rightarrow_{\text{rmc}(S)}$. Thus, the following claim holds obviously.

**Theorem C.1.** Let $R$ be a constrained TRS, and $S \subseteq \text{DP}(R)$. Then, there is no strongly infinite minimal $(S,R)$-chain if there is no strongly infinite minimal $(\text{rmc}(S),R)$-chain.

**Proof.** It follows from $\rightarrow_S \subseteq \rightarrow_{\text{rmc}(S)}$ that $(S,R)$-chains are $(\text{rmc}(S),R)$-chains. For this reason, this theorem holds. \hfill $\blacksquare$

**Definition C.2** (simple projection [Hir07]). Let $\mathcal{F}$ be a signature. A simple projection for $\mathcal{F}$ is a mapping $\pi$ that assigns to every $n$-ary function symbol $f$ in $\mathcal{F}$ an argument position $i \in \{1,\ldots,n\}$. The application of $\pi$ to every term $f(t_1,\ldots,t_n)$ with an $n$-ary symbol $f$ in $\mathcal{F}$, is defined as $\pi(f(t_1,\ldots,t_n)) = t_{\pi(f)}$.

**Theorem C.3** (subterm criterion). Let $R$ be an unconstrained TRS, $S \subseteq \text{DP}(R)$, and $\pi$ be a simple projection for $\mathcal{D}^\emptyset$ such that $\{(\pi(s),\pi(t)) \mid s \rightarrow t \in S\} \subseteq \succ$. Let $S_{\succ} = \{(\pi(s),\pi(t)) \mid s \rightarrow t \in S\} \cap \succ$. Then, there is no strongly infinite minimal $(S_{\succ},R)$-chain.

**Proof.** The proof follows from the original one in [Hir07] for unconstrained TRSs. \hfill $\blacksquare$

**Definition C.4** (DP processor based on the subterm criterion). Let $R$ be a constrained TRS, $S \subseteq \text{DP}(R)$, and $\pi$ be a simple projection for $\mathcal{D}^\emptyset$ such that $\{(\pi(s),\pi(t)) \mid s \rightarrow t \Leftarrow \phi \in S\} \subseteq \succ$. Let $S_{\succ} = \{(\pi(s),\pi(t)) \mid s \rightarrow t \Leftarrow \phi \in S\} \cap \succ$. A DP processor $\text{Proc}$ is defined as follows:

$$\text{Proc}(\langle S,R \rangle) = \{ (S \setminus S_{\succ},R) \}.$$  

**Theorem C.5.** The DP processor in Definition C.4 is sound.

**Proof.** This theorem follows from Theorem C.3 and Theorem A.5. \hfill $\blacksquare$
Appendix D. Encoding DP Processor based on Polynomial Interpretation

Here, we show how to encode the conditions A1, A2, B1, B2, and C as constraints over the interpretable signature. Let \( \mathcal{H} \) be a tree homomorphism for \( D^2 \) such that \( \mathcal{H}(f^2) \) corresponds to a linear expression \( k_0^f + k_1^f x_1^f + \cdots + k_n^f x_n^f \) where \( n = \text{arity}(f) \), the coefficients \( k_0^f, \ldots, k_n^f \) are variables on integers, and coefficient variables for marked symbols are distinct. Notice that \( \mathcal{H}(f) \) needs not to be mentioned, because \( \mathcal{H}(f) \) is not used due to the satisfaction of A1. We define sets \( K \) and \( K_0 \) of coefficient variables as follows:

\[
K = \{ k_0^f, \ldots, k_n^f \mid n = \text{arity}(f), \ \mathcal{H}(f^2) \text{ corresponds to } k_0^f + k_1^f x_1^f + \cdots + k_n^f x_n^f \},
\]

\[
K_0 = \{ k_0^f \mid n = \text{arity}(f), \ \mathcal{H}(f^2) \text{ corresponds to } k_0^f + k_1^f x_1^f + \cdots + k_n^f x_n^f \}.
\]

The cardinality of \( K \cup K_0 \) is at most \( \sum_{f^2 \in F_S} \text{arity}(f^2) \) where \( F_S \) is the set of marked symbols appearing in \( S \). For a finite variable set \( X = \{x_1, \ldots, x_n\} \), we denote \( \forall x_1 \cdots \forall x_n. \phi \) (\( \exists x_1 \cdots \exists x_n. \phi \), resp.) by \( \forall X. \phi \) (\( \exists X. \phi \), resp.). We first assign 0 to some variables in \( K \) as A1 and A2 are satisfied. We update \( K \) by removing all \( k_i^f \) that are assigned with 0, and assign 0 to \( k_i^f \) in \( \mathcal{H}(f^2) \), i.e., we remove \( k_i^f x_i \) from \( \mathcal{H}(f^2) \). Suppose that all variables in \( S \) are renamed as variables in different pairs are distinct. Then the conditions B1, B2, and C are encoded to constraints as follows:

B1. \( \bigwedge_{s \leftarrow t \in F_S} (\forall X_B. (\phi \Rightarrow \mathcal{H}(s) \supset \mathcal{H}(t))) \),

B2. \( \bigvee_{s \leftarrow t \in F_S} (\forall X_B. (\phi \Rightarrow \mathcal{H}(s) \supset \mathcal{H}(t))) \), and

C. \( \forall a \rightarrow f^2(s_1, \ldots, s_n) \leftarrow \forall \psi \in F_S (X_C. (\text{the constraint in } C\text{'}))^6 \)

where \( X_B = \text{Var}(\mathcal{H}(s)) \cup \text{fv}(\phi) \) and \( X_C = \text{Var}(\mathcal{H}(u), \mathcal{H}(f^2(s_1, \ldots, s_n))) \cup \text{fv}(\phi) \cup \text{fv}(\psi) \). We denote these constraints by \( \langle \text{B1} \rangle \), \( \langle \text{B2} \rangle \), and \( \langle C \rangle \), respectively. Then, the conjunction of B1, B2 and C with \( \mathcal{H} \) having unknown coefficients is the following constraint:

\( \exists K. \exists K_0. (\langle \text{B1} \rangle \land \langle \text{B2} \rangle \land \langle C \rangle) \).

If this constraint is valid w.r.t. \( M_Z \), then there is no strongly infinite minimal \( (S, R) \)-chain. Unfortunately, this constraint is not over linear integer arithmetics but a formula over non-linear integer arithmetics because it contains \( kx \) where \( k \) is a coefficient variable and \( x \) is a variable in dependency pairs. However, by using incomplete constraint solvers [Apt03] or SMT solvers for quantified non-linear arithmetics over integers, the validity of this constraint or the satisfiability of \( \langle \text{B1} \rangle \land \langle \text{B2} \rangle \land \langle C \rangle \) may be solved. A naïve approach is shown in Appendix F.

Appendix E. Refinement of Estimated Dependency Graphs

In general, the dependency graph of \( R \) is not computable. For this reason, approximations of the graphs are often used where a graph \( (V, E) \) is an (over-)approximation of a graph \( (V', E') \) if \( V \subseteq V' \) and \( E \subseteq E' \). The estimated dependency graphs of constrained TRSs are defined as well as those of unconstrained TRSs.

Definition E.1 (estimated dependency graph). Let \( R \) be a constrained TRS. The estimated dependency graph (EDG) of \( R \), written as \( EDG(R) \), is a directed graph \( (V, E) \) such

\[ \text{If } f^2(s_1, \ldots, s_n) \rightarrow v \leftarrow \psi \text{ is the same rule as } u \rightarrow f^2(t_1, \ldots, t_n) \leftarrow \phi, \text{ then we rename variables in } f^2(s_1, \ldots, s_n) \rightarrow v \leftarrow \psi. \]
that $V = DP(R)$, and $E = \{ (s \rightarrow t \leftarrow \phi, u \rightarrow v \leftarrow \psi) \mid s \rightarrow t \leftarrow \phi, u \rightarrow v \leftarrow \psi \in DP(R), \text{Ren}(\text{Cap}(t)) \text{ and } u \text{ are unifiable} \} \, ^7$.

Note that $EDG(R)$ is an approximation of $DG(R)$.

**Example E.2.** Consider the constrained TRS $R_f$ again. The estimated dependency graph $EDG(R_{f})$ is illustrated in Figure 5. $EDG(R_{f})$ has the SCCs $\{(9),(11)\}$, $\{(13),(15)\}$, and $\{(17),(19)\}$ that are not SCCs in $DG(R_{f})$.

Consider the arc from $(9)$ to $(11)$ in $EDG(R_{f})$. In $DG(R_{f})$, there is no arc from $(9)$ to $(11)$. Let $\sigma$ be a substitution such that $M_{PA} \models (s(x) \succ 0)\sigma$. To connect $(11)$ with $(9)$ by a substitution $\theta$, $\theta$ has to satisfy $s(x)\sigma \xrightarrow{R_{i}} p(x')\theta$ and $M_{PA} \models (0 \succ p(x'))\theta$. However, $s(x)\sigma \xrightarrow{R_{i}} p(x')\theta$ implies $(s(x)\sigma)^{M_{PA}} = (p(x')\theta)^{M_{PA}}$. However, this is not true because $(s(x)\sigma)^{M_{PA}} > 0$ and $(p(x')\sigma)^{M_{PA}} < 0$.

In the definition of EDGs, constraints in the dependency pairs are ignored in computing arcs. For this reason, the arcs from $(9)$ to $(11)$ are computed. To avoid computing such arcs, Proposition A.1 provides a criterion to detect fake arcs in EDGs.

**Theorem E.3.** Let $R$ be a constrained TRS over $(\mathcal{F}, \mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{M})$ that is locally sound for $\mathcal{M}$, and $u \rightarrow f^{2}(t_{1}, \ldots, t_{n}) \leftarrow \phi, f^{2}(s_{1}, \ldots, s_{n}) \rightarrow v \leftarrow \psi \in DP(R)$ such that $\text{Var}(u) \cap \text{Var}(s_{1}, \ldots, s_{n}) = \emptyset$. $DG(R)$ has no arc from $u \xrightarrow{\phi}, f^{2}(t_{1}, \ldots, t_{n})$ to $f^{2}(s_{1}, \ldots, s_{n}) \xrightarrow{\psi} v$ if any of the following holds:

- $t_{i} \in \mathcal{T}(\mathcal{G}, \mathcal{V}), s_{i} \notin \mathcal{T}(\mathcal{G}, \mathcal{V})$ and $\text{Var}(t_{i}) \subseteq \text{fv}(\phi)$ for some $i$, or
- $\phi \land \psi \land \left( \bigwedge_{i \in \{j \mid s_{j}, t_{j} \in \mathcal{T}(\mathcal{G}, \mathcal{V}), \text{Var}(t_{j}) \subseteq \text{fv}(\phi) \}} t_{i} \simeq s_{i} \right)$ is not satisfiable for $\mathcal{M}$.

**Proof.** Suppose that there is an arc from $u \xrightarrow{\phi}, f^{2}(t_{1}, \ldots, t_{n})$ to $f^{2}(s_{1}, \ldots, s_{n}) \xrightarrow{\psi} v$. Let $\sigma$ and $\theta$ be substitutions such that $\text{fv}(\phi) \subseteq \text{Dom}(\sigma)$, $\text{Ran}(\sigma|_{\text{fv}(\phi)}) \subseteq \mathcal{T}(\mathcal{G})$, $\mathcal{M} \models \sigma(\phi)$, $\mathcal{M} \models \theta(\psi)$, and $t_{i}\sigma \xrightarrow{R_{i}} s_{i}\theta$. Then, it follows from Proposition A.1 that the constraint $\phi \land \psi \land \left( \bigwedge_{i \in \{j \mid s_{j}, t_{j} \in \mathcal{T}(\mathcal{G}, \mathcal{V}), \text{Var}(t_{j}) \subseteq \text{fv}(\phi) \}} t_{i} \simeq s_{i} \right)$ is satisfiable w.r.t. $\mathcal{M}$. Thus, we suppose that $t_{i} \in \mathcal{T}(\mathcal{G}, \mathcal{V}), s_{i} \notin \mathcal{T}(\mathcal{G}, \mathcal{V})$ and $\text{Var}(t_{i}) \subseteq \text{fv}(\phi)$ for some $i$. It follows from $\text{Var}(t_{i}) \subseteq \text{fv}(\phi)$ that $t_{i}\sigma \in \mathcal{T}(\mathcal{G})$. Since $R$ is locally sound for $\mathcal{M}$, we have $t_{i}\sigma \xrightarrow{R_{i}} s_{i}\sigma \in \mathcal{T}(\mathcal{G})$, and hence $s_{i} \in \mathcal{T}(\mathcal{G}, \mathcal{V})$. This contradicts $s_{i} \notin \mathcal{T}(\mathcal{G}, \mathcal{V})$.

Checking the conditions in Theorem E.3 is decidable if the satisfiability of constraints w.r.t. $\mathcal{M}$ is decidable. Thus, removing fake arcs from EDGs by Theorem E.3 is computable in that case.

---

7Ren($t'$) is a term obtained by linearizing the term $t'$. Cap is defined as Cap($x$) = $x$, Cap($g(t_{1}, \ldots, t_{n})$) = $g($Cap($t_{1}$), $\ldots$, Cap($t_{n}$)) if $g$ is not a defined symbol of $R$, and otherwise, Cap($f(t_{1}, \ldots, t_{n})$) = $y \in \mathcal{V}$ [Art00].
Example E.4. Due to Theorem E.3, it is possible to obtain from $EDG(R_t)$ the same graph as $DG(R_t)$.

Another solution for the problem shown in Example E.5 is to extend the definition of unification, considering constraints [Fal08, Fal09]. However, we did not take such an approach due to the use of $Ren$ and $Cap$ in the definition of EDGs.

Example E.5. Consider the following constrained TRS $R_{\text{minus}}$.

$$R_{\text{minus}} = \begin{cases} 
    \text{minus}(x, y) & \rightarrow x \\
    \text{minus}(x, y) & \rightarrow p(\text{minus}(x, p(y))) \\
    \text{minus}(x, y) & \rightarrow s(\text{minus}(x, s(y)))
\end{cases}.$$

The dependency pairs of $R_{\text{minus}}$ are as follows:

$$DP(R_{\text{minus}}) = \{ \begin{array}{c}
(20) \text{minus}^{+}(x, y) \to \text{minus}^{+}(x, p(y)) \iff y \geq 0 \\
(21) \text{minus}^{+}(x, y) \to \text{minus}^{+}(x, s(y)) \iff 0 > y \end{array} \}.$$

$EDG(R_{\text{minus}})$ has the SCC $\{(20), (21)\}$. It is impossible to reduce the initial DP problem $\{(20), (21)\}, R_{\text{minus}}$ by the DP processors in Definition 4.13 or Definition C.4, although there is no strictly infinite minimal $\{(20), (21), R_{\text{minus}}\}$-chain. Thus, we cannot prove termination of $R_{\text{minus}}$, by using $EDG(R_{\text{minus}})$.

On the other hand, from Theorem E.3, we can know that there are no arcs between (20) and (21). Thus, from $EDG(R_{\text{minus}})$, we can obtain the same graph as $DG(R_{\text{minus}})$. Since $\{(20), (21)\}$ is not an SCC in the graph, the initial DP problem is reduced to the set $\{(20), R_{\text{minus}}\}, \{(21), R_{\text{minus}}\}$. Moreover, the DP problems $\{(20), R_{\text{minus}}\}$ and $\{(21), R_{\text{minus}}\}$ are reduced by the DP processor in Definition 4.13 to $\{(\emptyset, R_{\text{minus}})\}$. Therefore, we succeed in proving termination of $R_{\text{minus}}$.

Appendix F. Experimental Results

We implemented the method in this paper in SML/NJ as follows.

- The implementation employs the SMT solver Yices [Dut06] as a constraint solver for linear arithmetics over integers.
- We implemented the subterm criterion technique and the method in Theorem 4.11.
- The cycle-based framework (Theorem 4.6 (1)) is implemented but the DP processor framework is not. The DP processor framework will be implemented in the near future.
- The implementation checks the non-termination criterion for dependency pairs before proving termination.
- Given a domain of coefficients, the module for Theorem 4.11 tries all valuations for $K$ and $K_0$, for each of which
  (1) the module checks whether $A1$ and $A2$ are satisfied or not, and
  (2) if so, the module checks the validity of $(B1) \land (B2) \land (C)$ by using the constraint solver.
- For each cycle in the refined EDGs, the main module checks non-existence of strongly infinite minimal chains by the following ways:

8The prototype implementation, named Cter, will be available from the following URL:

http://www.trs.cm.is.nagoya-u.ac.jp/cter/
Table 1: the results of our experiments.

|        | # of cycles | max of $|K|$ | w/o disproving | with disproving |
|--------|-------------|-------------|----------------|-----------------|
|        |             |             | the domains $[-k..k]$ of $K$ | the domains of $K$ is $[-1..1]$ |
| $f(R_f)$ | 12 | 4 | SN (0.9) | SN (0.9) | SN (0.9) | SN (0.9) |
| sum    | 4 | 3 | SN (0.4) | SN (0.4) | SN (0.4) | SN (0.4) |
| sum2   | 2 | 8 | SN (5.8) | SN (5.8) | SN (5.8) | SN (5.8) |
| sqsum  | 10 | 2 | SN (0.7) | SN (0.7) | SN (0.7) | SN (0.7) |
| fib    | 4 | 4 | SN (0.4) | SN (0.4) | SN (0.4) | SN (0.4) |
| $R_{\text{plus}}^r$ | 13 | 1 | fail (0.6) | fail (0.9) | fail (2.9) | fail (11.1) |
| $R_{\text{sn}}$ | 12 | 4 | fail (2.0) | fail (15.8) | fail (37.6) | error (37.8) |

The degree of running times in brackets is “seconds”.

(1) call the module for the subterm criterion technique, and
(2) if failed, call the module for Theorem 4.11.

- For an interval $[-k..k]$, the implementation first tries all possibilities of polynomials over $[-(k - 1)\ldots(k - 1)]$ and then tries polynomials with the coefficients $k$ and $-k$.

Table 1 summarizes the results of the experiments for five examples. Examples sum, sum2, sqsum, fib, and $R_{\text{plus}}^r$ are seen in Appendix G. Columns 4–8 show results (“SN”, “$-$SN”, “fail”, or “error”) with average time (sec.) of five trials. “SN” means the success in proving termination, “$-$SN” means the success in disproving termination, “fail” means the failure to prove or disprove termination, and “error” means that the execution halts by a system error (lack of resource). Columns 4–7 show results without checking the non-termination criterion, and Column 8 shows results with checking the non-termination criterion. An example where the implementation failed in proving termination is also seen in Appendix G. The implementation was executed under OS FreeBSD 6.2-RELEASE, on an Intel Pentium III-S Dual CPU at 1.26 GHz and 2 GByte of primary memory. The parameters used in the experiments are as follows:

- the domains of variables in $K$ or $K_0$ are intervals $[-1..1]$, $[-2..2]$, $[-5..5]$, and $[-10..10]$, and
- the timeout is 1 minutes.

Note that the domain $[-1..1]$ corresponds to the restriction of tree homomorphisms to be linear. The upper complexity bound of processes of the module for Theorem 4.11 is $|K|^{2k+1} \times 2^{2dn}$ where $k$ specifies the domain $[-k..k]$, $d$ is a constant, and $n$ is the size of formulas checked in the process. Note that the upper decision complexity bound of $n$-size formulas over linear integer arithmetics is $2^{2dn}$ [Coo72, Opp78].
Appendix G. List of Examples in the Experiments

**sum: summation**

Then function *sum* computes the summation of input integer *n*, that is, \( \text{sum}(n) = \sum_{i=1}^{n} i \) where \( \text{sum}(n) = 0 \) if \( n \leq 0 \).

\[
R_{\text{sum}} = R_{\text{plus}} \cup \begin{cases}
    \text{sum}(x) \rightarrow u(x,s(0),0) \\
    u(x,i,z) \rightarrow u(x,s(i),\text{plus}(z,i)) & \Leftarrow s(x) > i \\
    u(x,i,z) \rightarrow z & \Leftarrow \neg (s(x) > i)
\end{cases}
\]

where

\[
R_{\text{plus}} = \begin{cases}
    \text{plus}(0,y) \rightarrow y \\
    \text{plus}(s(x),y) \rightarrow s(\text{plus}(x,y)) \\
    \text{plus}(p(x),y) \rightarrow p(\text{plus}(x,y)) \\
    s(p(x)) \rightarrow x \\
    p(s(x)) \rightarrow x
\end{cases}
\]

This constrained TRS is obtained from the following C program:

```c
int sum(int x){
    int i = 1, z = 0;
    while( x+1 > i ){
        z += i;
        i ++;
    }
    return z;
}
```

**sum2: variant of sum**

This is a variant of *sum*. The difference between them is how to compute \( z + i \).

\[
R'_{\text{sum}} = \begin{cases}
    \text{sum2}(x) \rightarrow u_1(x,s(0),s(0),0) \\
    u_1(x,i,j,z) \rightarrow u_2(x,i,i,z) & \Leftarrow s(x) > i \\
    u_1(x,i,j,z) \rightarrow z & \Leftarrow \neg (s(x) > i) \\
    u_2(x,i,j,z) \rightarrow u_2(x,i,s(j),s(z)) & \Leftarrow s(x) > j \\
    u_2(x,i,j,z) \rightarrow u_1(x,s(i),j,z) & \Leftarrow \neg (s(x) > j) \\
    s(p(x)) \rightarrow x \\
    p(s(x)) \rightarrow x
\end{cases}
\]

This constrained TRS is obtained from the following C program:

```c
int sum2(int x){
    int i = 1, j = 1, z = 0;
    while( x+1 > i ){
        j = i;
        while( x+1 > j ){
            z ++;
            j ++;
        }
        i ++;
    }
```
return z;
}

sqsum: square summation

The function \texttt{sqsum} computes $\sum_{i=1}^{n} i^2$ of input integer $n$ where \texttt{sqsum}(n) = 0 if $n \leq 0$.

\[
R_{\text{sqsum}} = R_{\text{plus}} \cup \{
\begin{array}{ll}
\text{sqsum}(x) & \rightarrow u(x, s(0), 0) \\
\text{u}(x, i, z) & \rightarrow u(x, s(i), \text{plus}(z, \text{times}(i, i))) \quad \leftarrow s(x) > i \\
\text{u}(x, i, z) & \rightarrow z \quad \leftarrow \neg(s(x) > i) \\
\text{times}(0, y) & \rightarrow 0 \\
\text{times}(s(x), y) & \rightarrow \text{plus}(\text{times}(x, y), y) \\
\text{times}(p(x), y) & \rightarrow \text{minus}(\text{times}(x, y), y) \\
\text{minus}(x, 0) & \rightarrow x \\
\text{minus}(x, s(y)) & \rightarrow \text{p}(\text{minus}(x, y)) \\
\text{minus}(x, p(y)) & \rightarrow s(\text{minus}(x, y)) \\
\end{array}
\]

This constrained TRS is obtained from the following C program:

```c
int sqsum(int x){
    int i = 1, int z = 0;
    while( x+1 > i ){
        z = z + i * i;
        i ++;
    }
    return z;
}
```

fib: Fibonacci

The function \texttt{fib} computes the $n$-th Fibonacci number for input integer $n$.

\[
R_{\text{fib}} = R_{\text{plus}} \cup \{
\begin{array}{ll}
\text{fib}(x) & \rightarrow u(x, s(0), 0, s(0)) \\
\text{u}(x, i, y, z) & \rightarrow u(x, s(i), z, \text{plus}(z, y)) \quad \leftarrow s(x) > i \\
\text{u}(x, i, y, z) & \rightarrow z \quad \leftarrow \neg(s(x) > i) \\
\end{array}
\]

This constrained TRS is obtained from the following C program:

```c
int fib(int x){
    int i = 1, j = 0, z = 1, tmp = 0;
    while( x+1 > i ){
        tmp = y;
        y = z;
        z += tmp;
    }
    return z;
}
```
Variant of addition

This is a variant of $R_{\text{plus}}$. A difference between them is overlapping of rules.

$$R'_{\text{plus}} = \begin{cases} 
\text{plus}(x, y) & \rightarrow 0 \\
\text{plus}(x, y) & \rightarrow s(\text{plus}(p(x), y)) \iff x > 0 \\
\text{plus}(x, y) & \rightarrow p(\text{plus}(s(x), y)) \iff 0 > x \\
\text{plus}(x, y) & \rightarrow s(\text{plus}(x, p(y))) \iff y > 0 \\
\text{plus}(x, y) & \rightarrow p(\text{plus}(x, s(y))) \iff 0 > y 
\end{cases}$$

The dependency pairs of $R'_{\text{plus}}$ are as follows:

$$DP(R'_{\text{plus}}) = \begin{cases} 
(22) & \text{plus}^{\sharp}(x, y) \rightarrow \text{plus}^{\sharp}(p(x), y) \iff x > 0 \\
(23) & \text{plus}^{\sharp}(x, y) \rightarrow \text{plus}^{\sharp}(s(x), y) \iff 0 > x \\
(24) & \text{plus}^{\sharp}(x, y) \rightarrow \text{plus}^{\sharp}(x, p(y)) \iff y > 0 \\
(25) & \text{plus}^{\sharp}(x, y) \rightarrow \text{plus}^{\sharp}(x, s(y)) \iff 0 > y 
\end{cases}.$$  

Consider the SCC $\{(22), (23), (24), (25)\}$. The non-existence of strictly infinite minimal $\{(22), (23), (24), (25)\}, R'_{\text{plus}}$-chain cannot be proved by Theorem 4.11. In every chain, if the pair (22) appears, then (23) never appears after the occurrence of (22). Thus, there is no strictly infinite $\{(22), (23), (24), (25)\}, R'_{\text{plus}}$-chain. However, our approach cannot capture this observation.

References