社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

左線形シャローなどの項書換え系の停止性の決定性

王　　易[†]　酒井　正彦[††]　西田　直樹[††]　草刈　圭一朗[††]　坂部　俊樹[††]

[†,††] 名古屋大学大学院情報科学研究科
〒 464-8603 名古屋市千種区不老町
E-mail: †ywang80@trs.cm.is.nagoya-u.ac.jp, ††{sakai,nishida,kusakari,sakabe}@is.nagoya-u.ac.jp

あらまし　本稿では、左線形シャロー項書換え系の停止性が決定可能であることを示す．本手法の手続きは依存グラフにおける引数の伝播の分析と，木オートマトン技術による項の到達可能性の検証で構成される．また，提案する手法が他のクラスにも適用可能かを検討する．

キーワード　依存対，グローイング項書換え系，木オートマトン，決定手続き

Decidability of Termination for Left-Linear Shallow Term Rewriting Systems and Related

Yi WANG[†], Masahiko SAKAI[††], Naoki NISHIDA[††],
Keiichiro KUSAKARI[††], and Toshiki SAKABE[††]

[†,††] Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8603 Japan
E-mail: †ywang80@trs.cm.is.nagoya-u.ac.jp, ††{sakai,nishida,kusakari,sakabe}@is.nagoya-u.ac.jp

Abstract　In this paper, we show that the termination is decidable for left-linear shallow term rewriting systems. The decision procedure consists of the analysis of argument propagation in the dependency graph and the reachability between two terms that is verified by using tree automata techniques. We also explore the potentiality of this method to other rewriting systems such as growing TRSs.

Key words　dependency pair, growing TRSs, tree automata, decision procedure

1. Introduction

Termination is one of the central properties of term rewriting systems (TRSs for short). A TRS terminates if it does not admit any infinite rewrite sequences; conversely, it does not terminate if admitting an infinite rewrite sequence. Termination guarantees that any expression cannot be infinitely rewritten, and hence, the existence of a normal form for it. As we go from simple to more general classes of term rewriting systems, the complexity of deciding termination increases until it becomes undecidable. For example, while in some strongly restricted case, termination becomes a decidable property, such as right-ground TRSs [4], which is the generalization of Huet and Lankford's result [3] for ground rewrite cases. Termination is an undecidable property for general TRSs, this is true even if one allows for only unary function symbols in the rules, or for only one rewrite rule. Therefore, it is fruitful to identify the decidability barrier and study decidability issues for some intermediate classes, especially if these classes are expressive enough to capture interesting rules.

The class of shallow TRS, since it has been proposed just a few years before, is attracting some interests from researchers. A TRS is called shallow if in each rule $l \to r$, if all variables in l, r occur at depth 0 or 1. While the decidability problems of reachability, joinability, confluence for the shallow cases are being well studied recently, its decidability property of termination requires further investigation. In year 2005, the affirmative result for right-linear shallow case was shown by Godoy and Tiwari [7]. In this paper, we give a result on the decidability of termination for left-linear shallow TRSs and present a practical decision procedure. Moreover, to show the potentiality of our method, we extend our method not only to cover the right-linear shallow cases but also to a more general class of growing TRSs. Since Nagaya and Toyama [10] obtained the decidability result for almost orthogonal growing TRSs, we will show by examples that our method is capable of tackling a different range of growing rewriting systems. What is more important is that our method can actually find an infinite rewriting sequence if R does not terminate.

Our proof relies on the decidability of reachability that can be verified by using tree automata techniques [1]. Concerning automated termination proofs for term rewriting systems, the dependency pair approach proposed in the reference [6] is one of the most powerful techniques. Based on the notion of dependency pair, by doing a transformation on the dependency graph, our method gathers all the essential conditions for the existence of an infinite dependency chain which implies the non-termination of TRS.

The organization of this paper is as follows: in section 2, we review the preliminary definitions of term rewriting systems; in section 3, we recall the definitions and results concerning dependency pair approach and tree automata techniques; in section 4, we give the detailed termination decision proof for left-linear shallow TRSs; in section 5, we extend our method to other term rewriting cases and explain them respectively; in section 6, we compare our method with two existing results related.

2. Preliminaries

We assume the reader is familiar with the standard definitions of term rewriting systems [2] and here we just review the main notations used in this paper.

A *signature* $\mathcal{F}$ is a set of function symbols, where every $f \in \mathcal{F}$ is associated with a non-negative integer number by an arity function: $arity\colon \mathcal{F} \to \mathbb{N}(=\{0,1,2,\ldots\})$. Function symbols of arity 0 are called *constant symbols*. The set $\mathcal{T}(\mathcal{F},\mathcal{V})$ of all *terms* built from a signature $\mathcal{F}$ and a countable infinite set $\mathcal{V}$ of *variables* such that $\mathcal{F}\cap\mathcal{V}=\emptyset$, is represented by $\mathcal{T}(\mathcal{F},\mathcal{V})$. The set of *ground terms* is denoted by $\mathcal{T}(\mathcal{F},\emptyset)$ ($\mathcal{T}(\mathcal{F})$ for short). We write $s=t$ when two terms s and t are identical. The *root symbol* of a term t is denoted by $root(t)$.

The set of all *positions* in a term t is denoted by $\mathcal{P}os(t)$. Here ε represents the root position. Positions are partially ordered by the prefix order $\leqq$, that is, $p \leqq q$ if there exists an r such that $p\cdot r = q$. We write $p < q$ if $p \leqq q$ and $p \neq q$. If $p \in \mathcal{P}os(t)$, then $t|_p$ denotes the *subterm* of t at position p, and $t[s]_p$ denotes the term that is obtained from t by replacing the subterm at position p by the term s. We denote by $\trianglerighteq$ the *subterm ordering*, that is, $t \trianglerighteq u$ if u is a subterm of t, and $t \triangleright u$ if $t \trianglerighteq u$ and $t \neq u$. The *height* of a term t is 0 if t is a variable or a constant, and $1 + max(\{height(s_i) \mid i \in \{1,\ldots,m\}\})$ if $t = f(s_1,\ldots,s_m)$.

Let C be a *context* with a hole $\Box$. We write $C[t]$ for the term obtained from C by replacing $\Box$ with a term t.

A *substitution* θ is a mapping from $\mathcal{V}$ to $\mathcal{T}(\mathcal{F},\mathcal{V})$ such that the set $\{x \in \mathcal{V} \mid \theta(x) \neq x\}$ called the *domain* of θ and denoted by $\mathcal{D}om(\theta)$ is finite. We usually identify a substitution θ with the set $\{x \mapsto \theta(x) \mid x \in \mathcal{D}om(\theta)\}$ of variable bindings. We denote $\mathcal{R}an(\theta)=\{\theta(x) \mid x \in \mathcal{D}om(\theta)\}$ as the *range* of θ. Substitutions are naturally extended to homomorphisms from $\mathcal{T}(\mathcal{F},\mathcal{V})$ to $\mathcal{T}(\mathcal{F},\mathcal{V})$. In the following, we write $t\theta$ instead of $\theta(t)$. The composition $\theta_1\theta_2$ of two substitutions θ_1 and θ_2 is defined by $x(\theta_1\theta_2) = (x\theta_1)\theta_2$ for all $x \in \mathcal{V}$.

A *rewrite rule* $l \to r$ is a directed equation which satisfies $l \notin \mathcal{V}$ and $\mathcal{V}ar(r)\subseteqq\mathcal{V}ar(l)$. We call l the *left-hand side* and r the *right-hand side* of the rewrite rule. The *reverse* of the rewrite rule $l \to r$ is $r \to l$. A *term rewriting system* TRS is a set of finite rewrite rules. If the two conditions $l \notin \mathcal{V}$ and $\mathcal{V}ar(r)\subseteqq\mathcal{V}ar(l)$ are not imposed, then we call it *extended TRS* (eTRS). In this paper, we will use eTRS implicitly for convenience. The *rewrite relation* $\to_R \subseteqq \mathcal{T}(\mathcal{F},\mathcal{V}) \times \mathcal{T}(\mathcal{F},\mathcal{V})$ associated with a TRS R is defined as follows: $s \to_R t$ if there exist a rewrite rule $l \to r \in R$, a substitution θ, and a position $p \in \mathcal{P}os(s)$ such that $s|_p = l\theta$ and $t = s[r\theta]_p$. The subterm $l\theta$ of s is called a *redex* and we say that s rewrites to t by contracting redex $l\theta$. We say that p is a redex position. The transitive closure of $\to_R$ is denoted by $\to_R^+$. The transitive and reflexive closure of $\to_R$ is denoted by $\to_R^*$. If $s \to_R^* t$, then we say that there is a *rewrite sequence* starting from s, which reduces to t or t is reachable from s by R. We use $\xrightarrow{\varepsilon}_R$ to denote the root rewrite step and $\xrightarrow{>\varepsilon}_R$ to denote the rewrite step where redex position occurs below the root. A term without redexes is called a *normal form*. We say that a term t *has* a normal form if there exists a rewrite sequence starting from t that reduces to a normal form.

A rewrite rule $l \to r$ is called *left-linear* (resp. *right-linear*) if no variable occurs twice in l (resp. r). It is called *linear* if it is both left- and right-linear. A TRS is called left-linear (resp. right-linear, resp. linear) if all of its rules are left-linear (resp. right-linear, resp. linear).

For a TRS R, a term $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ *terminates* if there is no infinite rewrite sequences starting from t. R terminates over $T\subseteqq\mathcal{T}(\mathcal{F},\mathcal{V})$ if all terms in T terminate. In the case that $T = \mathcal{T}(\mathcal{F},\mathcal{V})$, we say that R terminates.

For a TRS R, a function symbol $f \in \mathcal{F}$ is a *defined symbol* of R if $f = root(l)$ for some rewrite rule $l \to r \in R$. The set of all the defined symbols of R is denoted by $D_R = \{root(l) \mid \exists l \to r \in R\}$. We write C_R for the set of all the *constructor symbols* of R which is defined as $\mathcal{F}\backslash D_R$. A term t has a *defined root symbol* if $root(t) \in D_R$.

3. Dependency pairs and tree automata

In this section, we recall the definition of the dependency pair approach [5,6] and the results concerning tree automata techniques that will be used for testing reachability.

Let R be a TRS over a signature $\mathcal{F}$. $\mathcal{F}^\sharp$ denotes the union of $\mathcal{F}$ and $D_R^\sharp = \{f^\sharp \mid f \in D_R\}$ where $\mathcal{F} \cap D_R^\sharp = \emptyset$ and $f^\sharp$ has the same arity as f. We call these new symbols *dependency pair symbols.* Given a term $t = f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ with f defined, we write $t^\sharp$ for the term $f^\sharp(t_1, \ldots, t_n)$. If $l \to r \in R$ and u is a subterm of r with a defined root symbol, then the rewrite rule $l^\sharp \to u^\sharp$ is called a *dependency pair* of R. The set of all dependency pairs of R is denoted by $\mathrm{DP}(R)$. For any subset $T \subseteqq \mathcal{T}(\mathcal{F}, \mathcal{V})$ consisting of terms with a defined root symbol, we denote the set $\{t^\sharp \mid t \in T\}$ by $T^\sharp$.

For a TRS R, a (possibly infinite) sequence of dependency pairs $s_1^\sharp \to t_1^\sharp$, $s_2^\sharp \to t_2^\sharp$, $\ldots$ is a *dependency chain* if there exist substitutions $\tau_1, \tau_2, \ldots$ such that $t_i^\sharp \tau_i \to_R^* s_{i+1}^\sharp \tau_{i+1}$ holds for every two consecutive dependency pairs $s_i^\sharp \to t_i^\sharp$ and $s_{i+1}^\sharp \to t_{i+1}^\sharp$ in the sequence.

Example 3.1 For TRS R_2 in Example 4.2, $D_{R_2} = \{f, g\}$, $\mathrm{DP}(R_2) = \{\, f^\sharp(x, y, z) \to g^\sharp(x, y, z),\ g^\sharp(u, x, v) \to f^\sharp(x, x, x) \,\}$.

Theorem 3.2 ([5, 6]) *For a non-extended TRS R, R does not terminate if and only if there exists an infinite dependency chain.*

The nodes of the *dependency graph* $\mathrm{DG}(R)$ are the dependency pairs of R and there is an arrow from a pair $s^\sharp \to t^\sharp$ to $u^\sharp \to v^\sharp$ if and only if there exist substitutions σ and τ such that $t^\sharp \sigma \to_R^* u^\sharp \tau$. Note that the dependency graph is not computable in general. Several approximation are worked out to compute the estimated dependency graphs, each of which consists the dependency graph of R as a subgraph. Some details can be found in [1]. In this paper, we generally denote approximation of dependency graph as $\mathrm{EDG}(R)$ for any TRS R. We write SCS for a strongly connected subgraph of DG(R) or EDG(R). [1]

Given a directed graph, a (possibly infinite) *backward path* is a path of the form $nd_1 \leftarrow nd_2 \leftarrow \cdots \leftarrow nd_i \leftarrow nd_{i+1} \leftarrow \cdots$ in the graph such that for every two consecutive nodes nd_i and nd_{i+1} there is an arrow from nd_{i+1} to nd_i.

Next, we give a result concerning tree automata.

Proposition 3.3 ([1, 9, 10]) *For a left-linear growing (shallow) TRS R, let $t_0, t_1, \ldots, t_n$ to be ground terms and denote $(\to_R^*)[L] = \{s \in \mathcal{T}(\mathcal{F}) \mid s \to_R^* t,\ t \in L\}$, then the following problems are decidable:*

(1) $\bigcap_{1 \leqq i \leqq n} (\to_R^*)[\{t_i\}] \neq \emptyset\,$?

(2) $\bigcap_{1 \leqq i \leqq n} (\to_R^*)[\{t_i\}] \cap \{t_0\} \neq \emptyset\,$?

[1] A graph is said to be strongly connected if each node is reachable from an arbitrary node.

4. Termination of left-linear shallow TRSs

In this section, we first describe the definition of shallow TRSs, then we prove the decidability of termination for left-linear shallow TRSs.

Definition 4.1 (Shallow TRS) *A TRS R is called shallow, if all its rewrite rules have the form $f(t_1, \ldots, t_n) \to g(s_1, \ldots, s_m)$, $x \to g(s_1, \ldots, s_m)$ or $f(t_1, \ldots, t_n) \to x$ where every t_i and s_i is either a variable of $\mathcal{V}$ or a ground term of $\mathcal{T}(\mathcal{F})$, and where $x \in \mathcal{V}$, and n, m can be* 0 *(if f or g have arity* 0*).*

Example 4.2 Let s, t, u, v be ground terms, the TRSs $R_1 = \{\, f(t, x, y) \to g(u, x, y),\ g(x, y, t) \to h(x, y),\ h(y, s) \to f(y, y, u) \,\}$, $R_2 = \{\, f(x, y, z) \to g(x, y, z),\ g(u, x, v) \to f(x, x, x) \,\}$ and $R_3 = \{\, f(x, y) \to f(u, y),\ f(u, z) \to f(z, v) \,\}$ are left-linear shallow.

Definition 4.3 (Labeling Function) *Let R be a shallow TRS and p be a backward path in EDG(R) of the form $nd_1 \leftarrow nd_2 \leftarrow \cdots \leftarrow nd_i \leftarrow nd_{i+1} \leftarrow \cdots$, and M be the maximum arity of the root symbol of the left-hand side of all the nodes in EDG(R). A labeling function $L_p \colon \mathbb{N} \to DP(R) \times \mathcal{P}(\mathcal{T}(\mathcal{F}))^M$ is defined as follows:*

(1) Denote nd_1 as $\langle f^\sharp(t_1, \ldots, t_n),\ g^\sharp(s_1, \ldots, s_m)\rangle$: $L_p(1) := (nd_1, S_1^1, \ldots, S_M^1)$ where if $1 \leqq i \leqq n$ and $t_i \notin \mathcal{V}$, then $S_i^1 := \{t_i\}$; else $S_i^1 := \emptyset$.

(2) Denote nd_i and nd_{i+1} as $\langle f^\sharp(t_1, \ldots, t_n),\ g^\sharp(s_1, \ldots, s_m)\rangle$ and $\langle h^\sharp(v_1, \ldots, v_k),\ f^\sharp(u_1, \ldots, u_n)\rangle$, and let $L_p(i) := (nd_i, S_1^i, \ldots, S_M^i)$: $L_p(i+1) := (nd_{i+1}, S_1^{i+1}, \ldots, S_M^{i+1})$ where if $1 \leqq j \leqq k$ and $v_j \notin \mathcal{V}$, then $S_j^{i+1} := \{v_j\}$; else if $1 \leqq j \leqq k$ and $\Lambda = \{l \in \{1, \ldots, n\} \mid v_j = u_l$ and $u_l \in \mathcal{V}\}$ is not empty, then $S_j^{i+1} := \bigcup_{l \in \Lambda} S_l^i$; else $S_j^{i+1} := \emptyset$.

Definition 4.4 (Argument Propagation Cycling) *Let R be a shallow TRS and p be a backward path in EDG(R) of the form $nd_1 \leftarrow nd_2 \leftarrow \cdots \leftarrow nd_i \leftarrow nd_{i+1} \leftarrow \cdots$. An argument propagation cycling (APC for short) is a finite sequence of labels of the form $L_p(i), L_p(i+1), \ldots, L_p(j)$ such that $L_p(i) = L_p(j)$.*

We say an APC is minimal *if all its proper subsequences are not APC.*

Lemma 4.5 *For left-linear shallow TRSs, the set $\mathcal{A}$ of all minimal APCs is finite and computable.*

Proof. For any left-linear shallow TRS R, observe that the number of nodes in $\mathrm{EDG}(R)$ are finite and since the number of all the ground terms at depth 1 in the left-hand side of all the nodes are finite, the value of S_k in the labeling function is finite set too. Thus, the union of the ranges of labels $L_p(\cdot)$ for all backward paths is finite. It follows from the minimality that $\mathcal{A}$ is finite.

$\mathcal{A}$ can be computed by the following procedure: generate all the backward paths starting from each node in EDG(R) while doing labeling synchronically, stop at the point when an APC is found for the first time and output this APC.

The soundness of this procedure is obvious as the procedure finds 'APCs' satisfying the Definition 4.4 and minimality. The completeness of this procedure can be argued as follows: if there is a minimal APC $L_p(i), \ldots, L_p(j)$ that is not in the output of the procedure, we can assume that the backward path corresponding to this APC is of the form $nd_1 \leftarrow \cdots \leftarrow nd_i \leftarrow \cdots \leftarrow nd_j \leftarrow \cdots$ such that no subsequences of $L_p(1), \ldots, L_p(j-1)$ are APCs. The existence of such a backward path can be shown by a study on the case if there is a APC $L_p(i'), \ldots, L_p(j')$ such that $j' \leqq i$ and the case if $i' < i < j' < j$. Thus, it is easy to know that this minimal APC will be found by the checking along the backward path above starting from node nd_1 in the procedure. It leads to a contradiction. □

We stress that for any minimal APC with length L in $\mathcal{A}$, there are $L-1$ minimal APCs *homogeneous* to it, here *homogeneous* is in the sense that for termination determination, they can be viewed as one case since they are from the same 'cycle'.

Definition 4.6 (Smooth APC) *Let R be a shallow TRS, an ACP of the form $L_p(x), \ldots, L_p(y)$ is smooth if the following conditions are satisfied:*

For $x \leqq i \leqq y$, let $nd_i = \langle f^\sharp(t_1, \ldots, t_n),\ g^\sharp(s_1, \ldots, s_m)\rangle$ and $nd_{i+1} = \langle h^\sharp(v_1, \ldots, v_k),\ f^\sharp(u_1, \ldots, u_n)\rangle$. For $1 \leqq j \leqq n$:

(1) If $u_j \in \mathcal{V}$ and $t_j \in \mathcal{V}$, then $\bigcap_{t \in S_j^i} (\rightarrow_R^*)[\{t\}] \neq \emptyset$.

(2) If $u_j \in \mathcal{T}(\mathcal{F})$ and $t_j \in \mathcal{V}$, then $\bigcap_{t \in S_j^i} (\rightarrow_R^*)[\{t\}] \cap \{u_j\} \neq \emptyset$.

(3) If $u_j \in \mathcal{V}$ and $t_j \in \mathcal{T}(\mathcal{F})$, then true.

(4) If $u_j \in \mathcal{T}(\mathcal{F})$ and $t_j \in \mathcal{T}(\mathcal{F})$, then $(\rightarrow_R^*)[\{t_j\}] \cap \{u_j\} \neq \emptyset$.

Lemma 4.7 *For a left-linear shallow R, there exists an infinite dependency chain if and only if there is a minimal smooth APC.*

Proof. We firstly prove the *if* part, the condition clauses of (1)–(4) in Definition 4.6 exhaust all the cases, observe that left-linear shallow TRSs only allow variable or ground term arguments at depth 1. Therefore, we can choose an ground instance of the right-hand side of nd_y that satisfies conditions (1)–(4), starting from which reversely to nd_x, a dependency chain can be constructed and it will be equal to the instance again when reaching node nd_x. Thus the dependency chain loops and does not terminate.

The *only if* part can be argued based on the fact that, in any infinite dependency chain corresponding to an SCS in the dependency graph, the property of left-linear shallow guarantees the existence of a corresponding 'transformed' infinite dependency chain by delaying the R-reductions at each position of the arguments at depth 1 in the right-hand side of the node if the argument at the corresponding position of the left-hand side of its successor node is a variable, and just copying the value to that position, until it becomes a ground term, then doing all the R-reductions including delayed ones to this ground term. Also notice that it will finally return back to the starting node with the same ground instance of the right-hand side. Thus, by considering the definition of minimal smooth APC, we can know that any 'transformed' dependency chain is corresponding to a minimal smooth APC. □

Theorem 4.8 *Termination of left-linear shallow TRSs are decidable.*

Proof. For any left-linear shallow TRS R, we can assume R is non-extended, otherwise R does not terminate. The decision procedure turns to be computing $\mathcal{A}$ first, and then checking the existence of any minimal smooth APC. Since from Lemma 4.5, $\mathcal{A}$ is computable, hence the theorem follows from Theorem 3.2, Proposition 3.3 and Lemma 4.7. □

5. Extension

In this section, we explore the potentiality of this method to other rewriting systems such as growing TRSs. First of all, we give the definition of growing TRSs.

Definition 5.1 (Growing TRS) *A rewrite rule $l \rightarrow r$ is growing if all variables in $Var(l) \cap Var(r)$ occur at depth 0 or 1 in l. A TRS R is growing if every rewrite rule in R is growing and R is rev-growing if the reverse of every rule is growing.*

Example 5.2 TRS $R_4 = \{\, f(u,x) \rightarrow g(x,t),\ g(x,y) \rightarrow h(x, p(x,y)),\ h(s,x) \rightarrow f(x,x) \,\}$ are left-linear growing TRSs, where s, t, u are ground terms of $\mathcal{T}(\mathcal{F})$ and p is a function symbol of arity 2.

The possible extensions of our method are shown in the table below, where the second column presents the condition that TRS R should satisfy, the third column gives the condition that DP(R) should satisfy.

No.	Term Rewriting System	Dependency Pair
1	left-linear shallow	–
2	right-linear shallow	–
3	left-linear growing	shallow
4	right-linear rev-growing	shallow
5	left-linear growing	right-linear
6	right-linear rev-growing	left-linear

Notice that (4) implies (2). We will explain all these extensions respectively.

(1) Theorem 4.8.

(2) **Theorem 5.3** *Termination of right-linear shallow TRSs are decidable.*

Proof. (Sketch) For right-linear shallow TRS R, DP(R) is right-linear shallow too. Notice that $(\rightarrow^*_{R^{-1}})[L]$ is regular, we simply reverse the labeling direction and our method will work well for this case. □

(3) **Theorem 5.4** *Termination of left-linear growing TRSs with its dependency pairs being shallow are decidable.*

Proof. (Sketch) For left-linear growing TRS R with DP(R) being shallow, notice it is allowed to do arguments propagation in exactly the same way as the one for left-linear shallow TRS case and $(\rightarrow^*_{R^{-1}})[L]$ is regular too, thus our decision method is applicable for this case.

(4) **Theorem 5.5** *Termination of right-linear rev-growing TRSs with its dependency pairs being shallow are decidable.*

Proof. (Sketch) For right-linear rev-growing TRS R with DP(R) being shallow, by reversing the labeling direction, it can be argued the same way as for the case of left-linear growing TRS R with DP(R) being shallow.

(5) **Conjecture 5.6** *Termination of left-linear growing TRSs with its dependency pairs being right-linear are decidable.*

(6) **Conjecture 5.7** *Termination of right-linear rev-growing TRSs with its dependency pairs being left-linear are decidable.*

6. Comparison

In this section, we compare our result with two existing results related with shallow and growing TRSs.

(1) Godoy and Tiwari [7] gave a result for right-linear shallow cases which is one of the solvable classes by our method.

(2) Nagaya and Toyama [10] obtained the decidability result for almost orthogonal growing TRSs. We claim that the applicable classes of Nagoya's and ours do not cover each other. Considering examples R_3 and R_4: R_3 is left-linear shallow, since there is a non-trivial critical pair $\langle f(u,z), f(z,v)\rangle$, R_3 is not almost orthogonal; DP(R_4) is neither shallow nor right-linear, since all its critical pairs are trivial overlays, R_4 is almost orthogonal.

7. Conclusion

In this paper we have shown that termination is a decidable property for left-linear shallow TRSs and hence we can automatically prove termination or non-termination of any left-linear shallow TRSs. We also showed that our method are applicable for more general cases as listed in the table in Section 5.

Based on the notion of dependency pair, we accomplished our proof by argument propagation on its estimated dependency graph. Then the conditions for the existence of an infinite dependency chain are being checked by tree automata techniques. According to the comparison with two existing results, we showed that this method can tackle a new field of rewriting systems which can not be solved by the existing ways. It is worthwhile to do further studies on the extension of this method, left-linear growing case for instance.

Acknowledgment This work is partly supported by MEXT. KAKENHI #15500007 and #16300005.

References

[1] Aart Middeldorp. Approximating Dependency Graphs using Tree Automata Techniques, Proceedings of the International Joint Conference on Automated Reasoning(IJCAR'01), volume 2083 of LNAI, pages 593–610, 2001.

[2] Franz Baader and Tobias Nipkow. *Term Rewriting and All That*, Cambridge University Press, 1998.

[3] Gérard Huet and Dallas Lankford. On the uniform halting problem for term rewriting systems, Technical Report 283, IRIA, 1978.

[4] Nachum Dershowitz. Termination of Linear Rewriting Systems, in S. Even and O. Kariv, editors, Automata, Languages and Programming, volume 115 of LNCS, pages 448–458, Springer-Verlag, 1981.

[5] Nao Hirokawa and Aart Middeldorp. Dependency Pairs Revisited, Proceedings of the 15th International Conference on Rewriting Techniques and Applications, Aachen, volume 3091 of LNCS, pages 249–268, 2004.

[6] Thomas Arts and Jürgen Giesl. Termination of Term Rewriting Using Dependency Pairs, *Theoretical Computer Science*, pages 133–178, 2000.

[7] Guillem Godoy and Ashish Tiwari. Termination of Rewrite Systems with Shallow Right-Linear, Collapsing and Right-Ground Rules, 20th Intl. Conf. on Automated Deduction, CADE, LNAI 3632, pages 164–176, 2005.

[8] Florent Jacquemard. Reachability and confluence are undecidable for flat term rewriting systems, Research Report

LSV-03-6, March 2003.

[9] Hubert Comon, Max Dauchet, Rmi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison and Marc Tommasi. Tree automata techniques and applications, 1999.

[10] Takashi Nagaya and Yoshihito Toyama. Decidability for Left-Linear Growing Term Rewriting Systems. In proc. 10th RTA, volume 1631 of LNCS, pages 256–270, 1999.