Master's Thesis

Transducer-Based String Rewrite Systems and Recursive Path Orders

1610151 Chenyang Pan

Supervisor	Nao Hirokawa
Main Examiner	Nao Hirokawa
Examiners	Hajime Ishihara
	Kazuhiro Ogata
	Mizuhito Ogawa

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

March 2019

Abstract

Keywords: infinite string rewriting, recursive path orders

Term rewriting is a well-known computational model which is Turing equivalent to lambda calculus, and has many applications e.g. automated theorem proving and functional programming, etc [8]. A significant property of term rewriting systems, named as convergence consisting of termination and confluence should be ensured in these fields. Termination ensures the finiteness of computation steps and confluence guarantees the uniqueness of result when different computation sequences arise. As a famous undecidable problem, the word problem will be totally unsolvable without convergence.

String Rewriting and Convergence. This thesis talks about a formalization of string rewrite systems with infinite rules and provides a decidable technique for termination proving. String rewrite systems are ordered equations of strings. Convergence means that different rewrite sequences from the same string can lead to same result finally. We hope that this property holds for all strings since convergence make it possible to solve word problems.

Word Problems and Solutions. A major purpose of string rewrite systems is to solve word problems for finitely presented monoids. The word problem for the equations E is described as follows:

Instance: Two strings u, v and axioms E

Question: Does $u = v$ hold under E?

Although the word problem is undecidable, the problem can still be solved by finding a convergent system R such that $\leftrightarrow_R^* = \leftrightarrow_E^*$ holds.

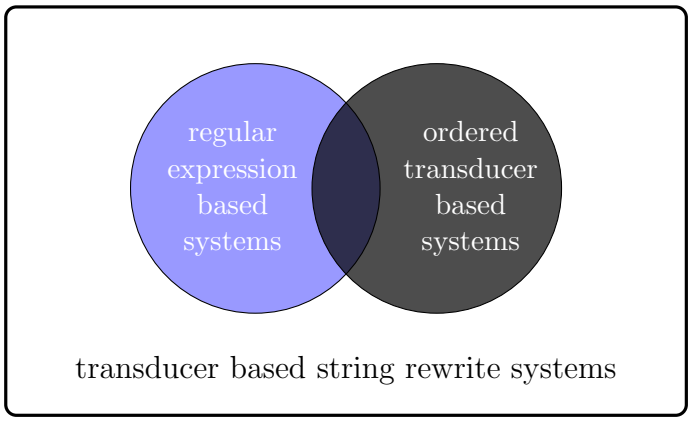
Automated Equational Reasoning. The formal methods to prove equation equivalence presented by word problems have been under research for decades. The technique [7] influenced by the standard completion procedure [6] and the extended version called unfailing completion [1] is used in most modern automated theorem provers.

However, some results [4, 3, 5, 9] in this field show that a finite convergent system of given equations does not always exist, which brings us difficulties when proving that equation equivalence does not hold.

Motivation and Approach. The above consequence gives us the question: If we are given an infinite string rewrite system claimed convergent, does it really have the convergence property?

This thesis uses finite state transducers to formalize these string rewrite systems which especially have infinite rules to check the confluence property. Transducers are a kind of abstract machine that recognizes pairs of strings. We only focus on termination, one of the necessary properties of convergence. Recursive path orders [2] provide a technique to prove termination. Instead of comparing the infinite many rules of given string rewrite systems, we develop a procedure to ensure that the language accepting by a transducer remains true under this order.

Unfortunately, the problem how to check an arbitrary finite transducer remains open. We develop two restricted classes of finite transducers indicated as follows.



Each of them can do string rewriting and check termination with the help of recursive path orders.

Contributions. The main contributions of the thesis are listed as:

- the regular expression based string rewriting,
- the recursive path order over regular expressions,
- the ordered transducer based string rewriting, and
- the recursive path order over ordered transducers.

References

[1] Leo Bachmair, Nachum Dershowitz, and David A. Plaisted. "Completion without Failure". In: *Rewriting Techniques*. Ed. by Hassan Aït-Kaci and Maurice Nivat. 1989, pp. 1–30.

[2] Nachum Dershowitz. "Orderings for Term-Rewriting Systems". In: *Theoretical Computer Science* 17.3 (1982), pp. 279–301.

[3] M. Jantzen. "A Note on A Special One-rule Semi-Thue system". In: *Information Processing Letters* 21.3 (1985), pp. 135–140.

[4] Matthias Jantzen. "On A Special Monoid with A Single Defining Relation". In: *Theoretical Computer Science* 16.1 (1981), pp. 61–73.

[5] Deepak Kapur and Paliath Narendran. "A Finite Thue System with Decidable Word Problem and without Equivalent Finite Canonical System". In: *Theoretical Computer Science* 35 (1985), pp. 337–344.

[6] Donald E. Knuth and Peter B. Bendix. "Simple Word Problems in Universal Algebras". In: *Computational Problems in Abstract Algebra*. 1970, pp. 263–297.

[7] Ursula Martin and Tobias Nipkow. "Ordered Rewriting and Confluence". In: *Proceedings of the Tenth International Conference on Automated Deduction*. 1990, pp. 366–380.

[8] Enno Ohlebusch. *Advanced Topics in Term Rewriting*. 1st. 2010.

[9] Craig C. Squier. "Word Problems and A Homological Finiteness Condition for Monoids". In: *Journal of Pure and Applied Algebra* 49.1 (1987), pp. 201–217.

Contents

Chapter 1

Introduction

Term rewriting is a well-known computational model which is Turing equivalent to lambda calculus, and has many applications e.g. automated theorem proving and functional programming, etc [11]. A significant property of term rewriting systems, named as convergence consisting of termination and confluence should be ensured in these fields. Termination ensures the finiteness of computation steps and confluence guarantees the uniqueness of result when different computation sequences arise. As a famous undecidable problem, the word problem will be totally unsolvable without convergence.

This chapter introduces term rewrite systems, whose terms have a specific structure called strings in an informal way, and show the importance of convergence by presenting a word problem whose decision procedure involves infinity, which leads to the motivation of our research.

String Rewriting and Convergence. This thesis primarily discusses a formalization of string rewrite system with infinite rules and develops a decidable technique for termination proving. String rewrite systems are ordered equations of strings.

Example 1.1. The following system R is an example of convergent string rewrite system:

$$baa \to aba$$
$$bba \to bab$$

The convergence means that different rewrite sequence from the same string can lead to same result finally. For instance, the string $bbaa$ is computed as follows using different rule:

$$baba \;{}_R\!\leftarrow bbaa \to_R bab a$$

The left part of the above sequence replaces the substring baa with aba using the first rule, while the right part uses the second rule. Both of them rewrite to the string $baba$. We hope that this property can hold for all strings since convergence makes it possible to solve word problems.

Word Problems and Solutions. A major purpose of string rewrite systems is to solve word problems for finitely presented monoids. The word problem for the equations E is described as:

Instance: Two strings u, v and axioms E

Question: Does $u = v$ hold under E?

Although the word problem is undecidable, the problem can still be solved by finding a convergent system R such that $\leftrightarrow_R^* = \leftrightarrow_E^*$ holds. Example 1.1 shows the solution for the axioms $\{aba = baa, bba = bab\}$, which is the presentation of bicyclic monoid [5].

Automated Equational Reasoning. The formal methods to prove equation equivalence presented by word problems have been studied for decades. The technique [10] influenced by the standard completion procedure [9] and the extended version called unfailing completion [1] has been used in most modern automated theorem provers. However, some results [7, 6, 8, 12] in this field showed that a finite convergent system of given equations does not always exist, which makes it difficult to prove that equation equivalence does not hold.

Turtle Graphics and Equlivance. Turtle graphics is computational vector graphics which has been used in many related fields. It can be described as a virtual turtle that only knows its location and direction. It follows some simple commands to change either its location or its heading. For example, the commands [go, turn left, go, turn right, go] from the bottom are indicated in Figure 1.1a while the commands [turn right, go, turn right] are indicated in Figure 1.1b.

(a) Turtle 1

(b) Turtle 2

Figure 1.1: Two turtle graphics ending in the same state

Its location can be represented by a point P given by a pair of coordinates (p_1, p_2) and its heading direction can be represented by a vector w. A state of the turtle is denoted by (P, w). We write a as the operation "go forward" and b as "turn left in θ direction". The states of turtle can be compared. Figure 1.1 shows $ababa = bab$ when $\theta = 90°$.

The following system is a convergent system for $\theta = 120°$:

$$R_{120} = \left\{ \begin{array}{rcl} aaa & \to & \varepsilon \\ bab^{n+1}ab & \to & ab^n a \\ ab^m ab^n ab & \to & bab^m ab^n a \end{array} \middle| \; m, n \in \mathbb{N} \right\}$$

Infinity appears in the rules.

Motivation and Approach. The above consequence shows the question: If we are given an infinite string rewrite system claimed convergent, does it really have the convergence property?

This thesis uses finite state transducers to formalize these string rewrite systems which especially have infinite rules to check the confluence property. Transducers are a kind of abstract machine that recognizes pairs of strings. For instance, the following transducer accepts the system R_{120}:

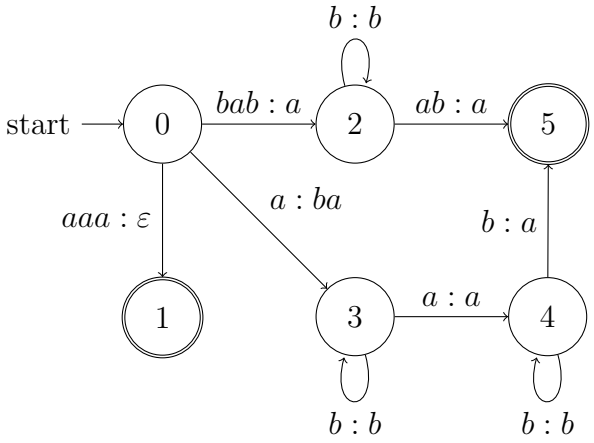


Figure 1.2: The transducer of R_{120}

We only focus on termination, one of the necessary properties of convergence. Recursive path orders [4] can be used to prove termination. Instead of comparing the infinite many rules of given string rewrite systems, we develop

a procedure to ensure that the language accepting by a transducer remains true under this order.

Unfortunately, the problem how to check an arbitrary finite transducer remains open. We develop two restricted classes of finite transducers indicated in Figure 1.3.

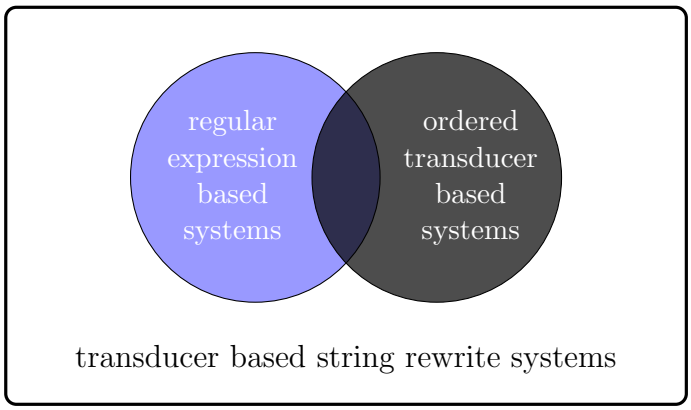


Figure 1.3: Two classes of transducers

Each of them can do rewriting and check the termination property with the help of recursive path orders.

Overview. Chapter 2 introduces some basic mathematical notions and notations which will be used to discuss string rewrite systems. We provide two formalizations of these systems with infinite rules called regular expression based systems and transducer based systems in Chapter 3 and Chapter 4. Each of them develops a formal method to check termination by the assistance of recursive path orders. Finally, we conclude the thesis and the limitations with related work.

Contributions. The main contributions of the thesis are listed as:

- the regular expression based string rewriting,
- the recursive path order over regular expressions,
- the ordered transducer based string rewriting, and
- the recursive path order over ordered transducers.

Chapter 2

Preliminaries

This chapter gives some basic definitions used in the field of string rewriting. We assume here that the reader is familiar with the elementary set theory.

2.1 Relations and Orders

Definition 2.1. A relation R between sets A and B is a *function* from A to B if for each element $a \in A$ there is exactly one element $b \in B$ such that $(a, b) \in R$.

Definition 2.2. Let R be a relation between sets A, B and let S be a relation between sets B, C. The *composition* $R \circ S$ is the following relation between A and C:

$$\{(a, c) \mid (a, b) \in R \text{ and } (b, c) \in S \text{ for some } b \in B\}$$

Definition 2.3. Let R be a relation on a set A.

- R is *reflexive* if $(a, a) \in R$ for all $a \in A$.
- R is *irreflexive* if $(a, a) \notin R$ for all $a \in A$.
- R is *symmetric* if $(a_1, a_2) \in R$ implies $(a_2, a_1) \in R$ for all $a_1, a_2 \in A$.
- R is *asymmetric* if $(a_1, a_2) \in R$ implies $(a_2, a_1) \notin R$ for all $a_1, a_2 \in A$.
- R is *anti-symmetric* if $(a_1, a_2) \in R$ and $(a_2, a_1) \in R$ imply $a_1 = a_2$ for all $a_1, a_2 \in A$.
- R is *transitive* if $(a_1, a_2) \in R$ and $(a_2, a_3) \in R$ implies $(a_1, a_3) \in R$ for all $a_1, a_2 \in A$.

Definition 2.4. Let $\to$ be a relation on a set A. We define these compositions

- $\to^0 = \{(a, a) \mid a \in A\}$
- $\leftarrow = \{(b, a) \mid a \to b\}$
- $\to^{i+1} = \to^i \circ \to$
- *reflexive closure* $\to^= = \to \cup \to^0$
- *symmetric closure* $\leftrightarrow = \leftarrow \cup \to$
- *transitive closure* $\to^+ = \bigcup_{n>0} \to^n$
- *reflexive transitive closure* $\to^* = \to^0 \cup \to^+$

Instead of $(a, b) \in \to$, we write $a \to b$ for some $a, b \in A$.

Definition 2.5. A relation is a *strict order* if it is irreflexive, transitive and asymmetric. We say a strict order $>$ on a set A is a *total order* if either $a_1 > a_2$ or $a_2 > a_1$ holds for all $a_1, a_2 \in A$.

Definition 2.6. Let $>$ be a strict order on A and let $\to$ be a relation on A. We say $\to$ is *compatible* with $>$ if $a > b$ holds for all $a \to b$.

Definition 2.7. A relation $\to$ is *well-founded* if there is no infinite sequence such that

$$a_1 \to a_2 \to a_3 \to \cdots$$

holds. Well-founded relations are also called *terminating* relations.

2.2 Regular Expressions and Automata

In the field of formal languages, regular expressions are used to describe possibly infinite sets of strings. Here we give a formal description of these terminologies and establish notations for string rewrite systems.

Let Σ be a set of symbols. The set Σ^* denotes the set of all strings over Σ, including ε which represents the empty string. More formally, the set Σ^* is the free monoid generated by Σ under the binary operation concatenation and the identity is the empty string. We omit the details of monoids.

Definition 2.8. The *concatenation* is a binary operation on strings, defined by $u \cdot v = uv$. It can be extended to subsets of $\Sigma^* \times \Sigma^*$. Given relations $\to_1$ and $\to_2$ on Σ^*, the concatenation of $\to_1 \cdot \to_2$ is defined as the relation:

$$\{(ux, vy) \mid u \to_1 v \text{ and } x \to_2 y\}$$

We call a set of strings as a *language*.

Definition 2.9. The *Kleene closure* S^* of a language S is defined as follows:

$$S^* = \bigcup_{i \in \mathbb{N}} S_i$$

with

$$S_i = \begin{cases} \{\varepsilon\} & \text{if } i = 0 \\ \{u \cdot v \mid u \in S_{i-1} \text{ and } v \in S\} & \text{if } i \geq 1 \end{cases}$$

Definition 2.10. The *length* $|w|$ of a string w is defined as follows:

$$|w| = \begin{cases} 0 & \text{if } w = \varepsilon \\ 1 + |w'| & \text{if } w = aw' \text{ for some } a \in \Sigma \end{cases}$$

We call the string w' is a substring of a string w if $w = uw'v$ for some strings u, v. We denote the set of all substrings of w as $\mathsf{Sub}(w)$.

Definition 2.11. *Regular expressions* are defined by the following BNF:

$$\alpha ::= \varepsilon \mid a \mid \alpha\alpha \mid \alpha^*$$

We define the *language of a regular expression* α as follows:

$$L(\alpha) = \begin{cases} \{\varepsilon\} & \text{if } \alpha = \varepsilon \\ \{a\} & \text{if } \alpha = a \text{ with } a \in \Sigma \\ \{uv \mid u \in L(\alpha_1) \text{ and } v \in L(\alpha_2)\} & \text{if } \alpha = \alpha_1\alpha_2 \\ L(\alpha_1)^* & \text{if } \alpha = \alpha_1^* \end{cases}$$

Definition 2.12. A *non-deterministic finite state automaton* A is a tuple $(Q, \Sigma, \Delta, 0, F)$ where

1. Q is a finite set called the states,
2. Σ is a finite set called the alphabet,
3. $\Delta \subseteq \{pa \to q \mid p, q \in Q, a \in \Sigma \cup \{\varepsilon\}\}$ is the transition rules,
4. $0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

The *language* $L(A)$ accepted by an automaton A is the set:

$$\{u \mid u \in \Sigma^* \text{ and } 0u \to_\Delta^* q \text{ for some } q \in F\}$$

We especially call these automata *deterministic finite state automata* if their transition rules set is not only a subset of $\{pa \to q \mid p, q \in Q, a \in \Sigma\}$ but also a function. We abbreviate each of them as NFA and DFA.

The following two theorems related to the membership problem will be used in the remaining part of this thesis.

Theorem 2.13. *For every regular expression α, there is a DFA A such that $L(A) = L(\alpha)$.*

Theorem 2.14. *The following problem is decidable:*

Instance: A DFA A and a string u.

Question: Does $u \in L(A)$ hold?

2.3 String Rewrite Systems

In this section, we introduce string rewrite systems. Throughout this section we will work on a finite alphabet Σ with symbols and strings $u, v \in \Sigma^*$.

Definition 2.15. A *string rewrite system* (SRS) R is a relation on Σ^* and each element of R is a *rewrite rule*.

Definition 2.16. Let R be an SRS. A *rewrite step* is a pair (u, v) if $u = wxw'$ and $v = wyw'$ for a rewrite rule $x \to y \in R$. We call v is a *reduct* of u and denote the relation of all rewrite steps as $\to_R$.

Definition 2.17. Let R be an SRS. We say u is *in normal form* if there does not exist string v such that $u \to_R v$. The set of all strings which are in normal form is denoted by $\mathsf{NF}(R)$. String v is a *normal form* of u if $u \to_R v$ and $v \in \mathsf{NF}(R)$.

Example 2.18. Consider the SRS $R = \{ab \to ba\}$. It is easy to have the rewrite step $aba \to_R baa$ and the string baa is in normal form since no string can be rewritten from the string baa.

2.4 Termination and Recursive Path Orders

Termination is an important property of string rewrite systems. Unfortunately, termination is undecidable. In this section, we recall the definition of well-founded orders and explain a common technique to check whether the relation $\to_R$ induced by a string rewrite system R is terminating. We simply say *R is terminating* if $\to_R$ is terminating.

The basic idea is to prove termination of string rewrite systems is to find a well-founded order strict on Σ^*. By the property of well-founded orders, it

is easy to get that string rewrite system R is terminating if $u \to_R v$ implies $u > v$ whenever $u, v \in \Sigma^*$. Obviously, Since the pairs u, v is sometimes infinite, we introduce a class of well-founded orders called reduction orders which allow us just check only these rewrite rules instead of all rewrite steps.

Definition 2.19. A *reduction order* is a well-founded strict order $>$ on Σ^* such that for every strings $u, v, x, y \in \Sigma^*$ if $u > v$ implies $xuy > xvy$ holds.

Definition 2.20. The *recursive path order* $>_{rpo}$ is the relation on Σ^* with respect to a strict order $>$ on Σ is inductively defined as follows: $u >_{rpo} v$ if one of the following holds:

(1) $u \neq v = \varepsilon$.

(2) $u = au', v = av'$ for a symbol $a \in \Sigma$ and some strings $u', v' \in \Sigma^*$, and $u' >_{rpo} v'$.

(3) $u = au', v = bv'$ for some symbols $a, b \in \Sigma$ and strings $u', v' \in \Sigma^*$, $au' >_{rpo} v'$ and $a > b$.

(4) $u = au'$ for a symbol $a \in \Sigma$ and a string $u' \in \Sigma^*$, and $u' \geq_{rpo} v$.

Recursive path orders are reduction orders. Another one reduction order called homeomorphic embedding will be used in this thesis.

Definition 2.21. The *homeomorphic embedding* $\trianglerighteq_{emb}$ is the relation on Σ^* that is inductively defined as follows: $u \trianglerighteq_{emb} v$ if one of the following holds:

(1) $u = \varepsilon = v$.

(2) $u = au', v = av'$ for a symbol $a \in \Sigma$ and some strings $u', v' \in \Sigma^*$, and $u' \trianglerighteq_{emb} v'$.

(3) $u = au'$ for a symbol $a \in \Sigma$ and a string, $u' \trianglerighteq_{emb} v$ and $u' \in \Sigma^*$.

Finally by the following theorem, we develop a method to prove termination for some particular systems.

Theorem 2.22. *An SRS R is terminating if there exists a reduction order $>$ such that $u > v$ holds for all $u \to v \in R$.*

In the next two chapters, we explain how to use recursive path orders as reduction orders on these string rewrite systems with infinite rules.

Chapter 3

Regular Expression based String Rewrite Systems

This chapter introduces a class of infinite string rewrite systems based on regular expressions and presents the recursive path order on regular expressions to check the termination property under this formalization.

3.1 String Rewrite Systems via Regular Expressions

Recalling regular expressions and their induced languages, we adapt the notion to string rewriting. In such a way, string rewrite systems with infinite many rules can be represented finitely.

Definition 3.1. A *regular expression based string rewrite system* (RSRS), is a set of pairs consisting of regular expressions. Let R be an RSRS. The induced SRS $L(R)$ is the SRS over Σ with following rules:

$$L(R) = \{u \to v \mid u \in L(\alpha) \text{ and } v \in L(\beta) \text{ for some } \alpha \to \beta \in R\}$$

Example 3.2. Consider the RSRS $\{ab^* \to ba\}$. We compute $L(R)$ as follows:

$$\begin{aligned} L(R) &= \{(u \to v \mid u \in L(ab^*) \text{ and } v \in L(ba)\} \\ &= \{a \to ba, ab \to ba, abb \to ba, \cdots\} \end{aligned}$$

The induced SRS is the system over $\{a, b\}$ with the following rules:

$$\{ab^n \to ba \mid n \in \mathbb{N}\}$$

Note that we use Definition 2.11 as the definition of regular expressions here. The or operation | is not considered here. If $(\alpha|\beta) \to \gamma$ appears in some RSRS, it can be easily replaced by $\{\alpha \to \gamma, \beta \to \gamma\}$. Moreover $\varnothing$ is not considered here since $L(\varepsilon) = \varepsilon$.

Decidability of the normal form existence should be considered in the first place since all we have is a set pairs consisting regular expressions. Rewriting under the induced string rewrite system must be guaranteed.

Theorem 3.3. *The following problem is decidable:*

Instance: An RSRS R and a string u.

Question: Does $u \in \mathsf{NF}(L(R))$ hold?

Proof. To decide if u is already in normal form, we need to check that whether there exists no pair of $\alpha \to \beta \in R$ and a substring $u' \in \mathsf{Sub}(u)$ such that $u' \in L(\alpha)$ holds by the definition of rewrite step. Since the membership problem of regular languages is decidable by Theorem 2.14, it suffices to check all pairs in R and all substrings of u. Because that both of R and $\mathsf{Sub}(u)$ are finite, the problem can be decided. □

We are also interested in the problem of computing a reduct of some string u if u is not in normal form.

Proposition 3.4. *Let R be a RSRS and let $u \notin \mathsf{NF}(L(R))$ be a string. The set consisting all reducts of u with respect to $L(R)$ is computable.*

Proof. Since u is not in normal form, there exists a substring u' such that $u' \in L(\alpha)$ for some rule $\alpha \to \beta \in R$, i.e. $u = w_1 u' w_2$ holds for some strings w_1, w_2. Since the language $L(\beta)$ is computable and $\mathsf{Sub}(u)$ is finite, all the reducts of u is computable. □

Example 3.5. Consider the RSRS over $\{a, b\}$ consisting of:

$$ab^* \to ba$$
$$ab^* \to b$$

Ths string $abba$ whose substring abb is in $L(ab^*)$. Thus, the string $abba$ is not in normal form. We can obtain the set of all reducts from $L(ba)$ and $L(b)$ as follows:

$$\{baa, ba\}$$

3.2 Recursive Path Orders

Most termination techniques are based on reduction orders introduced in Chapter 2.4. When it comes to these systems with infinite rules, checking all rules is impossible in finite time. However, if we succeed in representing them in regular expression based systems, the termination property can be proved by orienting regular expression pairs.

Definition 3.6. Let α be a regular expression. The set $\mathcal{P}(\alpha)$ of symbols in α is defined as follows:

$$\mathcal{P}(\alpha) = \begin{cases} \varnothing & \text{if } \alpha = \varepsilon \\ \{a\} & \text{if } \alpha = a \text{ with } a \in \Sigma \\ \mathcal{P}(\alpha_1) \cup \mathcal{P}(\alpha_2) & \text{if } \alpha = \alpha_1\alpha_2 \\ \mathcal{P}(\alpha_1) & \text{if } \alpha = \alpha_1^* \end{cases}$$

The string $\widehat{\alpha}$ of α is defined as follows:

$$\widehat{\alpha} = \begin{cases} \varepsilon & \text{if } \alpha = \varepsilon \\ a\widehat{\alpha_1} & \text{if } \alpha = a\alpha_1 \text{ with } a \in \Sigma \\ \widehat{\alpha_1} & \text{if } \alpha = \gamma^*\alpha_1 \end{cases}$$

Example 3.7. Consider the regular expression $\alpha = (a^*b)^*a$. We have $\mathcal{P}(\alpha) = \{a, b\}$ and $\widehat{\alpha} = a$. The function $\mathcal{P}(\alpha)$ accumulates all the symbols in α and the function $\widehat{\alpha}$ eliminates all Kleene stars in α. Now we prove the eliminated string is in the language of α.

Lemma 3.8. *Let α be a regular expression. The following propositions hold:*

(1) $\widehat{\alpha} \in L(\alpha)$.

(2) $u \trianglerighteq_{emb} \widehat{\alpha}$ *for all* $u \in L(\alpha)$.

Proof. First, we show $\widehat{\alpha} \in L(\alpha)$ by structural induction on α. We distinguish four cases by the definition of $\widehat{\alpha}$:

(a) If $\alpha = \varepsilon$ then $\widehat{\alpha} = \varepsilon \in \{\varepsilon\} = L(\alpha)$ follows.

(b) If $\alpha = a\alpha_1$ then $\widehat{\alpha_1} \in L(\alpha_1)$ follows from the induction hypothesis. Let $u \in L(\alpha_1)$. We obtain $au \in L(\alpha)$, which yields $\widehat{\alpha} = a\widehat{\alpha_1} \in L(\alpha)$.

(c) If $\alpha = \gamma^*\alpha_1$ then $\widehat{\alpha} = \widehat{\alpha_1}$. Since $\widehat{\alpha_1} \in L(\alpha_1)$ holds by the induction hypothesis. The following proposition holds:

$$\widehat{\alpha_1} \in L(\gamma)^*L(\alpha_1) = L(\gamma^*\alpha_1)$$

It entails $\widehat{\alpha} \in L(\alpha)$.

Therefore, we have $\widehat{\alpha} \in L(\alpha)$. Then we show the second proposition by structural induction on α. Likewise we distinguish three cases:

(a) If $\alpha = \varepsilon$ then u can only be ε. Thus, we obtain $u \trianglerighteq_{emb} \widehat{\alpha} = \varepsilon$ by the definition of $\trianglerighteq_{emb}$.

(b) If $\alpha = a\alpha_1$ then we know $u \in L(\alpha) = \{au_1 \mid u_1 \in L(\alpha_1)\}$. Thus $u = au_1$. From the induction hypothesis, we obtain $u_1 \trianglerighteq_{emb} \widehat{\alpha_1}$, which yields that $u = au_1 \trianglerighteq_{emb} a\widehat{\alpha} = \widehat{a\alpha}$ holds by the definition of $\trianglerighteq_{emb}$. Thus $u \trianglerighteq_{emb} \widehat{\alpha}$.

(c) If $\alpha = \gamma^*\alpha_1$ then let $w \in L(\gamma^*)$ and let $u_1 \in L(\alpha_1)$. Similar arguments lead to $u_1 \trianglerighteq_{emb} \widehat{\alpha_1}$. Since $u = wu_1$, we obtain $wu_1 \trianglerighteq_{emb} u_1$ by the definition of $\trianglerighteq_{emb}$. In addition, $u_1 \trianglerighteq_{emb} \widehat{\alpha_1} = \widehat{\alpha}$. Thus $u \trianglerighteq_{emb} \widehat{\alpha}$.

Therefore $u \trianglerighteq_{emb} \widehat{\alpha}$ for all $u \in L(\alpha)$. □

The above lemma shows that the string $\widehat{\alpha}$ is minimum with respect to $\trianglerighteq_{emb}$ in the language of $L(\alpha)$.

For convenience, we define a relation between symbols and alphabets when none of symbols in the alphabet is greater than the given symbol.

Definition 3.9. Let $a \in \Sigma$ be a symbol, let $B \subseteq \Sigma$ be a set and let $>$ be a strict order on Σ. We denote $a > B$ if $a > b$ holds for all $b \in B$.

Lemma 3.10. *Let $a \in \Sigma$ be a symbol, let Σ_1 be a subset of Σ and let $>$ be a strict order. If $a > \Sigma_1$ holds then $a >_{rpo} u$ holds for all $u \in (\Sigma_1)^*$.*

Proof. Suppose $a > \Sigma_1$ and $b \in \Sigma_1$. From Definition 3.9, we obtain $a > b$. Let u be an arbitrary string in Σ_1^* then we show $a >_{rpo} u$ by induction on $|u|$. If $u = \varepsilon$ then $a >_{rpo} u$ is easy to check. We consider the inductive step by assuming $u = bu'$, and $a >_{rpo} u$ is directly inferred from the conditions $a >_{rpo} u'$ and $a > b$. Hence, The above claim holds. □

The following basic property is easily proved by the definition of $>_{rpo}$.

Lemma 3.11. *Let u and v be strings. The following propositions hold:*

(1) If $u >_{rpo} v$ then $uw >_{rpo} v$ for all string w.

(2) If $u >_{rpo} wv$ then $u >_{rpo} w$ for all string w.

Proof. We first prove (1) by induction on derivation of $u >_{rpo} v$:

(a) If $u \neq v = \varepsilon$ then $uw >_{rpo} \varepsilon = v$ is obvious.

(b) If $u = au'$ and $v = av'$ then $u'w >_{rpo} v'$ which follows from the induction hypothesis. Finally, we obtain $uw = au'w >_{rpo} av' = v$ by the definition of $>_{rpo}$.

(c) If $u = au', v = bv'$ and $a > b$ by a similar argument to (b). We obtain $uw = au'w >_{rpo} bv' = v$.

(d) If $u = au'$ and $u' \geq_{rpo} v$, the induction hypothesis yields $u'w >_{rpo} v$. Thus $uw = au'w >_{rpo} v$ holds by applying the definition of $>_{rpo}$.

Then we show the second proposition by induction on $|u|$. If $u = \varepsilon$ then it is trivial. We consider the inductive case if $u = au'$:

(a) If $w = aw'$ then $u' >_{rpo} w'v$ holds by the definition of $>_{rpo}$, from which we obtain $u' >_{rpo} pw$. Thus $u = au' >_{rpo} aw' = w$ is derived directly from the definition of $>_{rpo}$.

(b) If $w = bw'$ and $a > b$ then we obtain that $u' >_{rpo} wv$ holds. The induction hypothesis yields that $u' >_{rpo} bw'$. Besides that $a > b$, we have $u >_{rpo} w$.

(c) If $u' >_{rpo} w$ then $u' >_{rpo} w$ holds directly from the induction hypothesis. Hence $u = au' >_{rpo} w$ by the definition of $>_{rpo}$.

Therefore $u >_{rpo} wv$ implies $u >_{rpo} w$ for all string w. □

Another important property between $\trianglerighteq_{emb}$ and $>_{rpo}$ is the following lemma.

Lemma 3.12. *Let $u, v \in \Sigma^*$ be strings. If $u \trianglerighteq_{emb} v$ then $u \geq_{rpo} v$.*

Proof. Assume that $u \trianglerighteq_{emb} v$. Considering the three cases in the definition of $\trianglerighteq_{emb}$, we prove $u \geq_{rpo} v$ by induction on $|u|$.

(a) If $u = \varepsilon = v$, $u \geq_{rpo} v$ because $\geq_{rpo}$ is reflexive.

(b) If $u = au', v = av'$ and $u' \trianglerighteq_{emb} v'$ then from the induction hypothesis, we obtain $u' \geq_{rpo} v'$. Thus $u = au' \geq_{rpo} av' = v$ holds by the definition of $>_{rpo}$.

(c) If $u = au'$ and $u' \trianglerighteq_{emb} v$, from the induction hypothesis we obtain $u' \geq_{rpo} v$. Thus $u = au' \geq_{rpo} v$ holds following to the the definition of $>_{rpo}$.

Therefore $u \trianglerighteq_{emb} v$ implies $u \geq_{rpo} v$. □

Now the recursive path order on regular expressions can be defined after introducing above definitions.

Definition 3.13. The *recursive path order* $\succ_{rpo}$ on regular expressions is the binary relation with respect to a strict order $>$ on Σ that is defined as follows: $\alpha \succ_{rpo} \beta$ if one of the following holds:

(1) $\widehat{\alpha} \neq \beta = \varepsilon$.

(2) $\alpha = a\alpha', \beta = a\beta'$ and $\alpha' \succ_{rpo} \beta'$.

(3) $\alpha = a\alpha', \beta = b\beta', a > b$ and $\alpha \succ_{rpo} \beta'$.

(4) $\alpha = a\alpha', \beta = \gamma^*\beta', a > \mathcal{P}(\gamma)$ and $\alpha \succ_{rpo} \beta'$.

(5) $\alpha = \gamma\alpha'$ and $\alpha' \succeq_{rpo} \beta$.

Here a, b are symbols in Σ and α', β', γ are regular expressions.

Example 3.14. Consider the strict order $a > b$. The RSRS $\{ab \to b^*a\}$ is compatible with the order $\succ_{rpo}$ as follows:

$$\dfrac{a > \mathcal{P}(b) \quad \dfrac{\dfrac{}{b \succ_{rpo} \varepsilon}\,(1)}{ab \succ_{rpo} a}\,(2)}{ab \succ_{rpo} b^*a}\,(4)$$

Some given RSRS is compatible with $\succ_{rpo}$ implies that the induced string rewrite system is compatible with $>_{rpo}$ obviously.

Theorem 3.15. *Let α and β be regular expressions. If $\alpha \succ_{rpo} \beta$ holds then $u >_{rpo} v$ for all $u \in L(\alpha)$ and $v \in L(\beta)$.*

Proof. Let α, β be regular expressions and let $u \in L(\alpha), v \in L(\beta)$ be strings. We distinguish five cases and show $u >_{rpo} v$ by induction on the derivation of $\alpha \succ_{rpo} \beta$:

(a) If $\widehat{\alpha} \neq \beta = \varepsilon$ then $\widehat{\alpha} \trianglerighteq_{emb} v$. From Lemma 3.12 we know $u \trianglerighteq_{emb} \widehat{\alpha}$ which yields $\widehat{\alpha} >_{rpo} v$ because $\widehat{\alpha} \neq \beta$.

(b) If $\alpha = a\alpha', \beta = a\beta'$ and $\alpha' \succ_{rpo} \beta'$ then let $u' \in L(\alpha')$ and let $v' \in L(\beta')$. We obtain $u' >_{rpo} v'$ from the induction hypothesis. By the definition of $>_{rpo}$ and $L(\cdot)$, we have $u = au' >_{rpo} av' = v$.

(c) If $\alpha = a\alpha', \beta = b\beta', a > b$ and $\alpha \succ_{rpo} \beta'$ then this case is similar to (b). We conclude that $u' >_{rpo} v'$ holds from the induction hypothesis. Thus, the definition of $>_{rpo}$ yields $u = au' >_{rpo} bu' >_{rpo} bv' = v$. Hence, the proof is complete.

(d) If $\alpha = a\alpha', \beta = \gamma^*\beta', a > \mathcal{P}(\gamma)$ and $\alpha >_{rpo} \beta'$ then we have $au' >_{rpo} v'$ for $au' \in L(\alpha')$ and $v' \in L(\beta')$ from the induction hypothesis. It is easy to show $au' >_{rpo} w$ where $w \in L(\gamma)$ derived from Lemma 3.10 and Lemma 3.11. Besides that $L(\gamma) \subseteq (\mathcal{P}(\gamma))^*$, we obtain $au' >_{rpo} wv'$.

(e) If $\alpha = \gamma\alpha'$ and $\alpha' >_{rpo} \beta$ then we have $u' >_{rpo} v$ for $u' \in L(\alpha')$ from the induction hypothesis. It is clear that $u = wu' >_{rpo} v$ for all $w \in L(\gamma)$ by induction on $|w|$.

□

Since we have found an order on regular expressions which is capable to orient each pair of strings induced by their languages under $>_{rpo}$. It is easy to conclude the following corollary with the help of Theorem 2.22.

Corollary 3.16. *If an RSRS is compatible with $\succ_{rpo}$ then the induced SRS is terminating.*

Note that the item (5) in Definition 3.13 also deals with Kleene star in the left side.

Example 3.17. Let $R = \{b^*a \to b\}$ be an RSRS. It is obvious that $b^*a \succ_{rpo} b$ if $a > b$. Since $R \subseteq \succ_{rpo}$ then $L(R)$ is terminating by Corollary 3.16.

The converse part of Theorem 3.15 which shows the completeness is proved by the following lemmas.

Lemma 3.18. *Let u be a string and let β be a regular expression. If $u >_{rpo} v$ for all $v \in L(\beta)$ then $u \succ_{rpo} \beta$.*

Proof. Let u be a string and let β be a regular expression. Assuming $u >_{rpo} v$ for $v \in L(\beta)$, we show $u \succ_{rpo} \beta$ by induction on $|u| + |\beta|$. The base case is $u = \varepsilon$ and $\beta = \varepsilon$. This claim obviously holds since the antecedent is false. Then we consider the inductive step by the structure of β:

(a) If $\beta = b\beta'$ then we know $u >_{rpo} v = bv'$ from the assumption where $bv' \in L(\beta)$. We distinguish three cases by the definition of $>_{rpo}$:

  (i) If $u = bu'$ and $u' >_{rpo} v'$ then it gives the condition to use the induction hypothesis which leads to $u' \succ_{rpo} \beta'$. Since adding the same prefix does not matter the order $\succ_{rpo}$, we have $u \succ_{rpo} \beta$.

  (ii) If $u = au', u >_{rpo} v'$ and $a > b$ then it yields that $au' \succ_{rpo} \beta'$ from the induction hypothesis. Besides that $a > b$, it is easy to know $u \succ_{rpo} \beta = b\beta$ holds.

(iii) If $u = au'$ and $u' >_{rpo} v$ then similar arguments show $u' \succ_{rpo} \beta$, which makes evident to $u = au' \succ_{rpo} \beta$ by the definition of $\succ_{rpo}$.

(b) If $\beta = \gamma^*\beta'$ and let $v \in \{wv' \mid w \in L(\gamma^*) \text{ and } v' \in L(\beta')\}$. Thus $u >_{rpo} w$ from Lemma 3.10. We consider the structure of u:

(i) If $u = bu'$ and $u' >_{rpo} v$ then from the induction hypothesis we have $u' \succ_{rpo} \beta$, which yields that $u \succ_{rpo} \beta$ by the definition of $\succ_{rpo}$.

(ii) If $u = bu'$ and $u' \not>_{rpo} w$ then we claim $b > \mathcal{P}(w)$ for all w and show it by contradiction. Assuming that there exists $a \in \mathcal{P}(\gamma)$ such that $a \geq b$, let $w_0 \in L(\gamma)$ be the minimum string containing b. Since $w \in L(w_0^*)$ holds, we obtain $|bu'|_b < |w|_a$, which leads to $bu' \not>_{rpo} w$. Consider the following fact:

$$\mathcal{P}(\gamma^*) = \bigcup_{w \in L(\gamma *)} \mathcal{P}(w)$$

We have $b > \mathcal{P}(\gamma^*)$. It remains to show $u \succ_{rpo} \beta'$ to prove $u \succ_{rpo} \gamma^*\beta'$. It is easy to obtain $u \succ_{rpo} \beta'$ from the induction hypothesis just by letting $w = \varepsilon \in L(\gamma^*)$. Thus, we have $u >_{rpo} v'$.

Therefore $u >_{rpo} \beta$ holds. □

Lemma 3.19. *Let α and β be regular expressions. If $\widehat{\alpha} \succ_{rpo} \beta$ then $\alpha \succ_{rpo} \beta$.*

Proof. We show $\alpha \succ_{rpo} \beta$ by structure induction on α and β. If $\beta = \varepsilon$ then by assuming $\widehat{\alpha} \succ_{rpo} \varepsilon$, we obtain $\varepsilon \neq \widehat{\widehat{\alpha}} = \widehat{\alpha}$ from the definition of $\succ_{rpo}$. Thus $\alpha \succ_{rpo} \beta$ holds by applying the definition again. Then we distinguish some cases:

(a) If $\alpha = a\alpha_1$ and $\beta = a\beta_1$ then $\widehat{\alpha} = a\widehat{\alpha_1} \succ_{rpo} a\beta_1$ holds. We obtain $\widehat{\alpha_1} \succ_{rpo} \beta_1$ by the definition of $\succ_{rpo}$, which yields $\alpha_1 \succ_{rpo} \beta$ from the induction hypothesis. Hence $\alpha \succ_{rpo} \beta$ also holds by applying the definition again.

(b) If $\alpha = a\alpha_1$ and $\beta = b\beta_1$ then it is easy to obtain $a > b$ and $\widehat{\alpha} \succ_{rpo} \beta_1$ by the definition, which leads to the conclusion that $a\alpha_1 \succ_{rpo} \beta_1$ from the induction hypothesis. In the same manner, we can see that $\alpha \succ_{rpo} \beta$.

(c) If $\alpha = a\alpha_1$ and $\beta = \gamma^*\beta_1$ then the proof of $a\alpha_1 \succ_{rpo} \beta_1$ runs as before. By combining the condition $a > \mathcal{P}(\gamma)$, the definition of $\succ_{rpo}$ yields that $\alpha \succ_{rpo} \beta$.

(d) If $\alpha = \gamma^* \alpha_1$, then $\widehat{\alpha_1} = \widehat{\alpha} \succ_{rpo} \beta$ holds. We have $\alpha_1 \succ_{rpo} \beta$ from the induction hypothesis. Thus, the same conclusion can be drawn by the definition of $\succ_{rpo}$.

Therefore $\alpha \succ_{rpo} \beta$ holds. □

Theorem 3.20. *Let α and β be regular expressions. If $u >_{rpo} v$ for all $u \in L(\alpha)$ and $v \in L(\beta)$ then $\alpha \succ_{rpo} \beta$ holds.*

Proof. By assuming $u >_{rpo} v$ for all $v \in L(\beta)$, we obtain $u \succ_{rpo} \beta$ from Lemma 3.8. Combining Lemma 3.18 and Lemma 3.19, we have $\alpha \succ_{rpo} \beta$. □

The following corollary is now a direct consequence of Theorem 3.15 which shows the soundness and Theorem 3.20 which shows completeness.

Corollary 3.21. *Let α and β be regular expressions. The proposition $u >_{rpo} v$ holds for all $u \in L(\alpha)$ and $v \in L(\beta)$ if and only of $\alpha \succ_{rpo} \beta$ holds.*

Decidability is obvious by combining Corollary 3.16.

Corollary 3.22. *The RPO-termination of RSRSs is* decidable.

Chapter 4

Ordered Transducer based String Rewrite Systems

In this chapter, we present another restricted transducers called *ordered transducers* which can be used to show the terminating property of the induced string rewrite systems with the help of recursive path orders.

4.1 String Rewrite Systems via Transducers

Transducers are finite state abstract machines which accept pairs of strings. First of all, we show the formal definition of transducers which induce a class of string rewrite systems with infinite rules.

Definition 4.1. A *transducer based string rewrite system* (TSRS) is a finite state transducer $T = (Q, \Sigma, \Delta, 0, F)$, where

1. Q is a finite subset of $\mathbb{N}$ called the *states*,
2. Σ is a finite set called the *alphabet*,
3. $\Delta \subseteq \{pu \to vq \mid p, q \in Q \text{ and } u, v \in \Sigma^*\}$ is the set of *transition rules*,
4. $0 \in Q$ is the *start state*, and
5. $F \subseteq Q$ is the set of *accept states*.

Let $p \in Q$ be a state. The relation derived by T and p is defined as:

$$R(T, p) = \{(u, v) \mid u, v \in \Sigma^* \text{ and } pu \to_\Delta^* qv \text{ for some } q \in F\}$$

If $p = 0$ then $R(T)$ with respect to Σ is the *induced string rewrite system* of T. We denote $R(T, p)$ as $R(p)$ if T is obvious from the context.

Example 4.2. Consider the following infinite string rewrite system R:

$$\{a^n a \to (ba)^n b \mid n \in \mathbb{N}\}$$

s We build the following TSRS T in Figure 4.1 such that $R(T) = R$.

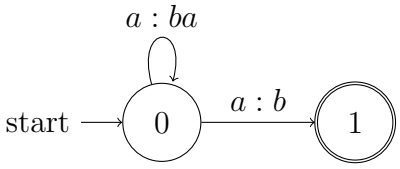


Figure 4.1: The TSRS T with $R(T) = R$

Building the above TSRS means that we have a finite representation of infinite string rewrite systems that seem to make automatic process possible.

Now we define the *left automaton* of some TSRS T to induce the left side of rules in the SRS $R(T)$.

Definition 4.3. Let $T = (Q, \Sigma, \Delta, 0, F)$ be a TSRS. The *left automaton* A_l of T is the automaton $(Q, \Sigma, \Delta_l, 0, F)$ with following transitions:

$$\Delta_l = \{pu \to q \mid pu \to vq \in \Delta\}$$

Example 4.4 (Continued from Example 4.2)**.** The left automaton A_l of T has the following transitions:

$$0a \to 0$$
$$0a \to 1$$

Here states 0 is the initial state and 1 is the accept state.

Using the left automaton, we can check whether a normal form exists in ordinary string rewriting.

Proposition 4.5. *The following problem is decidable:*

Instance: A TSRS T and a string u.

Question: Does $u \in \mathsf{NF}(R(T))$ hold?

Proof. To decide that if u is already in normal form, we need to guarantee that each substring of u does not match any left rule in $R(T)$. More precisely, to ensure that there does not exist $u' \in \mathsf{Sub}(u)$ such that $u' \in L(A_l)$ holds. Since the substring set is finite and the membership problem is decidable by Theorem 2.14, we conclude that the above problem is also decidable. □

Example 4.6 (Continued from Example 4.4)**.** Consider the string aab. Since the string aa is a substring of aab and $aa \in L(A_l)$, we obtain that aab is not in normal form.

We are also interested in the problem of computing a reduct of some string u if u is not in normal form.

Lemma 4.7. *Let $T = (Q, \Sigma, \Delta, 0, F)$ be a TSRS and let A_l be the left automaton of T. If $0u \to^n_{\Delta_l} q$ for some state q and $n \in \mathbb{N}$ then $0u \to^n_{\Delta} vq$ for some string v.*

Proof. A simple fact from Definition 4.3 says that for all $px \to q \in \Delta_l$, the proposition $px \to yq \in \Delta$ must hold for some string y. We show that $0u \to^n_{\Delta_l} q$ implies $0u \to^n_{\Delta} vq$ by induction on n.

- If $n = 0$, we have $u = \varepsilon$ and $q = 0$. Thus $0\varepsilon \to^0_{\Delta} \varepsilon 0$.
- If $n > 0$, we suppose that there exists $p \in Q$ and strings u_1, u_2 such that $u = u_1u_2$ and:

  $$0u_1 \to^{n-1}_{\Delta_l} p \text{ and } pu_2 \to q \in \Delta_l$$

  The induction hypothesis yields $0u_1 \to^{n-1}_{\Delta} v_1p$ for some string v_1. By the above fact, we have $pu_2 \to_{\Delta} v_2q$ for some string v_2. Hence we obtain:

  $$0u_1u_2 \to^{n-1}_{\Delta} v_1pu_2 \to_{\Delta} v_1v_2q$$

  Therefore $0u \to^n_{\Delta} vq$ holds when $v = v_1v_2$.

□

Example 4.8 (Continued from Example 4.6)**.** We know that $aa \in L(A_l)$ which yields $0aa \to^2_{\Delta_l} 1$. From the original TSRS T, we have $0aa \to^2_{\Delta} bab1$.

Proposition 4.9. *Let T be a TSRS and let u be a string not in normal form. The set of all reducts of u with respect to $R(T)$ is computable.*

Proof. Let A_l be the left automaton of T. Obviously, there exist a substring $u' \in \mathsf{Sub}(u)$ and an accept state r such that:

$$u = w_1u'w_2 \text{ and } 0u' \to^n_{\Delta_l} r$$

By Lemma 4.27, we know that $0u' \to^n_{\Delta} v'r$ for some string v'. Since the language $L(A_l)$ is computable and $\mathsf{Sub}(u)$ is finite, all reducts of u is computable. □

Rewriting under the new representation proposed above succeeds. Now we consider the problem how to check the termination property of the induced string rewrite system in the next section.

4.2 Recursive Path Orders

Recursive path orders seem to be not so compatible with transducer based string rewrite systems to check the termination property of the induced string rewrite systems. The main reason is that transducers lack the term structure, which is useful when extending orders. Thus, we develop a technique to employ recursive path orders iteratively along with state transitions.

In the following, the concept *configurations* will be introduced to develop a decidable technique to check the RPO-termination of a restricted class of TSRS.

Definition 4.10. Let $T = (Q, \Sigma, \Delta, 0, F)$ be a TSRS.

- The set $S(p)$ consisting successor states of $p \in Q$ is defined as:

$$\{q \mid q \in Q \text{ and } up \to qv \in \Delta \text{ for some } u, v \in \Sigma^*\}$$

- A *configuration* of T is a triple (u, v, p), where $u, v \in \Sigma^*$ and $p \in Q$. The set $S(u, v, p)$ consisting successor configurations of (u, v, p) is the following set:

$$\{(ux, vy, q) \mid x, y \in \Sigma^* \text{ and } qx \to yp \in \Delta \text{ for some } p \in Q\}$$

  The set of all configurations is denoted by $\mathcal{C}$.

- An *extended configuration* is a quadruple (o, u, v, p), where $o \in \{\mathtt{s}, \mathtt{w}\}$ and $(u, v, p) \in \mathcal{C}$. We extend $S(u, v, p)$ to $S(o, u, v, p)$ as follows:

$$S(o, u, v, p) = \{(o, u', v', q) \mid (u', v', q) \in S(u, v, p)\}$$

Definition 4.11. Let $c = (o, u, v, p)$ be an extended configuration and let $n \in \mathbb{N}$ be a number. We denote $>^{\mathtt{w}}_{rpo}$ as $>^{=}_{rpo}$ and $>^{\mathtt{s}}_{rpo}$ as $>_{rpo}$. The property $P_n(c)$ means: For all $x \to y \in R(p)$,

$$\text{If } |uxvy| = n \text{ then } ux >^{o}_{rpo} vy$$

Let C, I be extended configuration sets. We just write $P_n(C)$ if $P_n(c)$ holds for all $c \in C$. More specifically, we denote $I \Vdash_n C$ if $P_k(I)$ implies $P_n(C)$ for all $k < n$.

We define a procedure to check whether the induced SRS of T is compatible with $>_{rpo}$ only by the assistance of T.

Definition 4.12. Let $T = (Q, \Sigma, \Delta, 0, F)$ be a TSRS and let $>$ be a strict total order on Σ. We define the following transformation rules:

[r1]

$$\frac{\{(o, au, av, q)\} \uplus C, I}{\{(o, u, v, q)\} \cup C, I}$$

[r2]

$$\frac{\{(o, au, bv, q)\} \uplus C, I}{\{(\mathtt{s}, au, v, q)\} \cup C, I} \text{ if } a > b$$

[r3]

$$\frac{\{(o, au, bv, q)\} \uplus C, I}{\{(\mathtt{w}, u, bv, q)\} \cup C, I} \text{ if } a < b$$

[c1]

$$\frac{\{(o, \varepsilon, v, q)\} \uplus C, I}{C \cup S(o, \varepsilon, v, q), I} \text{ if } q \notin F \cup S(q)$$

[c2]

$$\frac{\{(o, u, \varepsilon, q)\} \uplus C, I}{C \cup S(o, u, \varepsilon, q), I} \text{ if } q \notin F \cup S(q)$$

[c3]

$$\frac{\{(o, u, \varepsilon, q)\} \uplus C, I}{C \cup S(o, u, \varepsilon, q), I} \text{ if } q \in F - S(q) \text{ and } u >^{o}_{rpo} \varepsilon$$

[c4]

$$\frac{\{(o, u, \varepsilon, q)\} \uplus C, I}{C \cup S(o, u, \varepsilon, q), I \cup \{(o, u, \varepsilon, q)\}} \text{ if } q \in S(q) - F \text{ and } (o, u, \varepsilon, q)^{\times}$$

[c5]

$$\frac{\{(o, \varepsilon, v, q)\} \uplus C, I}{C \cup S(o, \varepsilon, v, q), I \cup \{(o, \varepsilon, v, q)\}} \text{ if } q \in S(q) - F \text{ and } (o, \varepsilon, v, q)^{\times}$$

[c6]

$$\frac{\{(o, u, \varepsilon, q)\} \uplus C, I}{C \cup S(o, u, \varepsilon, q), I \cup \{(o, u, \varepsilon, q)\}} \text{ if } q \in F \cap S(q), u >^{o}_{rpo} \varepsilon \text{ and } (o, u, \varepsilon, q)^{\times}$$

[d]

$$\frac{\{(o, u, v, q)\} \uplus C, I}{C, I} \text{ if } (o, u, v, q)^{\circ}$$

Here C, I are sets of extended configurations. We denote these rules as $\vdash_T$ or $\vdash$ if T is obvious from the context. We call an extended configuration (o, u, v, q) is *marked* in I if $(o, u, v, q) \in I$. For convenience, $(o, u, v, q)^\circ$ means that it is marked and $(o, u, v, q)^\times$ means that it is not marked.

We want to have that each rule in the induced SRS of some TSRS is compatible with $>_{rpo}$ after applying Definition 4.12 iteratively.

Example 4.13. Consider the following TSRS T with:

$$R(T) = \{a^n a \to (ba)^n b \mid n \in \mathbb{N}\}$$

We can prove that it is compatible with $>_{rpo}$ by induction on n. Figure 4.2 is the corresponding TSRS.

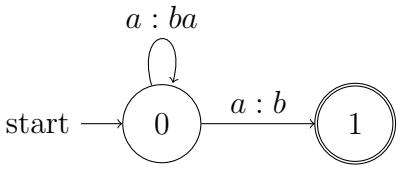


Figure 4.2: TSRS T with $R(T) = R$

Let $c = (o, u, v, q)$ be the extended configuration with $o = \mathtt{s}, u = \varepsilon, v = \varepsilon$ and $q = 0$. We start the transformation from $(\{c\}, \varnothing)$ as follows:

$$\begin{aligned}
\{c\}, \varnothing &\vdash_{c1} \{(\mathtt{s}, a, ba, 0), (\mathtt{s}, a, b, 1)\}, \varnothing \\
&\vdash_{r2} \{(\mathtt{s}, a, a, 0), (\mathtt{s}, a, b, 1)\}, \varnothing \\
&\vdash_{r1} \{(\mathtt{s}, \varepsilon, \varepsilon, 0,), (\mathtt{s}, a, b, 1)\}, \varnothing \\
&\vdash_{c4} \{(\mathtt{s}, a, ba, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash_{r2} \{(\mathtt{s}, a, a, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash_{r1} \{(\mathtt{s}, \varepsilon, \varepsilon, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash_{d} \{(\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash_{r2} \{(\mathtt{s}, a, \varepsilon, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash_{c3} \varnothing, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\}
\end{aligned}$$

No more transformation rules can be applied and it ends in $(\varnothing, I)$ for some extended configuration set I.

In the definition of $\vdash$, the extended configuration which is currently dealing with must be checked whether it is marked or not. Our technique fails when transformation rules are applied without these side conditions.

Example 4.14. The induced system indicated in Example 4.13 cannot be proved terminating since $\vdash$ does not terminate as follows:

$$\begin{aligned}
\{c\}, \varnothing &\vdash_{c1} \{(\mathtt{s}, a, ba, 0), (\mathtt{s}, a, b, 1)\}, \varnothing \\
&\vdash_{r2} \{(\mathtt{s}, a, a, 0), (\mathtt{s}, a, b, 1)\}, \varnothing \\
&\vdash_{r1} \{(\mathtt{s}, \varepsilon, \varepsilon, 0,), (\mathtt{s}, a, b, 1)\}, \varnothing \\
&\vdash_{c4} \{(\mathtt{s}, a, ba, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash_{r2} \{(\mathtt{s}, a, a, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash_{r1} \{(\mathtt{s}, \varepsilon, \varepsilon, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash_{c1} \{(\mathtt{s}, a, ba, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash_{r2} \{(\mathtt{s}, a, a, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash_{r1} \{(\mathtt{s}, \varepsilon, \varepsilon, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\} \\
&\vdash \cdots
\end{aligned}$$

We conclude Theorem 4.22 by restricting the class of TSRSs to ensure soundness.

Definition 4.15. Let $T = \{Q, \Sigma, \Delta, 0, F\}$ be a TSRS and let $>$ be a strict total order. We group these transition rules in Δ by some state $r \in Q$ as follows:

- The *state relation* of T, denoted as Δ^{ε} is the relation:

$$\{p \to q \mid pu \to vq \in \Delta\}$$

- The *loop rule set* induced by r is the set:

$$\Delta^{\leftrightarrow}(r) = \{pu \to vq \in \Delta \mid p = r = q\}$$

- The *left rule set* induced by r is the set:

$$\Delta^{\leftarrow}(r) = \{pu \to vq \in \Delta \mid q = r \text{ and } p \neq r\}$$

- The *right rule set* induced by r is the set:

$$\Delta^{\rightarrow}(r) = \{pu \to vq \in \Delta \mid p = r \text{ and } q \neq r\}$$

We define some properties on r as follows:

- r is called *removable* if $r \to^*_{\Delta_\varepsilon} p$ for no $p \in F$.

- r is called *movable* if both left and rule rule set of r are not empty and $\Delta^{\leftrightarrow}(r) = \varnothing$.
- r is called *eliminative* if there is a rule $pu \to vp \in \Delta^{\leftrightarrow}(r)$ with $p = r$ such that $u >^{=}_{rpo} v$ or $v >_h u \neq \varepsilon$.

Definition 4.16. A TSRS $T = (Q, \Sigma, \Delta, 0, F)$ is called *ordered* if there are no eliminative, removable or movable states and every $px \to yq \in \Delta$ satisfies $p \leq q$ and:

- If $p < q$ then $x >^{=}_{rpo} y$ and
- if $p = q$ then $y >_{rpo} x$, $a = b$ and $|x|_a \geq |y|_b$.

Here $a = max(x)$ and $b = max(y)$.

Let T be an unordered TSRS. The ordered version of T can be derived from the following derivation rules.

Definition 4.17. Let $T = (Q, \Sigma, \Delta, 0, F)$ be a TSRS. We define some derivation rules on Δ as follows:

1. $$\frac{\Delta \uplus \{pu \to qv\}}{\Delta} \quad \text{if } q \text{ is removable}$$
2. $$\frac{\Delta \uplus \{pu \to pv\}}{\Delta} \quad \text{if } p \text{ is eliminative}$$
3. $$\frac{\Delta \uplus \{pu \to qv\}}{\bot} \quad \text{if } v >_{rpo} u \text{ and } q \in \Delta^{\leftrightarrow}$$
4. $$\frac{\Delta \uplus \{pu \to pv\}}{\bot} \quad \text{if } v >_h u \text{ or } |u|_a < |v|_b$$ where $a = max(u)$ and $b = max(v)$
5. $$\frac{\Delta \uplus \Delta^{\leftarrow}(q) \uplus \Delta^{\rightarrow}(q)}{\Delta \cup \{pux \to rvy \mid pu \to vq \in \Delta^{\leftarrow}(q) \text{ and } qx \to yr \in \Delta^{\rightarrow}(q)\}} \quad \text{if } q \text{ is movable}$$

We denote these derivation rules as $\Rightarrow$.

Applying the above derivation rules to some TSRS should not affect our technique.

Fact 4.18. Let $T = (Q, \Sigma, \Delta, 0, F)$ be a TSRS and let $\Delta \neq \bot$ be a transition set. If $\Delta \Rightarrow \Delta'$ then propositions $R(T) \subseteq >_{rpo}$ and $R(T') \subseteq >_{rpo}$ are equivalent where $T' = (Q, \Sigma, \Delta', 0, F)$.

The example of $\Rightarrow$ is showed in Example 4.31.

Definition 4.19. Let $T = (Q, \Sigma, \Delta, 0, F)$ and $T^\circ = (Q, \Sigma, \Delta^\circ, 0, F)$ be TSRSs. We say that T° is *the ordered TSRS* of T if $\Delta \Rightarrow^* \Delta^\circ$ holds and $\Delta^\circ \in \mathsf{NF}(\Rightarrow) - \{\bot\}$.

Now these transformation rules in Definition 4.12 can be used when we have the ordered TSRS T° since $R(T) \subseteq >_{rpo}$ if and only if $R(T^\circ) \subseteq >_{rpo}$.

Before showing the RPO-termination, we need some lemmas.

Lemma 4.20. *Let T° be an ordered TSRS. If $C, I \vdash_{T^\circ} C', I'$ except for the rule **d** then $P_n(C')$ implies $P_m(C)$ for all $n \in \mathbb{N}$ where $m \geq n$.*

Proof. We prove it holds for each transformation rule except **d**. Let $n \in \mathbb{N}$.

[r1] By assuming the antecedent, we find $m \in \mathbb{N}$ such that the following holds:

$$P_n((o, u, v, q)) \implies P_m((o, au, av, q))$$

Let $m = n+2$. We take an arbitrary pairs $x \to y \in R(q)$. It is sufficient to show (1) implies (2):

$$|uxvy| = n \implies ux >^\circ_{rpo} vy \tag{1}$$
$$|auxavy| = n + 2 \implies aux >^\circ_{rpo} avy \tag{2}$$

Suppose (1) and $|auxavy| = n + 2$. We conclude $|uxvy| = n$ directly. Thus $ux >^\circ_{rpo} vy$ holds. By the definition of $>^\circ_{rpo}$, we obtain $aux >^\circ_{rpo} avy$.

[r2] Similar arguments are shared with **[r1]**. Thus, we just need to show that if $a > b$ then for some m:

$$P_n((\mathtt{s}, au, v, q)) \implies P_m((o, au, bv, q))$$

Let $m = n+1$. We take an arbitrary pair $x \to y \in R(q)$. It is sufficient to show (3) implies (4):

$$|auxvy| = n \implies aux >_{rpo} vy \tag{3}$$
$$|auxbvy| = n + 1 \implies aux >^\circ_{rpo} bvy \tag{4}$$

Suppose (3) and $|auxbvy| = n + 1$. We conclude $|auxvy| = n$. Thus $aux >_{rpo} vy$ holds. Besides that $a > b$, it yields $aux >_{rpo} bvy$ by Definition 2.20. Hence $aux >^{\circ}_{rpo} bvy$.

[r3] Similar to the above proof, we conclude that what we have to show is: if $a < b$ then there exists m such that:

$$P_n((\mathtt{w}, u, bv, q)) \implies P_m((o, au, bv, q))$$

Let $m = n+1$. We take an arbitrary pair $x \to y \in R(q)$. It is sufficient to show that (5) implies (6):

$$|uxbvy| = n \implies aux >^{=}_{rpo} vy \tag{5}$$

$$|auxbvy| = n + 1 \implies aux >^{\circ}_{rpo} bvy \tag{6}$$

Suppose (5) and $|auxbvy| = n + 1$. We conclude $|uxbvy| = n$. Thus $ux >^{=}_{rpo} bvy$. Besides that $a < b$, it yields $aux >_{rpo} bvy$ by Definition 2.20. Hence $aux >^{\circ}_{rpo} bvy$ holds.

[cn] Let $c = (o, u, v, q)$ be the configuration that is altered. We show that there exists m satisfing:

$$P_n(c) \implies P_m(S(c))$$

Let $m = n$. To show the above claim we simply prove:

$$\{(u, v)\} \cdot R(p) \subseteq >^{o}_{rpo} \implies \bigcup_{(o,x,y,q)\in S(c)} \{(ux, vy)\} \cdot R(q) \subseteq >^{o}_{rpo}$$

Let R_1 be $\{(u, v)\} \cdot R(p)$ and let R_2 be the arbitrary union in the right part. Instead of implication elimination, it is finally concluded that showing $R_2 \subseteq R_1$ is sufficient to prove the original claim. By calculating:

$$\begin{aligned} R_1 &= \{(u, v)\} \cdot R(p) \\ &= \{(u, v)\} \cdot \{u' \to v' \mid u', v' \in \Sigma^* \text{ and } pu' \to^*_\Delta v'r \text{ for some } r \in F\} \end{aligned}$$

According to the definition of $S(c)$, the following holds:

$$\begin{aligned} R_2 &= \bigcup_{(o,x,y,q)\in S(c)} \{(ux, vy)\} \cdot R(q) \\ &= \{(u, v)\} \cdot \bigcup_{(x,y,q)\in S(\varepsilon,\varepsilon,p)} \{(x, y)\} \cdot R(q) \end{aligned}$$

After the calculation, we show $R_2 \subseteq R_1$ as follows:

Suppose $(uu', vv') \in R_2$. Next, we know that there exists a tuple $(x, y, q) \in S(p)$ such that $u' = xu'_1, v' = yv'_1$ and $u'_1 \to v'_1 \in R(q)$, which means:

$$qu'_1 \to^*_\Delta v'_1 r$$

By the definition of $S(\cdot)$, we have $px \to yq \in \Delta$. Combining the above conditions, it is concluded that there is a state $r \in F$ such that $pxu'_1 \to^*_\Delta yqu'_1 r$. Thus $(uu', vv') \in R_1$ holds.

Hence $R_1 \subseteq >^o_{rpo}$ implies $R_2 \subseteq >^o_{rpo}$ since $R_2 \subseteq R_1$.

□

Lemma 4.21. *Let T be an ordered TSRS and let $C_0, I_0 \vdash C_1, I_1 \vdash \cdots$ be a sequence. If $C_j, I_j \vdash C_{j+1}, I_{j+1}$ then ${I_{j+1}}' \Vdash_n {C_{j+1}}'$ implies $I_j \Vdash_n C_j$ for all $n \in \mathbb{N}$.*

Proof. By Lemma 4.20, the consequent is obvious for these derivation rules except **d**since no assumption in $P_n(I)$ is used. Now we prove it holds only when **d** is applied. Let $n \in \mathbb{N}$. Suppose $C_j, I_j \vdash_d C_{j+1}, I_{j+1}, I_j + 1 \Vdash_n C_j + 1$ and $P(I_j)$. Let $c = \{(o, u, v, q)\} = C_j - C_{J+1}$. Thus, we only have to show $P(o, u, v, q)$ since $I_j = I_{j+1}$. Since sequence starts from with C_0, I_0 and I_j is not empty, there must exist $i < j$ in the sequence:

$$C_0, \varnothing \vdash^* C_{i-1}, I_{i-1} \vdash_{cn} C_i, I_i \vdash^* C_j, I_j \vdash_d C_{j+1}, I_{j+1} \vdash^* \cdots$$

The side condition of **d** reveals us that:

If $I_i - I_{i-1} = \{(o, u, v, q)\}$ then let $(x, y) \in R(u, v, q)$. We show $x >^o_{rpo} y$ by induction on $|xy|$. Thus $x' >^o_{rpo} y'$ follows from the induction hypothesis $P_k(I)$ and $k < n$. It yields $x = w_1 x'$ and $y = w_2 y'$ for some w_1, w_2 such that $qw_1 \to w_2 q \in \Delta, w_1 \neq \varepsilon$ and $w_2 \neq \varepsilon$. Hence $|xy| > |x'y'|$. Only r1,r2,r3 are applied in $C_i, I_i \vdash^* C_j, I_j$. Subsequently $P(C_j)$ implies $P(C_i)$ by Lemma 4.20. Thus $x >^o_{rpo} y$.

Therefore the above lemma holds. □

Theorem 4.22. *Let $T^\circ = (Q, \Sigma, \Delta, 0, F)$ be an ordered TSRS. The following holds:*

$$(C, \varnothing) \vdash^* (\varnothing, I) \implies R(T^\circ) \subseteq >_{rpo}$$

Here $C = \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\}$.

Proof. Suppose that $\{(\mathtt{s}, \varepsilon, \varepsilon, 0)\}, \varnothing) \vdash^* (\varnothing, I)$ holds. From Lemma 4.21, it yileds that for all $n \in \mathbb{N}$:

$$I \Vdash_n \varnothing \implies \varnothing \Vdash_n (s, \varepsilon, \varepsilon, 0)$$

Easy evidence shows that $P_n(I)$ implies $P_n(\varnothing)$ since $P_n(\varnothing)$ is true, $P_n(\varnothing)$ implies $P_n(\mathtt{s}, \varepsilon, \varepsilon, 0)$. Hence, it yields that $R(T^\circ) \subseteq >_{rpo}$ by the strict version $\mathtt{s}$. □

Since we find a way to deal with TSRSs, which is capable to orienting their induced string rewrite system by recursive path orders. It is easy to conclude the following corollary according to Theorem 2.22.

Corollary 4.23. *Let $T = (Q, \Sigma, \Delta, 0, F)$ be an ordered TSRS. The induced SRS is terminating if $\{(\mathtt{s}, \varepsilon, \varepsilon, 0)\}, \varnothing \vdash^* \varnothing, I$ holds for some I.*

Decidability and completeness should also be considered by using the following lemmas.

Lemma 4.24. *Let T° be an ordered TSRS and let $C_0, I_0 \vdash_{T^\circ} C_1, I_1 \vdash_{T^\circ} \cdots$ be a sequence. Given $o \in \{\mathtt{s}, \mathtt{w}\}$, a string u and a state p, the following set is finite:*

$$\{v \mid (o, u, v, p) \in \bigcup I_i\}$$

Proof. If u is not the empty string then all satisfied v can only be ε according to the structure of I. We consider the case when $u = \varepsilon$. Suppose that there exists an infinite sequence:

$$\cdots \vdash C_j \uplus \{(o', \varepsilon, y', p\}, I_j \vdash_{c6} C_k \cup \{(o', x, y'y, p)\}, I_k \vdash \cdots$$

We claim that there exists $q \leq p$ and $i < j$ such that the following holds:

$$C_{i-1} \uplus \{(o, w, \varepsilon, q)\}, I_{i-1} \vdash C_i \uplus \{(o, wx, y, p)\}, I_i \vdash^* C_j \uplus \{(o', \varepsilon, y', p)\}, I_j$$

The proof is simple since if $p = 0$ then $w = \varepsilon$; If $p \neq 0$ then we take the q such that $qx_0 \to y_0 p \in \Sigma$ for some x_0, y_0. By Definition 4.17, we have $x_0 >^{=}_{rpo} y_0$. Thus we can take a string w.

Now we prove $max(y') < max(y)$. It is obvious that $max(w) < max(y)$. Let $a = max(x)$. We have $max(x) = max(y)$ and $|x|_a = |y|_b$ by Definition 4.17. Thus $max(y') < max(y)$.

Finally, we obtain the direct consequence $I_k = I_k \cup \{(o', \varepsilon, y, p)\}$ from the following sequence:

$$C_k \cup \{(o', x, y'y, p)\}, I_k \vdash^*_{rn} C_k \cup \{(o', \varepsilon, y, p)\}, I_k \cup \{(o', \varepsilon, y, p)\} \vdash \cdots$$

Accordingly the set $\{v \mid (o, u, v, p) \in \bigcup I_i\}$ is finite. □

Lemma 4.25. *Let T° be an ordered TSRS. The relation $\vdash_{T^\circ}$ terminates.*

Proof. Suppose that there exists an infinite sequence $C_1, I_1 \vdash C_2, I_2 \vdash \cdots$. Let $\vdash_t$ be the composition of rules **r1**,**r2**,**r3**,**c1**,**c2**,**c3**,and **d**. Then $\vdash$ is terminating obviously. Thus the infinite sequence must be

$$C_1, I_1 \vdash_t^* \circ \vdash_{cn} C_2, I_2 \vdash_t^* \circ \vdash_{cn} \cdots$$

By the fact that $I = I'$ holds if $C, I \vdash_t C', I'$, we have $|I_{i+1} - I_i| = 1$ for all $i \in \mathbb{N}$ in the above sequence. Then we take the infinite sequence:

$$c_1, c_2, c_3, \cdots$$

Here $c_i = (o, u, v, q)$ is the first element of $I_{i+1} - I_i$. Since the set $\{\mathtt{s}, \mathtt{w}\}$ where o belongs and Q where q belongs are finite. In addition, the set where v belongs is finite by Lemma 4.24 and $\trianglerighteq_{emb}$ is a well partial order. Thus, there exists an extend order which is also a well partial order on c_i, which follows from Dickson's lemma.

There exists $i, j \in \mathbb{N}$ such that $u_j \trianglerighteq_{emb} u_i$ where $(o_i, u_i, v_i, q_i) = c_i$ and $(o_j, u_j, v_j, q_j) = c_j$. Let $\vdash = \vdash_t^* \circ \vdash_{cn}$. Thus, the following infinite sequence exists:

$$\cdots \vdash C_{i+1}, I_i \cup \{(o_i, u_i, v_i, q_i)\} \vdash \cdots \vdash C_{j+1}, I_j \cup \{(o_j, u_j, v_j, q_j)\} \vdash \cdots$$

Here $\vdash$ is applied. On the other hand, none of rule **c4**, **c5**, **c6** can be applied by the side condition since $u_j \trianglerighteq_{emb} u_i$. Now we have a contradiction. Therefore, the relation $\vdash_{T^\circ}$ is terminating. □

Note that the given TSRS must be ordered. By Definition 4.17, we present a TSRS that is not ordered, which does not follow the above theorem.

Example 4.26. We consider the following TSRS indicated in Figure 4.3.

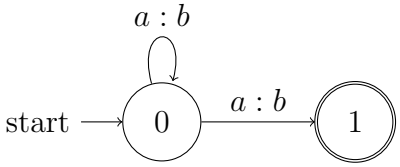


Figure 4.3: A TSRS which it not ordered

The above TSRS induces the system $\{a^n a \to b^n b\}$ which is compatible with $>_{rpo}$ obviously. However, our technique will fail since the above relation

$\vdash$ is not terminating:

$$\begin{aligned}
\{c\}, \varnothing &\vdash_{c1} \{(\mathtt{s}, a, b, 0), (\mathtt{s}, a, b, 1)\}, \varnothing \\
&\vdash_{r2} \{(\mathtt{s}, a, \varepsilon, 0), (\mathtt{s}, a, b, 1)\}, \varnothing \\
&\vdash_{c4} \{(\mathtt{s}, aa, b, 0,), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, a, \varepsilon, 0)\} \\
&\vdash_{r2} \{(\mathtt{s}, aa, \varepsilon, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, a, \varepsilon, 0)\} \\
&\vdash_{c4} \{(\mathtt{s}, aaa, b, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, a, \varepsilon, 0), (\mathtt{s}, aa, \varepsilon, 0)\} \\
&\vdash_{r2} \{(\mathtt{s}, aaa, \varepsilon, 0), (\mathtt{s}, a, b, 1)\}, \{(\mathtt{s}, a, \varepsilon, 0), (\mathtt{s}, aa, \varepsilon, 0)\} \\
&\vdash \cdots
\end{aligned}$$

Here $c = (\mathtt{s}, \varepsilon, \varepsilon, 0)$.

Lemma 4.27. *Let T° be an ordered TSRS and let $c = (\mathtt{s}, \varepsilon, \varepsilon, 0)$ be an extended configuration. The normal from of $(c, \varnothing)$ with respect to $\vdash_{T^\circ}$ is (C, I) for some I, where*

$$C \subseteq \{(\mathtt{s}, \varepsilon, v, q) \mid q \in F\} \cup \{(\mathtt{w}, \varepsilon, v, q) \mid q \in F \text{ and } v \neq \varepsilon\}$$

Proof. Suppose (C, I) is the normal form of $(c, \varnothing)$. By Theorem 4.22, it is already in normal form if $C = \varnothing$. If C is not empty then we analyze the elements in C. Let $(o, u, v, q) \in C$.

- If $u \neq \varepsilon$ and $v \neq \varepsilon$ then **r1**,**r2** or **r3** is applied.
- If $u \neq \varepsilon$ and $v = \varepsilon$ then **d**, **c1**,**c3** or **c5** can be applied.
- If $u = \varepsilon, v \neq \varepsilon$ and $q \notin F$ then **d**,**c2** or **c6** is applied.
- If $u = \varepsilon, v = \varepsilon$ and $q \notin F$ then **c2**,**c3**,**c5**,**c6** or **d** is applied.
- If $u = \varepsilon, v = \varepsilon, q \in F$ and $o = \mathtt{w}$ then one rule in **c1** or **c4** is applied.

The remaining cases are:

- Case 1: $u = \varepsilon, v \neq \varepsilon, q \in F$ and $o = \mathtt{w}$,
- Case 2: $u = \varepsilon, v \neq \varepsilon, q \in F$ and $o = \mathtt{s}$, and
- Case 3: $u = \varepsilon, v = \varepsilon, q \in F$ and $o = \mathtt{s}$.

Hence, Lemma 4.27 holds. □

Then we can show the proof of completeness with the help of the termination property of $\vdash$.

Lemma 4.28. *If $(C, I) \vdash (C', I')$ then $P_n(C)$ implies $P_n(C')$ for all $n \in \mathbb{N}$.*

Proof. Suppose $(C, I) \vdash (C', I')$. It follows directly when **r1**,**r2**,**r3** or **d** is applied. Similar arguments are shared with Lemma 4.20 when **c1**,**c2**,**c3**,**c4**,**c5** or **c6** is applied. □

Theorem 4.29. *Let $T^\circ = (Q, \Sigma, \Delta, 0, F)$ be an ordered TSRS. The following holds:*

$$R(T^\circ) \subseteq >_{rpo} \Longrightarrow\ (C, \varnothing) \vdash^* (\varnothing, I)$$

Here $C = \{(\mathtt{s}, \varepsilon, \varepsilon, 0)\}$.

Proof. Suppose $R(T^\circ) \subseteq >_{rpo}$. We prove it by contradiction. Assume that the normal form of $(C, \varnothing)$ is not $(\varnothing, I)$ for some I. By Lemma 4.27, we have $C \neq \varnothing$ and

$$C \subseteq \{(\mathtt{s}, \varepsilon, v, q) \mid q \in F\} \cup \{(\mathtt{w}, \varepsilon, v, q) \mid q \in F \text{ and } v \neq \varepsilon\} = C_\infty$$

On the other hand, we have $P(C)$ by Lemma 4.28 which leads to a contradiction since $P(C_\infty)$ does not hold. □

The following corollary is now a direct consequence of Theorem 4.22 that shows soundness and of Theorem 4.29 that shows completeness.

Corollary 4.30. *Let $T^\circ = (Q, \Sigma, \Delta, 0, F)$ be the ordered TSRS of T. The following two propositions are equivalent.*

- *For all $u \to v \in R(T)$, $u >_{rpo} v$ holds.*
- *For some extend configuration set I, $(\{(\mathtt{s}, \varepsilon, \varepsilon, 0)\}, \varnothing) \vdash^*_{T^\circ} (\varnothing, I)$ holds.*

The RPO-termination of ordered TSRSs is decidable.

Now the termination property of R_{120° presented in Chapter 1 is easy to prove using our approach.

Example 4.31. The following system is the induced SRS of T_{120° indicated in Figure 4.4.

$$R_{120} = \left\{ \begin{array}{rcl} aaa & \to & \varepsilon \\ bab^{n+1}ab & \to & ab^n a \\ ab^m ab^n ab & \to & bab^m ab^n a \end{array} \;\middle|\; m, n \in \mathbb{N} \right\}$$

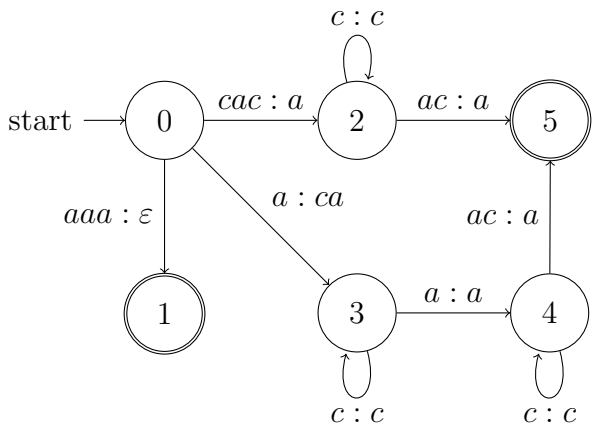


Figure 4.4: TSRS T_{120°

By Definition 4.17, we compute the ordered TSRS of T_{120° indicated in Figure 4.5.

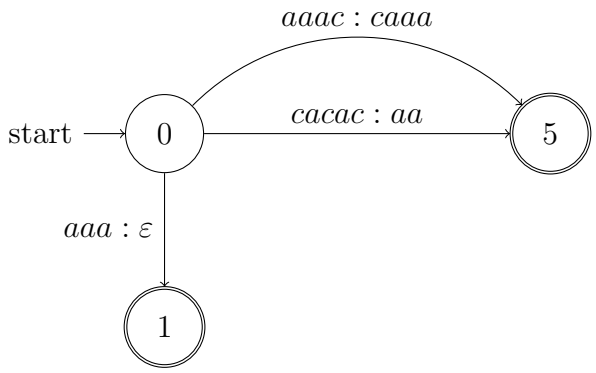


Figure 4.5: Ordered TSRS $T^\circ_{120^\circ}$

Let $c = (\mathtt{s}, \varepsilon, \varepsilon, 0)$. We prove that the string rewrite system R_{120° is terminating as follows:

$$\begin{aligned}\{c\}, \varnothing &\vdash_{T^\circ_{120^\circ}} \{(\mathtt{s}, aaac, caaa, 5), (\mathtt{s}, cacac, aa, 5), (\mathtt{s}, aaa, \varepsilon, 5)\}, \varnothing \\ &\vdash^*_{T^\circ_{120^\circ}} \{(\mathtt{s}, c, \varepsilon, 5), (\mathtt{w}, c, \varepsilon, 5), (\mathtt{s}, aaa, \varepsilon, 5)\}, \varnothing \\ &\vdash^*_{T^\circ_{120^\circ}} \varnothing, \varnothing\end{aligned}$$

Chapter 5

Conclusion

We presented two methods to indicate string rewrite systems by finite state transducers, which are *regular expression based string rewrite systems* and *ordered transducer based string rewrite systems*. Our motivation is to check termination of string rewrite systems with infinite rules like the congruent system for turtle graphics equivalence when $\theta = 120°$. We conclude the thesis by introducing related work.

5.1 Related Work

Recurrence Terms The research [3] brings us a finite schematization of infinite terms by *recurrence terms*. Termination and confluence of the new finitely presented system are directly related to the infinite ones it denoted. However, The problem how to check these properties remains open. These infinite systems represented in our approach can also be represented in recurrence terms. But our technique provides a decidable way to check termination.

Instead of showing the detailed definition of recurrence terms, we illustrate an infinite system R:

$$\{a^n b \to b \mid n \in \mathbb{N}\}$$

It can be represented by the following TRS R' based on the f-rooted recurrence terms:

$$f(a, n) : b : x \to b : x$$

If the above system is given, This system is not capable of rewriting strings like aab. In order to fill in the gap between $f(a, n) : x$ and $a : a : x$, one has to employ the following equations E' to convert between recurrence terms

and terms.

$$f(a,0) : x = x$$
$$f(a,s(n)) : x = a : f(a,n) : x$$

Rewriting is now performed as so-called *rewriting modulo* E', which is defined as follows:

Definition 5.1. Let R be a TRS and let E be a set of equations. The rewrite relation $\to_{R/E}$ is the following relation:

$$\leftrightarrow_E^* \cdot \to_R \cdot \leftrightarrow_E^*$$

Automatically matching the string $a : a : b : x$ to the term $f(a,n) : x$ containing the new function symbol f under E' raises the decidable *pattern marching* problem under *equational systems for recurrence terms*, which are the equational part of our rewrite modulo.

Definition 5.2. Let Σ be an alphabet. The *equational system for recurrence terms* over Σ is the following set:

$$\left\{ \begin{array}{rcl} f(a,0) : x & = & x \\ f(a,s(n)) : x & = & a : f(a,n) : x \end{array} \;\middle|\; a \in \Sigma \right\}$$

Here : is right associative.

Decidability of the matching problem is ensured in [3].

Theorem 5.3. *The following problem is decidable:*

> *instance: An equational system E for recurrence terms, a term l containing f and a term t containing no f*
>
> *question: Do p and σ exist such that $t|_p = l\sigma$ and $l\sigma \leftrightarrow_E^* t|_p$?*

And the position p and the substitution σ are computable.

Note that for an arbitrary equational system E, the above problem is undecidable.

Example 5.4. Consider the term $t = a : a : b : x$. We have $t \to_{R'/E'} b : x$ since $t|_p$ matches $f(a,n) : x$ when $p = \epsilon$ and $\sigma = \{n \mapsto s(s(0)), x \mapsto b : x\}$. Hence

$$a : a : b : x \;\leftrightarrow_{E'}^*\; f(a,s(s(0))) : b : x \;\to_{R'}\; b : x \;\leftrightarrow_{E'}^*\; b : x$$

By the above theorem, we can decide whether a given string can be converted to f-rooted recurrence term by checking all its subterms. If it is then a next reduct after the conversion is computable. Just like Example 5.4, the string $a : a : b : x$ matches $f(a, n) : x$ at the root position. After that, ordinary rewriting of R', which is also computable, is performed.

We expect that RPO-termination of $\to_{R'/E'}$ implies RPO-termination of $\to_R$ in the above example. However, from a simple proof we have that there does not exist a simplification order $>$ such that for every terms s, t, u and v

$$s \leftrightarrow^*_{E'} t > u \leftrightarrow^*_{E'} v \text{ implies } s > v$$

It means that we cannot use simplification orders to check termination of these systems involving equational systems for recurrence terms. Moreover, no reduction order exists such that the above claim holds either.

Besides that, RPO-termination of $\to_{R'}$ does not imply RPO-termination of $\to_R$ either. The following is a simple example

Example 5.5. Consider the above system $R' = \{f(a, n) : b : x \to b : x\}$. If $a > b$ then

$$R' \subseteq >_{rpo}$$

holds by the definition of $\trianglerighteq_{emb}$. Thus R' is terminating.

On the other hand, the original system $R = \{a^n b \to b \mid n \in \mathbb{N}\}$ is not terminating since the following infinite sequence exists

$$b \to_R b \to_R b \to_R \cdots$$

Transducers and Rational Relations. Product construction of transducers was attempted to use during the first stage of this research while another work [13] shows that the equivalence problem for transducers is undecidable. On the other hand, the languages which transducers accept are rational relations [2]. Moreover we also succeeded in illustrating recursive path orders in transducers.

5.2 Limitations

The major weakness of our work is that the class of transducers is restricted. We cannot fully take advantages of transducers to express more string rewrite systems. Besides that, other termination techniques or simplification orders should also be considered to enlarge the class. Furthermore, as a technique to check the termination property, the transducer must be given. Automated generation from infinite systems to transducers should also be considered.

As future work, we plan to introduce lexicographical orders over transducer based systems since their definition is simpler than recursive path orders. Another direction is to enlarge the class of by alerting the states of transducers in a better manner. Also, we consider the implementation of our technique.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Prof. Nao Hirokawa for his guidance and valuable advice. The door of his office was always open whenever I ran into a trouble. I am also grateful to Prof. Mizuhito Ogawa, Prof. Hajime Ishihara and Prof. Kazuhiro Ogata for their helpful comments on my research. Many thanks should also go to all members in Hirokawa, Ogawa Ishihara and Ogata laboratories for their insightful suggestions and assistance.

Bibliography

[1] Leo Bachmair, Nachum Dershowitz, and David A. Plaisted. "Completion without Failure". In: *Rewriting Techniques*. Ed. by Hassan Aït-Kaci and Maurice Nivat. 1989, pp. 1–30.

[2] Jean Berstel and Jacques Sakarovitch. "Recent results in the theory of rational sets". In: *Mathematical Foundations of Computer Science 1986*. Ed. by Jozef Gruska, Branislav Rovan, and Juraj Wiedermann. Berlin, Heidelberg, 1986, pp. 15–28.

[3] Hong Chen, Jieh Hsiang, and Hwa-Chung Kong. "On Finite Representations of Infinite Sequences of Terms". In: *Conditional and Typed Rewriting Systems*. Ed. by S. Kaplan and M. Okada. Berlin, Heidelberg, 1991, pp. 99–114.

[4] Nachum Dershowitz. "Orderings for Term-Rewriting Systems". In: *Theoretical Computer Science* 17.3 (1982), pp. 279–301.

[5] Luís Descalço and Nikola Ruskuc. "Subsemigroups of the Bicyclic Monoid". In: *IJAC* 15 (Feb. 2005), pp. 37–57.

[6] M. Jantzen. "A Note on A Special One-rule Semi-Thue system". In: *Information Processing Letters* 21.3 (1985), pp. 135–140.

[7] Matthias Jantzen. "On A Special Monoid with A Single Defining Relation". In: *Theoretical Computer Science* 16.1 (1981), pp. 61–73.

[8] Deepak Kapur and Paliath Narendran. "A Finite Thue System with Decidable Word Problem and without Equivalent Finite Canonical System". In: *Theoretical Computer Science* 35 (1985), pp. 337–344.

[9] Donald E. Knuth and Peter B. Bendix. "Simple Word Problems in Universal Algebras". In: *Computational Problems in Abstract Algebra*. 1970, pp. 263–297.

[10] Ursula Martin and Tobias Nipkow. "Ordered Rewriting and Confluence". In: *Proceedings of the Tenth International Conference on Automated Deduction*. 1990, pp. 366–380.

[11] Enno Ohlebusch. *Advanced Topics in Term Rewriting*. 1st. 2010.

[12] Craig C. Squier. "Word Problems and A Homological Finiteness Condition for Monoids". In: *Journal of Pure and Applied Algebra* 49.1 (1987), pp. 201–217.

[13] Paavo Turakainen. "On some transducer equivalence problems for families of languages". In: *International Journal of Computer Mathematics* 23.2 (1988), pp. 99–124.