On problems dual to unification: the string-rewriting case

Zumrut Akcam Kibis
University at Albany, State University of New York, zumrutakcam@gmail.com

The University at Albany community has made this article openly available. Please share how this access benefits you.

Follow this and additional works at: https://scholarsarchive.library.albany.edu/legacy-etd

Part of the Computer Sciences Commons

Recommended Citation
https://scholarsarchive.library.albany.edu/legacy-etd/1993

This Dissertation is brought to you for free and open access by the The Graduate School at Scholars Archive. It has been accepted for inclusion in Legacy Theses & Dissertations (2009 - 2024) by an authorized administrator of Scholars Archive. Please see Terms of Use. For more information, please contact scholarsarchive@albany.edu.
ON PROBLEMS DUAL TO UNIFICATION
THE STRING-REWrittINg CASE

by

Zümrüt Akçağ-Kıbis

A Dissertation
Submitted to the University at Albany, State University of New York
in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy

College of Engineering and Applied Sciences
Department of Computer Science
May 2018
ON PROBLEMS DUAL TO UNIFICATION

THE STRING-REWRITING CASE

by

Zümrüt Akçam-Kılıç

© Copyright 2018
To my beloved family
ABSTRACT

Unification, with or without background theories such as associativity and commutativity, is an area of great theoretical and practical interest. Recently unification has become an important tool in areas such as program synthesis and cryptographic protocol analysis.

In this dissertation, we investigate problems which are dual to the unification problem, namely the Fixed Point (FP) problem, Common Term (CT) problem and the Common Equation (CE) problem. We show that the fixed point problem is reducible to the common term problem. We study these three problems for subclasses of convergent string rewriting systems. Our results include: (i) the fixed point problem is undecidable for finite convergent string rewriting systems, (ii) the common term problem is undecidable for the class of dwindling string rewriting systems, and (iii) for the class of finite, monadic and convergent systems, the common equation problem is decidable in polynomial time.
ACKNOWLEDGMENT

In my doctoral studies at SUNY Albany, I have the privilege of working with great professors, researchers and friends. I am grateful to get to know some gracious souls and I would like to take the opportunity to express my gratitude to those who made my Ph.D. experience memorable.

I would like to thank my advisor Dr. Paliath Narendran for his support, guidance and encouragement. Dr. Narendran is more than a Ph.D. advisor, he is a lifelong mentor. He is always willing to answer your questions and shares his insights. I feel honored to work with him, this dissertation would not be possible without him.

I would also like to thank my committee members Dr. Neil Murray and Dr. Daniel J. Dougherty for their valuable guidance and time. Dr. Dougherty’s paper with Dr. Otto and Dr. Narendran is the basis for one of my proofs. Dr. Murray was the department chair for most of my Ph.D. years and he was very supportive of me. He found me an assistantship position in the middle of the academic year and listened to me whenever I had a problem. I am so thankful to him.

I also had the chance to meet one of the leading researchers in our field, Dr. Franz Baader, who motivated me with his vast knowledge. Dr. Franz Baader shared some of his insights with me for the future work of this project which I appreciated a lot.

I would like to thank Dr. Jennifer Goodall for the community she created for the women in the departments of Computer Science, Informatics and Information Science. She is a great mentor and she helped us discover a new world by sending us to amazing conferences such as Grace Hopper Conference. I also want to thank Dr. George Berg for being so supportive of us along with Dr. Goodall.

I would also like to thank Dr. S.S. Ravi for my algorithms skills and his patience for me to build them up. He was also the chair of our coffee club which were precious times to remember. I would like to thank Dr. Seth Chaiken, Dr. Harry B. Hunt III, Dr. Mei-Hwa Chen for teaching me the core subjects and for the valuable discussions.

I am grateful to my academic older brother Serdar Erbatur who listened to my complaints in my first two years of Ph.D. He continues to be a mentor and he always gives me advice on research and life.
My special thanks to my dear friend Veena Ravishankar for being an important part of my Ph.D. experience. We shared a lot of stories, work experience and lifelong memories. I also want to express my gratitude to my dear friends Kim Cornell, Amanda Fernandez, Daniel Hono, Andrew Matusiewicz, Paul Olsen and Peter Hibbs for their wonderful friendship. It is impossible to forget our lunch times and work nights in the old department. Also my roommates and friends Füsun Şahin, Meryem Top and Elanur Selçuk will bring a smile to my face whenever I remember my Ph.D years. I am very grateful to my friends for being a part of my Ph.D story.

Finally I would like to thank my husband Eyyüb Yunus for his support. He is always there for me when I am overwhelmed, frustrated and happy. He always calmed me down and helped me focus on my work. I am also thankful to my brother, who inspired me to start my Ph.D. He supported me starting from the application process till my graduation and he will continue to be my guide in academia. Most importantly to my parents, especially to my mom, and to my aunts who supported me at every step of my education process, thank you so much. I hope I will continue you to make you proud.
# CONTENTS

ABSTRACT .................................................................................. iv

ACKNOWLEDGMENT ................................................................. v

LIST OF FIGURES ........................................................................ viii

1. Introduction ........................................................................... 1
   1.0.1 Outline & Overview of the Results ............................. 5

2. Preliminaries and Notations .................................................. 7
   2.1 Definitions ....................................................................... 7
      2.1.1 Term Rewriting Systems ........................................... 7
      2.1.2 String Rewriting Systems ......................................... 8
      2.1.3 Automata ................................................................. 9

3. String Rewriting Case .......................................................... 11
   3.1 Fixed Point Problem ....................................................... 11
   3.2 Common Term Problem ................................................ 14
      3.2.1 CT for Dwindling Systems ....................................... 14
   3.3 Common Equation Problem .......................................... 20
      3.3.1 Two-mapping CE Problem for Monadic Systems ....... 22
      3.3.2 One-mapping CE Problem for Monadic Systems ....... 24

4. Conclusion ............................................................................. 26

APPENDICES .............................................................................. 28

A. Appendix A .......................................................................... 28
B. Appendix B .......................................................................... 29
C. Appendix C .......................................................................... 35
LIST OF FIGURES

2.1 Some of the classes of String Rewriting Systems 1 .......................... 9

B.1 DFA $M$ concatenation with a single letter $a$. ................................. 29

B.2 DFA $M'$ can have less than $|F|$ states. ........................................... 30

B.3 DFA $M$ concatenation with a string $Z = a_1a_2...a_n$. ......................... 31

B.4 Different Structures of Green and Orange States. ................................. 34

C.1 How the monadic cone looks for $\alpha$ .................................................. 39
CHAPTER 1

Introduction

Unification, with or without background theories such as associativity and commutativity, is an area of great theoretical and practical interest. Unification deals with solving symbolic equations. For instance, given two terms $s = f(a, x)$ and $t = f(y, b)$ where $f$ is a binary function symbol, $a$ and $b$ are constants, and $x$ and $y$ are variables, can we find a substitution such that these two terms will be syntactically equal? The solution to the question above should be ‘yes’ with the following substitution: $\sigma = \{ x \mapsto b, y \mapsto a \}$ (See Section 2 for the definition of substitution).

Equational unification or semantic unification has background theories which differs it from the syntactic unification. The equality of the terms is decided modulo equational theory. For example, $f(f(x, y), z)$ and $f(c, f(b, a))$ are equivalent modulo commutativity with the substitution $\sigma = \{ x \mapsto a, y \mapsto b, z \mapsto c \}$. However in this work, we aim to do the opposite or dual of this process. Given the substitution(s) and an equational theory, we would like to find if there exists a term whose two instances are equivalent, or an equation that both the substitutions satisfy. Here we investigate these problems that can clearly be viewed as dual to the unification problem. Our main motivation for this work is theoretical, but, as explained below, we begin with a practical application that is shared by many fields.

In every major research field, there are variables or other parameters that change over time. These variables are modified — increased or decreased — as a result of a change in the environment. Computing invariants, or expressions whose values do not change under a transformation, is very important in many areas such as Physics, e.g., invariance under the Lorentz transformation.

In Computer Science, the issue of obtaining invariants arises in axiomatic semantics or Floyd-Hoare semantics, in the context of formally proving a loop to be correct. A loop invariant is a condition, over the program variables, that holds before and after each iteration. Our research is partly motivated by the related question of finding expressions, called fixed points, whose values will be the same before and after each iteration, i.e., will remain unchanged as long as the iteration goes on. For instance, for a loop whose body is

$$X = X + 2; \, \, Y = Y - 1;$$
the value of the expression \(X + 2Y\) is a fixed point.

We can formulate this problem in terms of properties of substitutions *modulo* a term rewriting system. One straightforward formulation is as follows: (Please note that the definitions for substitution, \(\text{Dom}\), \(\text{Sig}\) and \(\text{Ran}\) are given in Section 2.)

**Fixed Point Problem (FP)**

**Input:** A substitution \(\theta\) and an equational theory \(E\).

**Question:** Does there exist a non-ground term \(t \in T(\text{Sig}(E), \text{Dom}(\theta))\) such that 
\[
\theta(t) \approx_E t
\]

Example 1: Suppose \(E\) is a theory of integers which contains linear arithmetic. Let \(\theta = \{x \mapsto x - 2, y \mapsto y + 1\}\) and we would like to find a term \(t\) such that \(\theta(t) \approx_E t\). Note that \(x + 2y\) is such a term, since
\[
\theta(x + 2y) = (x - 2) + 2 \cdot (y + 1) \approx_E x + 2y
\]

Example 2: What is the fixed point/invariant of the given loop?

**Algorithm 1 Fixed Point Loop**

1. \(\{x = X_0, y = Y_0\}\)
2. \(\textbf{while} \ x > 0 \ \textbf{do}\)
3. \(x = x - 1\) \hspace{1cm} \(\triangleright \ \theta = \{x \mapsto x - 1, y \mapsto y + x - 1\}\)
4. \(y = y + x\)
5. \(\textbf{end while}\)

Note that the value of the expression \(y + \frac{x \cdot (x - 1)}{2}\) is unchanged, since

- **Before Iteration:** \(y + \frac{x \cdot (x - 1)}{2}\)
- **After Iteration:**

\[
y + x - 1 + \frac{(x - 1) \cdot (x - 2)}{2} = y + \frac{x^2 - 3x + 2 + 2(x - 1)}{2} = y + \frac{x \cdot (x - 1)}{2}
\]

Thus \(y + \frac{x \cdot (x - 1)}{2}\) is a fixed point of \(\theta\).
Note that fixed points may not be unique. Consider the term rewriting system

\[ \{ a(b(y)) \rightarrow a(y), \; c(b(z)) \rightarrow c(z) \} \]

and let \( \theta = \{ x \mapsto b(x) \} \). We can see that both \( a(x) \) and \( c(x) \) are fixed points of \( \theta \).

We plan to explore two related formulations, both of which can be viewed as dual to the well-known unification problem. There are two ways to “dualize” the unification problem:

**Common Term Problem (CT):**

**Input:** Two ground\(^2\) substitutions \( \theta_1 \) and \( \theta_2 \), and an equational theory \( E \).

(i.e., \( \forall \mathcal{Ran}(\theta_1) = \emptyset \) and \( \forall \mathcal{Ran}(\theta_2) = \emptyset \))

**Question:** Does there exist a non-ground term \( t \in T(\Sigma(E), \text{Dom}(\theta_1) \cup \text{Dom}(\theta_2)) \) such that \( \theta_1(t) \approx_E \theta_2(t) \)?

**Example 2:** Consider the two substitutions \( \theta_1 = \{ x \mapsto p(a), \; y \mapsto p(b) \} \) and \( \theta_2 = \{ x \mapsto a, \; y \mapsto b \} \).

If we take the term rewriting system \( R_{lin}^1 \) in the Appendix A as our background equational theory \( E \), then there exists a common term \( t = x - y \) that satisfies \( \theta_1(t) \approx_E \theta_2(t) \).

\[ \theta_1(x - y) \approx_E p(a) - p(b) \approx_E a - b \]

and

\[ \theta_2(x - y) \approx_E a - b \]

We can easily show that the fixed point problem can be reduced to the CT problem.

**Lemma 1.1.** The fixed point problem is reducible to the common term problem.

**Proof.** Let \( \theta_2 \) be the identity substitution. Assume that the fixed point problem has a solution, i.e., there exists a term \( t \) such that \( \theta(t) \approx_E t \). Then the CT problem for \( \theta \) and \( \theta_2 \) has a solution since \( \theta_2(t) \approx_E t \) (because \( \theta_2(s) = s \) for all \( s \)). The “only if” part is trivial, again because \( \theta_2(s) = s \) for all \( s \).

\(^2\)This may not be strictly necessary.
Alternatively, suppose that \( \mathcal{D}om(\theta) \) consists of \( n \) variables, where \( n \geq 1 \). If we map all the variables in \( \mathcal{V} \mathcal{R}an(\theta) \) to new constants, this will create a ground substitution \( \theta_1 = \{x_1 \mapsto a_1, x_2 \mapsto a_2, \ldots, x_n \mapsto a_n\} \). \( \theta_1 \) will be the one of the substitutions for the CT problem. The other substitution, \( \theta_2 \), is the composition of the substitutions \( \theta \) and \( \theta_1 \). The substitution \( \theta_1 \) will replace all of the variables in \( \mathcal{V} \mathcal{R}an(\theta) \) with the new constants, thus making \( \theta_2 \) a ground substitution. Now if \( \theta(t) \approx_E t \), then \( \theta_2(t) = \theta_1(\theta(t)) \approx_E \theta_1(t) \); in other words, \( t \) is a solution to the common term problem.

The “only if” part can also be explained in terms of the composition above. Suppose that \( \theta_1(s) \) and \( \theta_2(s) \) are equivalent, i.e., \( \theta_1(s) \approx_E \theta_2(s) \) for some \( s \). Since \( \theta_2 = \theta_1 \circ \theta \), the equation can be rewritten as \( \theta_1(\theta(s)) \approx_E \theta_1(s) \). Since \( a_1, \ldots, a_n \) are new constants and are not included in the signature of the theory, for all \( t_1 \) and \( t_2 \), \( \theta_1(t_1) \approx_E \theta_1(t_2) \) holds if and only if \( t_1 \approx_E t_2 \) (See [1], Section 4.1, page 60) Thus \( \theta_1(\theta(s)) \approx_E \theta_1(s) \) implies that \( \theta(s) \approx_E s \), making \( s \) a fixed point.

**Common Equation Problem (CE):**

**Input:** Two substitutions \( \theta_1 \) and \( \theta_2 \) with the same domain, and an equational theory \( E \).

**Question:** Does there exist a non-ground, non-trivial \( t_1 \not\approx_E t_2 \) equation \( t_1 \approx_E^? t_2 \), where \( t_1, t_2 \in T(\text{Sig}(E), \mathcal{D}om(\theta_1)) \) such that both \( \theta_1 \) and \( \theta_2 \) are \( E \)-unifiers of \( t_1 \approx_E^? t_2 \)?

By trivial equations, we mean equations which are identities in the equational theory \( E \), i.e., an equation \( s \approx_E^? t \) is trivial if and only if \( s \approx_E t \). We exclude this type of trivial equations in the formulation of this question.

**Example 3:** Let \( E = \{p(s(x)) \approx x, s(p(x)) \approx x\} \). Given two substitutions \( \theta_1 = \{x_1 \mapsto s(s(a)), x_2 \mapsto s(a)\} \) and \( \theta_2 = \{x_1 \mapsto s(a), x_2 \mapsto a\} \), we can see that \( \theta_1(t_1) \approx_E \theta_1(t_2) \) and \( \theta_2(t_1) \approx_E \theta_2(t_2) \), with the equation

\[ p(x_1) \approx_E x_2 \]

However, there is no term \( t \) on which the substitutions agree, i.e., there aren’t any solutions for the common term problem in this example. Thus, CT and CE problems are not equivalent as we observe in the example above.
1.0.1 Outline & Overview of the Results

In this dissertation we will discuss (and survey) these three problems for convergent string rewriting systems. The current literature and our own discoveries are summarized in the Table 1.1, the rectangle boxes indicating our own results:

<table>
<thead>
<tr>
<th></th>
<th>Convergent</th>
<th>Length-reducing</th>
<th>Dwindling</th>
<th>Monadic</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
<td><strong>undecidable</strong></td>
<td>NP-complete</td>
<td>NP</td>
<td>P</td>
</tr>
<tr>
<td>CT</td>
<td>undecidable</td>
<td>undecidable</td>
<td><strong>undecidable</strong></td>
<td>P</td>
</tr>
<tr>
<td>CE</td>
<td>undecidable</td>
<td>undecidable</td>
<td><strong>undecidable</strong></td>
<td>P</td>
</tr>
</tbody>
</table>

Table 1.1: Complexity results of the problems in String Rewriting Systems.

We briefly summarize our results below. For basic definitions of concepts in term rewriting systems and string rewriting systems, please see Chapter 2.

(i) **Fixed Point Problem is undecidable for convergent string rewriting systems.** We prove this result with a reduction from the emptiness problem for non-looping deterministic linear bounded automata (DLBA). In particular, we only consider DLBAs that reconstruct its input in the cases where the input is accepted. This additional requirement is realized by a reduction from the Post Correspondence Problem (PCP) by using the construction given in [2]. The details of this proof can be found in Section 3.1.

(ii) **Common Term Problem is undecidable for dwindling and convergent string rewriting systems.** The undecidability proof of this problem uses a reduction from the Generalized Post Correspondence Problem (GPCP) which is a variant of PCP with fixed start and end dominoes. Section 3.2 explains in more depth how this reduction works.

(iii) **Common Equation Problem is undecidable for dwindling and convergent string rewriting systems.** The CT problem is a particular case of the CE problem. Assume that we have two domain variables in the substitutions. Suppose further that in one substitution we have different mappings for these two variables while in the other substitution we map them to
the empty string. This then becomes an instance of the common term problem. For more information on this proof, please check out Section 3.3.

(iv) **Common Equation Problem is polynomial-time solvable for monadic and convergent string rewriting systems.** There are two variations of the Common Equation problem, namely the case where each substitution consists only of one mapping and the case where there are more mappings. We refer to them as the one-mapping and two-mapping cases. In both cases, we reduce the problem to problems on finite automata which can be solved in polynomial time. The details are given in Section 3.3.1 for the two-mapping problem and Section 3.3.2 for the one-mapping problem. The key lemmas are given in Appendix C.
CHAPTER 2
Preliminaries and Notations

2.1 Definitions

We start by presenting some notation and definitions on term rewriting systems and particularly string rewriting systems. Only some definitions are given in here, but for more details, refer to the books [1] for term rewriting systems and [3] for string rewriting systems.

2.1.1 Term Rewriting Systems

A signature $\Sigma$ consists of finitely many ranked function symbols. Let $X$ be a (possibly infinite) set of variables. The set of all terms over $\Sigma$ and $X$ is denoted as $T(\Sigma,X)$. $\text{Var}(t)$ shows the set of variables for term $t$ and a term is a ground term iff $\text{Var}(t) = \emptyset$. The set of ground terms, or terms with no variables is denoted $T(\Sigma)$. The constant symbols are denoted with $\Sigma(0)$.

Let $\sigma$ be a substitution such that $\sigma : V \to T(\Sigma,V)$, where $V$ defines the countably infinite set of variables. The Domain of a substitution identifies the set of variables that does not map to themselves such that $\text{Dom}(\sigma) := \{ x \in V \mid \sigma(x) \neq x \}$. The range of a substitution can be defined as $\text{Ran}(\sigma) := \{ \sigma(x) \mid x \in \text{Dom}(\sigma) \}$ and variable set in $\text{Ran}(\sigma)$ can be shown as

$$\forall \text{Ran}(\sigma) := \bigcup_{x \in \text{Dom}(\sigma)} \text{Var}(\sigma(x)).$$

Let $E$ be an equational theory and $S$ is an E-unification problem over $\Sigma$. $\text{Sig}(S)$ denotes the set of all function symbols in $S$.

A term rewriting system (TRS) is a set of rewrite rules that are defined on the signature $\Sigma$, in the form of $l \to r$, where $l$ and $r$ are called the left- and right-hand-side (lhs and rhs) of the rule, respectively. The rewrite relation induced by a term rewriting system $R$ is denoted by $\rightarrow_R$. The reflexive and transitive closure of $\to_R$ is denoted $\to_R^*$. A TRS $R$ is called terminating iff there is no infinite chain of terms. A TRS $R$ is confluent iff, for all terms $t$, $s_1$, $s_2$, if $s_1$ and $s_2$ can be derived from $t$, i.e., $s_1 \leftarrow_R^* t \to_R^* s_2$, then there exists a term $t'$ such that $s_1 \to_R^* t' \leftarrow_R^* s_2$. A TRS $R$
is convergent iff it is both terminating and confluent.

A term is irreducible iff no rule of TRS $R$ can be applied to that term. The set of terms that are irreducible modulo $R$ is defined by $\text{IRR}(R)$ and also called as terms in their normal forms. A term $t'$ is said to be an $R$-normal form of a term $t$, iff it is irreducible and reachable from $t$ in a finite number of steps; this is written as $t \rightarrow^1_R t'$.

2.1.2 String Rewriting Systems

String rewriting systems are a restricted class of term rewriting systems where all functions are unary. These unary operators, that are defined by the symbols of a string, applied in the order in which these symbols appear in the string, i.e., if $g, h \in \Sigma$, the string $gh$ will be seen as the term $h(g(x))$. The set of all strings over the alphabet $\Sigma$ is denoted by $\Sigma^*$ and the empty string is denoted by the symbol $\lambda$. Thus the term rewriting system \{ $p(s(x)) \rightarrow x$, $s(p(x)) \rightarrow x$ \} is equivalent to the string-rewriting system

\{ $sp \rightarrow \lambda$, $ps \rightarrow \lambda$ \}

If $R$ is a string rewriting system (SRS) over alphabet $\Sigma$, then the single-step reduction on $\Sigma^*$ can be written as:

For any $u, v \in \Sigma^*$, $u \rightarrow_R v$ iff there exists a rule $l \rightarrow r \in R$ such that $u = xly$ and $v = xry$ for some $x, y \in \Sigma^*$; i.e.,

$\rightarrow_R = \{(xly, xry) \mid (l \rightarrow r) \in R, x, y \in \Sigma^*\}$

For any string rewrite system $R$ over $\Sigma$, the set of all irreducible strings, $\text{IRR}(R)$, is a regular language: in fact, $\text{IRR}(R) = \Sigma^* \setminus \{\Sigma^*l_1\Sigma^* \cup \ldots \cup \Sigma^*l_n\Sigma^*\}$, where $l_1, \ldots, l_n$ are the left-hand sides of the rules in $T$.

Throughout the rest of the paper, $a, b, c, \ldots, h$ will denote elements of the alphabet $\Sigma$, and $l, r, u, v, w, x, y, z$ will denote strings over $\Sigma$. Concepts such as normal form, terminating, confluent, and convergent have the same definitions in the string rewriting systems as they have for the term rewriting systems. An SRS $T$ is called canonical if and only if it is convergent and inter-reduced, i.e., no lhs is a substring of another lhs.

\footnote{It may be more common to view $gh$ as $g(h(x))$ with function application done in the reverse order.}
A string rewrite system $T$ is said to be:

- **monadic** iff the rhs of each rule in $T$ is either a single symbol or the empty string, e.g., $abc \rightarrow b$.

- **dwindling** iff, for every rule $l \rightarrow r$ in $T$, the rhs $r$ is a **proper prefix** of its lhs $l$, e.g., $abc \rightarrow ab$.

- **length-reducing** iff $|l| > |r|$ for all rules $l \rightarrow r$ in $T$, e.g., $abc \rightarrow ba$.

![Figure 2.1: Some of the classes of String Rewriting Systems](image)

2.1.3 Automata

We will briefly mention about deterministic finite automaton (DFA), deterministic linear bounded automaton (DLBA) and Multiple-Entry Finite Automata (MEFA).

**Deterministic Finite Automaton (DFA)**

A finite automaton or finite state acceptor $\mathcal{M}$ can be defined with a quintuple such that $\mathcal{M} = \langle \Sigma, Q, \gamma, q_0, F \rangle$, where

- $\Sigma$ identifies the finite nonempty set of input symbols,

- $Q$ is a finite nonempty set of states,

- $\gamma: Q \times \Sigma \rightarrow Q$ is the transition function,

---

2. The trivial forms of monadic rules such as $a \rightarrow b$ can be ignored. We can get rid of such rules by changing every occurrence of $a$ to $b$. 

$q_0 \in Q$ is the initial state,

$F \subseteq Q$ shows the set of accepting states.

**Multiple-Entry Finite Automata (MEFA)**

For more information about Multiple-Entry Finite Automata, please check Gill and Kou’s paper [4]. We will provide some basic notation for MEFAs.

A multiple-entry finite automaton is a quadruple $\mathcal{M} = < \Sigma, \hat{Q}, \hat{\gamma}, \hat{F} >$. $\hat{Q}$ indicates the set of initial states. In MEFA, any state can serve as an initial state. In this dissertation, we view the MEFAs as a set of DFAs combined as one.

**Deterministic Linear Bounded Automaton (DLBA)**

As name suggests, linear bounded automaton is a restricted type of Turing machine, such that the head is not allowed to move beyond where input starts/ends. It is a single tape Turing machine with left and right endmarkers, and it does not print any other symbol when it reaches to these endmarkers. It can be defined with a quintuple $\mathcal{M} = < \Sigma, Q, \gamma, q_0, q_a >$ and it is called deterministic LBA if $|\gamma(q, a)| \leq 1$ for all $q \in Q - \{q_a\}$ and $a \in \Sigma$. 
CHAPTER 3
String Rewriting Case

3.1 Fixed Point Problem

Note that for string rewriting systems the fixed point problem is equivalent to the following problem:

**Input:** A string-rewriting system $R$ on an alphabet $\Sigma$, and a string $\alpha \in \Sigma^+$. 

**Question:** Does there exist a string $W$ such that $\alpha W \xleftarrow{\ast} R W$?

This is a particular case of the Common Term Problem discussed in the next section and is thus decidable in polynomial time for finite, monadic and convergent string rewriting systems. It is also a particular case of the conjugacy problem. Thus for finite, length-reducing and convergent systems it is decidable in $NP$ [5]. The $NP$-hardness proof in [5] also applies in our particular case: thus the problem is $NP$-complete for finite, length-reducing and convergent systems.

Theorem 3.2 shows that fixed point problem is undecidable for finite and convergent string rewriting systems.

**Theorem 3.1.** The following problem is undecidable:

**Input:** A non-looping\(^1\), deterministic linear bounded automaton $M$ that restores its input for accepted strings.

**Question:** Is $L(M)$ empty?

**Proof.** Using the construction given in [2] in a straightforward way, we can prove the undecidability of this problem with a reduction from the well-known Post Correspondence Problem (PCP). Recall that an instance of PCP is a finite collection of pairs of strings over an alphabet $\Sigma$ (‘‘dominos’’ in [6]) and the question is if there exists a sequence of indices $i_1, i_2, \ldots, i_n$ such that $x_{i_1}x_{i_2} \ldots x_{i_n} = y_{i_1}y_{i_2} \ldots y_{i_n}$. Alternatively, we can define the PCP in terms of two homomorphisms $\psi$ and $\phi$ from

\(^1\)i.e., no configuration will be repeated. Thus, the DLBA will halt on all inputs.
\( C^* = \{c_1, \ldots, c_n\}^* \) to \( \Sigma^* = \{a, b\}^* \). The question now is whether there exists a non-empty string \( m_C \) such that \( \psi(m_C) = \phi(m_C) \).

The reduction proof is done by validating the “solution string” \( w \in \Sigma^* \), for the two homomorphisms using the DLBA \( \mathcal{M} \) with the following configuration, \( \mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r) \). Assume \( w = w_1 w_2 \) where \( w_1 \in C^* \) and \( w_2 \in \Sigma^* \). \( \mathcal{M} \) will check the correctness of \( w \) by going over one symbol at a time in the string \( w_1 \) and its corresponding mapping in \( w_2 \). \( \mathcal{M} \) uses a marking technique, marking the symbols \( c_i \) and their correct mappings with overlined symbols. If every letter in the string is overlined, then clearly the string is “okay” according to the first homomorphism; \( \mathcal{M} \) will then replace the overlined letters with the same non-overlined letters as before and repeat the same steps for the second homomorphism. Thus the DLBA will accept and re-create \( w_1 w_2 \) if and only if it is a solution string.

We construct a string rewriting system \( R \) from the above-mentioned DLBA \( \mathcal{M} \). The construction of \( R \) is similar to the one in [7]. Let \( \mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r) \). Here \( \Sigma \) denotes the input alphabet, \( \Gamma \) is the tape alphabet, \( Q \) is the set of states, \( q_0 \in Q \) is the initial state, \( q_a \in Q \) the accepting state and \( q_r \in Q \) the rejecting state. We assume that the tape has two end-markers: \( \hat{c} \in \Gamma \) denotes the left end-marker and \( \hat{s} \in \Gamma \) is the right end-marker. We also assume that on acceptance the DLBA comes to a halt at the left-end of the tape. Finally, \( \delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\} \) is the transition function of \( \mathcal{M} \).

The alphabet of \( R \) is \( \Gamma = \Sigma \cup \Sigma' \cup Q \) where \( \Sigma' \) is a replica of the alphabet \( \Sigma \) such that \( \Sigma' \cap \Sigma = \emptyset \). \( R \) has the rules

\[
q_i a_k \rightarrow a'_k q_j \quad \text{if} \ (q_i, a_k, q_j, R) \in \delta \\
q'_i q_i a_k \rightarrow q_j a_l a_k \quad \text{for all} \ a'_l \in \Sigma', \ \text{if} \ (q_i, a_k, q_j, L) \in \delta \\
q_a \hat{c} \rightarrow \hat{c}
\]

Since the linear bounded automaton is deterministic, \( R \) is locally confluent. Besides, \( \mathcal{M} \) ultimately always halts, and that means there will be no infinite chain of rewrites for \( R \), and thus \( R \) is terminating.
Lemma 3.1. $M$ accepts $w$ iff $q_0 \mapsto_R^+ q_a \mapsto_R w$.

Proof. By inspection of the rules we can see that $M$ makes the transition

$$u_1 q_1 v_1 \vdash_M u_2 q_2 v_2$$

if and only if $u'_1 q_1 v_1 \rightarrow_R u'_2 q_2 v_2$. □

Lemma 3.2. $\mapsto_R^+ w$ is a fixed point for $q_0$ in $R$ iff $M$ accepts $w$.

Proof. Suppose $M$ accepts $w$. Observe that $q_0 \mapsto_R^+ q_a \mapsto_R w$ by Lemma 3.1, and $R$ has the rule $q_a \mapsto \cdot$. Thus we get $q_0 \mapsto_R^+ q_a \mapsto_R w \rightarrow \mapsto_R^+ w$.

For the “only if” part, suppose $\mapsto_R^+ w$ is a fixed point for $q_0$, i.e., $q_0 \mapsto_R^+ q_a \mapsto_R w$. Now note that the only rule that can remove a state-symbol from a string is the rule $q_a \mapsto \cdot$. But once that rule is applied, no other rules are applicable. Therefore, there must be a reduction sequence such that $q_0 \mapsto_R^+ q_a \mapsto_R w$. This proves that $w$ is accepted by the DLBA $M$. □

Theorem 3.2. The fixed point problem is undecidable for finite and convergent string rewriting systems.
3.2 Common Term Problem

Note that for string rewriting systems the common term problem is equivalent to the following problem:

**Input:** A string-rewriting system $R$ on an alphabet $\Sigma$, and two strings $\alpha, \beta \in \Sigma^*$.

**Question:** Does there exist a string $W$ such that $\alpha W \leftrightarrow_R^* \beta W$?

This is also known as *Common Multiplier Problem* which has been shown to be decidable in polynomial time for monadic and convergent string-rewriting systems (see, e.g., [8], Lemma 3.7). However, the CT problem is undecidable for convergent and length-reducing string rewriting systems in general [9].

In this section, we focus on the decidability of the CT problem for *convergent and dwindling* string rewriting systems. The dwindling convergent systems are especially important because they are widely used in the field of protocol analysis; in particular, digital signatures, one-way hash functions and standard axiomatization of encryption and decryption. This class is also known as subterm-convergent theories in the literature [10, 11, 12, 13]. Tools such as TAMARIN prover [14] and YAPA [15] use subterm-convergent theories since these theories have nice properties (e.g., finite basis property [16]) and decidability results [10].

3.2.1 CT for Dwindling Systems

We show that the CT (Common Term) problem is undecidable for string rewriting systems that are dwindling and convergent. We define CT as the following decision problem:

**Given:** A finite, non-empty alphabet $\Sigma$, strings $\alpha, \beta \in \Sigma^*$ and a dwindling, convergent string rewriting system $S$.

**Question:** Does there exist a string $W \in \Sigma^*$ such that $\alpha W \approx_S \beta W$?

We show that the Generalized Post Correspondence Problem (GPCP) reduces to the CT problem, where GPCP stands for a variant of the modified post correspondence problem such that

---

1In fact, Otto et al. [8] showed that there is a fixed convergent length-reducing string rewriting system for which the CT problem is undecidable.
we will provide the start and finish dominoes in the problem instance. This slight change does not
effect the decidability of the problem in any way, i.e., GPCP is also undecidable [17, 18].

**Given:** A finite set of tuples \{(x_i, y_i)\}_{i=0}^{n+1} such that each \(x_i, y_i \in \Sigma^+\), i.e., for all \(i\), \(|x_i| > 0\), \(|y_i| > 0\), and \((x_0, y_0), (x_{n+1}, y_{n+1})\) are the start and end dominoes, respectively.

**Question:** Does there exist a sequence of indices \(i_1, \ldots, i_k\) such that

\[ x_0 x_{i_1} \ldots x_{i_k} x_{n+1} = y_0 y_{i_1} \ldots y_{i_k} y_{n+1} \]

We work towards showing that the CT problem defined above is undecidable by a many-one
reduction from GPCP. First, we show how to construct a string-rewriting system that is dwindling
and convergent from a given instance of GPCP.

Let \(\{(x_i, y_i)\}_{i=1}^n\) be the set of “intermediate” dominoes and \((x_0, y_0), (x_{n+1}, y_{n+1})\), the start
and end dominoes respectively, be given. Suppose \(\Sigma\) is the alphabet given in the instance of GPCP.
Without loss of generality, we may assume \(\Sigma = \{a, b\}\). Then set \(\hat{\Sigma} := \{a, b\} \cup \{c_0, \ldots, c_{n+1}\} \cup \{\hat{c}_1, \hat{c}_2, B, a_1, a_2, a_3, b_1, b_2, b_3\}\) which will be our alphabet for the instance of CT.

Next we define a set of string homomorphisms used to simplify the discussion of the reduc-
tion. Namely, we have the following:

\[
\begin{align*}
  h_1(a) &= a_1 a_2 a_3, & h_2(a) &= a_1 a_2, & h_3(a) &= a_1 \\
  h_1(b) &= b_1 b_2 b_3, & h_2(b) &= b_1 b_2, & h_3(b) &= b_1
\end{align*}
\]

such that each \(h_i : \Sigma \rightarrow \hat{\Sigma}^+\) is a homomorphism.

We are now in a position to construct the string rewriting system \(S\), with the following col-
lections of rules, named as the Class D rules:

\[
\begin{align*}
  \hat{c}_1 h_1(a) &\rightarrow \hat{c}_1 h_3(a), & \hat{c}_2 h_1(a) &\rightarrow \hat{c}_2 h_2(a) \\
  \hat{c}_1 h_1(b) &\rightarrow \hat{c}_1 h_3(b), & \hat{c}_2 h_1(b) &\rightarrow \hat{c}_2 h_2(b) \\
  h_i(a) h_1(a) &\rightarrow h_i(a) h_1(a), & h_i(a) h_1(b) &\rightarrow h_i(a) h_i(b) \\
  h_i(b) h_1(a) &\rightarrow h_i(b) h_1(a), & h_i(b) h_1(b) &\rightarrow h_i(b) h_i(b)
\end{align*}
\]
for $i \in \{2,3\}$.

The erasing rules of our system consists of three classes. Class I rules are defined as:

$$c_1 h_3(x_0) B c_0 \rightarrow \lambda$$
$$c_2 h_2(y_0) c_0 \rightarrow \lambda$$

and Class II rules (for each $i = 1,2,\ldots,n$),

$$h_3(x_i) B c_i \rightarrow \lambda$$
$$h_2(y_i) c_i B \rightarrow \lambda$$

and finally Class III rules,

$$h_3(x_{n+1}) c_{n+1} \rightarrow \lambda$$
$$h_2(y_{n+1}) c_{n+1} B \rightarrow \lambda$$

Clearly given an instance of $GPCP$ the above set of rules can effectively be constructed from the instance data. Also, by inspection, we have that our system is confluent (there are no overlaps among the left-hand sides of rules), terminating, and dwindling.

We then set $\alpha = c_1$ and $\beta = c_2$ to complete the constructed instance of $CT$ from $GPCP$.

It remains to show that this instance of $CT$ is a “yes” instance if and only if the given instance of $GPCP$ is a “yes” instance, i.e., the $CT$ has a solution if and only if the $GPCP$ does. In that direction, we prove some results relating to $S$.

**Lemma 3.3.** Suppose $c_1 h_3(w_1) B \gamma \rightarrow^1 \lambda$ and $c_2 h_2(w_2) \gamma \rightarrow^1 \lambda$ for some $w_1, w_2 \in \{a, b\}^*$, then $\gamma \in \{c_1 B, c_2 B, \ldots, c_n B\}^* c_0$.

*Proof.* Suppose $\gamma$ is a minimal counter example with respect to length and $\gamma \in IRR(R)$. In order for the terms to be reducible, $\gamma = c_i B \gamma'$ (this follows by inspection of $S$). After we replace the $\gamma$ at
the equation in the lemma, we get:
\[
\begin{align*}
\mathbf{c}_1 h_3(w_1) B c_i B \gamma' & \rightarrow \mathbf{c}_1 h_3(w_1)' B \gamma' \rightarrow^1 \lambda \\
\mathbf{c}_2 h_2(w_2) c_i B \gamma' & \rightarrow \mathbf{c}_2 h_2(w_2)' \gamma' \rightarrow^1 \lambda
\end{align*}
\]

by applying the Class II rules and finally Class I rule to erase the \( \mathbf{c} \) signs. Then, however, \( \gamma' \) is also a counterexample, and \(|\gamma'| < |\gamma|\), which is a contradiction. \(\square\)

We are now in a position to state and prove the main result of this section.

**Theorem 3.3.** The CT problem is undecidable for dwindling convergent string-rewriting systems.

*Proof.* We first complete the “only if” direction. Suppose CT has a solution such that \( \mathbf{c}_1 Z \downarrow \mathbf{c}_2 Z \) where \( Z \) is a minimal solution. We show that \( Z \) corresponds to a solution for \( GPCP \). Let \( Z = h_1(Z_1)Z_2 \) such that \( h_1(Z_1) \) is the longest prefix of \( Z \) such that the following relationship holds: \( Z = Z' Z_2 \) and \( Z' = h_1(Z_1) \) for some string \( Z_1 \).

\( h_1(Z_1) \) can be rewritten to \( h_3(Z_1) \) and \( h_2(Z_1) \) by applying the Class D rules. Thus, we will get

\[
\begin{align*}
\mathbf{c}_1 h_1(Z_1) Z_2 & \rightarrow^* \mathbf{c}_1 h_3(Z_1) Z_2 \\
\mathbf{c}_2 h_1(Z_1) Z_2 & \rightarrow^* \mathbf{c}_2 h_2(Z_1) Z_2
\end{align*}
\]

In order for terms to be reducible simultaneously, \( Z_2 \) must be of the form \( Z_2 = c_{n+1} B Z_2' \). Thus

\[
\begin{align*}
\mathbf{c}_1 h_3(Z_1) Z_2 & = \mathbf{c}_1 h_3(Z_1) c_{n+1} B Z_2' \\
\mathbf{c}_2 h_2(Z_1) Z_2 & = \mathbf{c}_2 h_2(Z_1) c_{n+1} B Z_2'
\end{align*}
\]

i.e., \( Z_1 = Z_1' x_{n+1} \) and \( Z_1 = Z_1'' y_{n+1} \). By applying the Class III rules, these equations will reduce to:

\[
\begin{align*}
\mathbf{c}_1 h_3(Z_1) c_{n+1} B Z_2' & \rightarrow \mathbf{c}_1 h_3(Z_1') B Z_2' \\
\mathbf{c}_2 h_2(Z_1) c_{n+1} B Z_2' & \rightarrow \mathbf{c}_2 h_2(Z_1'') Z_2'
\end{align*}
\]

We now apply Lemma 3.3 to conclude that \( Z_2' \in \{ c_1 B, c_2 B, \ldots, c_n B \}^* c_0 \).
At this point we have that:

\[ Z_2 = c_{i_1+1}Bc_{i_1}Bc_{i_2} \cdots Bc_{i_k}Bc_0 \text{ for some } i_1, \ldots, i_k \]

Then the sequence of dominoes

\[(x_0, y_0), (x_{i_1}, y_{i_1}), \ldots, (x_{i_k}, y_{i_k}), (x_{n+1}, y_{n+1})\]

will be a solution to the given instance of *GPCP* with solution string \( Z_1 \) since the left-hand sides of the Class I, II, III rules consist of the images of domino strings under \( h_2 \) and \( h_3 \). More specifically, there is a finite number of \( B \)'s and \( c_i \)'s in \( Z_2 \), so there must be a decomposition of \( h_1(Z_1) \):

\[
h_1(Z_1) = h_1(x_0)h_1(x_{i_1}) \cdots h_1(x_{i_k})h_1(x_{n+1})
\]

and

\[
h_1(Z_1) = h_1(y_0)h_1(y_{i_1}) \cdots h_1(y_{i_k})h_1(y_{n+1})
\]

Thus, we have the following reductions with Class D rules:

\[
\begin{align*}
&\mathcal{c}_1 h_1(Z_1) Z_2 \rightarrow^* \mathcal{c}_1 h_3(Z_1) Z_2 \\
&\mathcal{c}_2 h_1(Z_1) Z_2 \rightarrow^* \mathcal{c}_2 h_2(Z_1) Z_2
\end{align*}
\]

Finally, by Class I, II, III rules:

\[
\begin{align*}
&\mathcal{c}_1 h_3(Z_1) Z_2 \rightarrow^* \mathcal{c}_1 h_3(x_0) B c_0 \rightarrow \lambda \\
&\mathcal{c}_2 h_2(Z_1) Z_2 \rightarrow^* \mathcal{c}_2 h_2(y_0) c_0 \rightarrow \lambda
\end{align*}
\]

and \( Z_1 \) is a solution to the instance of the *GPCP*.

We next prove the “if” direction. Assume that the given instance of *GPCP* has a solution. Let \( w \) be the string corresponding to the matching dominoes, and let

\[(x_0, y_0), (x_{i_1}, y_{i_1}), \ldots, (x_{i_k}, y_{i_k}), (x_{n+1}, y_{n+1})\]
be the sequence of tiles that induces the match. Let $Z = c_{n+1}Bc_{i_1}Bc_{i_2} \cdots Bc_{i_k}Bc_0$. We show that $\zeta_1 h_1(w)Z \downarrow \zeta_2 h_1(w)Z$.

First apply the Class $D$ rules to get:

$$\zeta_1 h_1(w)Z \rightarrow^* \zeta_1 h_3(w)Z$$

$$\zeta_2 h_1(w)Z \rightarrow^* \zeta_2 h_2(w)Z$$

but then we can apply Class I, II, III rules to reduce both of the above terms to $\lambda$.  

This result strengthens the earlier undecidability result of Otto [9] for string-rewriting systems that are length-reducing and convergent.
3.3 Common Equation Problem

To clarify the CE problem for string rewriting systems, let us consider two substitutions $\theta_1$ and $\theta_2$ such that

$$\theta_1 = \{x_1 \mapsto \alpha_1, x_2 \mapsto \alpha_2\}$$
$$\theta_2 = \{x_1 \mapsto \beta_1, x_2 \mapsto \beta_2\}$$

Think of the letters of the alphabet as monadic function symbols as mentioned in Section 2.1. We have two cases for the equation $e_1 = e_2$: (i) both $e_1$ and $e_2$ have the same variable in them, or (ii) they have different variables, i.e., one has $x_1$ and the other $x_2$. Thus in the former, which we call the “one-mapping” case, we are looking for different irreducible strings $W_1$ and $W_2$ such that

1. $\alpha_1 W_1 \xrightarrow{\ast} R \alpha_1 W_2$ and $\beta_1 W_1 \xrightarrow{\ast} R \beta_1 W_2$, or
2. $\alpha_2 W_1 \xrightarrow{\ast} R \alpha_2 W_2$ and $\beta_2 W_1 \xrightarrow{\ast} R \beta_2 W_2$.

In the latter (“two-mappings case”) case, we want to find strings $W_1$ and $W_2$, not necessarily distinct, such that

$$\alpha_1 W_1 \xrightarrow{\ast} R \alpha_2 W_2 \quad \text{and} \quad \beta_1 W_1 \xrightarrow{\ast} R \beta_2 W_2$$

The one-mapping case can be illustrated by an example. Consider the term rewriting system

$$\{a(a(b(z))) \rightarrow a(b(z))\}$$

and two substitutions $\theta_1 = \{x \mapsto b(c)\}$ and $\theta_2 = \{x \mapsto b(b(c))\}$. Now $a(a(x)) = a(x)$ is a common equation. Considering this in the string rewriting setting, we have $R = \{baa \rightarrow ba\}$, $\alpha = b$ and $\beta = bb$. Now $W_1 = aa$ and $W_2 = a$ is a solution.

Hence we define CE as the following decision problem:

**Input:** A string-rewriting system $R$ on an alphabet $\Sigma$, and strings $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \Sigma^*$.  

**Question:** Do there exist irreducible strings $W_1, W_2 \in \Sigma^*$ such that one of the following conditions is satisfied?
i. \( \alpha_1 W_1 \xleftarrow{\ast} R \alpha_2 W_2, \quad \alpha_1 \neq \alpha_2 \lor \beta_1 \neq \beta_2 \)
\( \beta_1 W_1 \xleftarrow{\ast} R \beta_2 W_2, \)

ii. \( \alpha_i W_1 \xleftarrow{\ast} R \alpha_i W_2, \quad i \in \{1, 2\} \land W_1 \neq W_2 \)
\( \beta_i W_1 \xleftarrow{\ast} R \beta_i W_2, \)

CE is also undecidable for dwindling systems, since in the string-rewriting case CT is a
particular case of CE. To see this, consider the case where \( \alpha_1 \neq \alpha_2 \) and \( \beta_1 = \beta_2 = \lambda \), i.e., consider
the substitutions

\[
\theta_1 = \{ x_1 \mapsto \alpha_1, x_2 \mapsto \alpha_2 \} \\
\theta_2 = \{ x_1 \mapsto \lambda, x_2 \mapsto \lambda \}
\]

where \( \alpha_1 \neq \alpha_2 \). This has a solution if and only if there are irreducible strings \( W_1, W_2 \in \Sigma^* \) such
that either

i. \( \alpha_1 W_1 \xleftarrow{\ast} R \alpha_2 W_2, \) or
\( W_1 \xleftarrow{\ast} R W_2, \)

ii. \( \alpha_i W_1 \xleftarrow{\ast} R \alpha_i W_2, \quad i \in \{1, 2\} \land W_1 \neq W_2 \)
\( W_1 \xleftarrow{\ast} R W_2. \)

Since \( W_1 \) and \( W_2 \) are irreducible strings, \( W_1 \xleftarrow{\ast} R W_2 \) makes \( W_1 \) equal to \( W_2 \). Thus, we
eliminate the second condition.

With the second condition being out of the picture, we only consider the first condition which
shows a similarity with the definition of Common Term (CT) problem. The CE problem for \( \theta_1 \) and
\( \theta_2 \) has a solution if and only if the CT problem for \( \alpha_1 \) and \( \alpha_2 \) has a solution. Therefore, CT is
reducible to CE problem.

It can also be shown, using the construction from [19] that there are theories for which CT is
decidable whereas CE is not\(^3\).

We now show that CE is decidable for monadic string rewriting systems. We start by plunging
into the two-mappings case first, since the solution for one-mapping case is similar to the

\(^3\)We can use Corollary 4.4 on page 101 of [19]
two-mapping case with a slightly simpler approach.

3.3.1 Two-mapping CE Problem for Monadic Systems

For monadic and convergent string rewriting systems, the two-mappings case of the Common Equation (CE) problem is decidable. This can be shown using Lemma 3.6 in [8]. (See also Theorem 3.11 of [8].) In fact, the algorithm runs in polynomial time as explained below:

**Theorem 3.4.** Common Equation (CE) problem, given below, is decidable in polynomial time for monadic, finite and convergent string rewriting systems.

**Input:** A string-rewriting system $R$ on an alphabet $\Sigma$, and strings $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \Sigma^*$.

**Question:** Do there exist strings $X, Y \in \Sigma^*$ such that $\alpha_1 X \xrightarrow{R} \alpha_2 Y$ and $\beta_1 X \xrightarrow{R} \beta_2 Y$?

**Proof.** The CE problem is a particular case of the simultaneous E-unification problem of [8], but with a slight difference: CE consists of only two equations, while simultaneous E-unification problem is defined for an arbitrary number of equations. Besides, simultaneous E-unification problem is PSPACE-hard. We will use their construction, but we will modify it to obtain our polynomial time result.

Given a monadic, finite and convergent string rewriting system $R$ and irreducible strings $x$ and $y$, let $RF(x, y)$ define the set of right factors needed to derive $y$, i.e., $RF(x, y) = \{z \in IRR(R) \mid xz \xrightarrow{R} y\}$. $RF(x, y)$ is a regular language for all $x$, $y$ [8, 20] and a DFA for it can be constructed in time polynomial in $|R|$, $|x|$ and $|y|$. We can characterize the solutions of the equation $\alpha_1 X \xrightarrow{R} \alpha_2 Y$ by an analysis similar to that used in Lemma 3.6 of [8] and its proof. Since $R$ is monadic, there exist $a, b \in \Sigma \cup \{\lambda\}$ and partitions of the strings $\alpha_1 = \alpha_{11} \alpha_{12}$, $\alpha_2 = \alpha_{21} \alpha_{22}$, $X = X_1 X_2$ and $Y = Y_1 Y_2$, such that

\[
\begin{align*}
\alpha_{12} X_1 &\xrightarrow{'} a, \\
\alpha_{22} Y_1 &\xrightarrow{'} b, \quad \text{and} \\
\alpha_{11} a X_2 & = \alpha_{21} b Y_2.
\end{align*}
\]
Now there are 2 main cases:

(a) $X_2$ is a suffix of $Y_2$:

$$
Y_2 = Z \cdot X_2
$$

$$
\alpha_{11} \cdot a = \alpha_{21} \cdot b \cdot Z
$$

$$
RF(\alpha_{12}, a) \times RF(\alpha_{22}, b) \cdot Z
$$

(b) $Y_2$ is a proper suffix of $X_2$:

$$
X_2 = Z'' \cdot Y_2
$$

$$
\alpha_{11} \cdot a \cdot Z'' = \alpha_{21} \cdot b
$$

$$
\alpha_{11} \cdot a \cdot U = \alpha_{21} \cdot b \cdot Z'' = U \cdot b
$$

$$
RF(\alpha_{12}, a) \cdot Z'' \times RF(\alpha_{22}, b)
$$

Similar partitioning can be done for the second equation.

Let $Sol(\alpha_1, \alpha_2)$ stand for a set of ‘minimal’ solutions:

$$
Sol(\alpha_1, \alpha_2) = \bigcup_{a, b \in \Sigma \cup \{\lambda\}} \bigcup_{\alpha_{11} \cdot a = \alpha_{21} \cdot b \cdot Z} RF(\alpha_{12}, a) \times RF(\alpha_{22}, b) \cdot Z
$$

$$
\cup \bigcup_{a, b \in \Sigma \cup \{\lambda\}} RF(\alpha_{12}, a) \cdot Z'' \times RF(\alpha_{22}, b)
$$

Note that this is a finite union of cartesian products of regular languages. More precisely, it is an expression of the form

$$(L_{11} \times L_{12}) \cup \ldots \cup (L_{m1} \times L_{m2})$$

where $m$ is a polynomial over $|\alpha_1|$, $|\alpha_2|$ and $|\Sigma|$ and each $L_{ij}$ has a DFA of size polynomial in $|R|$ and max$(|\alpha_1|, |\alpha_2|)$.

To find the complexity of a DFA concatenation with a letter or string, check the Lemma B.1 for the former and Lemma B.2 for the latter.

The set of all solutions for the equation $\alpha_1 X = \alpha_2 Y$ is

$$
\Delta(\alpha_1, \alpha_2) = \left\{ (w_1, x_1, z_1 x_1) \mid (w_1, z_1) \in Sol(\alpha_1, \alpha_2) \text{ and } x_1 \in IRR(R) \right\}
$$

The minimal solutions for the second equation with $\beta_1$ and $\beta_2$, $Sol(\beta_1, \beta_2)$, can be found by
following the same steps. Thus \( \text{Sol}(\beta_1, \beta_2) \) can also be expressed as the union of cartesian products of regular languages:

\[
(L'_{11} \times L'_{12}) \cup \ldots \cup (L'_{n1} \times L'_{n2})
\]

where \( n \) is also a polynomial over \( |\beta_1|, |\beta_2| \) and \( |\Sigma| \). The set of all solutions for \( \beta_1X = \beta_2Y \) equals to

\[
\Delta(\beta_1, \beta_2) = \left\{ (w_2x_2, z_2x_2) \mid (w_2, z_2) \in \text{Sol}(\beta_1, \beta_2) \text{ and } x_2 \in \text{IRR}(R) \right\}
\]

The solutions for both the equations are the tuples \((w, z) \in \Delta(\alpha_1, \alpha_2) \cap \Delta(\beta_1, \beta_2)\). That is, there must be \( w_1, w_2, z_1, z_2, x_1, x_2 \) such that \((w_1, z_1) \in \text{Sol}(\alpha_1, \alpha_2), (w_2, z_2) \in \text{Sol}(\beta_1, \beta_2) \) and

\[
w = w_1 x_1 = w_2 x_2 \\
z = z_1 x_1 = z_2 x_2
\]

If \( x_1 \) is a suffix of \( x_2 \), i.e., \( x_2 = x_2' x_1 \), then

\[
w_1 = w_2 x_2' \\
z_1 = z_2 x_2'
\]

(Similarly we repeat the same steps when \( x_2 \) is a suffix of \( x_1 \).)

Recall that \((w_1, z_1) \in L_{i1} \times L_{i2} \) for some \( i > 0 \), and \((w_2, z_2) \in L'_{j1} \times L'_{j2} \) for some \( j > 0 \). Thus \( x'_2 \in L_{i1} \setminus L'_{j1} \) and \( x'_2 \in L_{i2} \setminus L'_{j2} \) where \( \setminus \) stands for the left quotient operation on languages, defined as \( A \setminus B := \{ v \in \Sigma^* \mid \exists u \in B : uv \in A \} \) (See Lemma B.3 for more detail). Thus there is a solution if the intersection of \( L_{i1} \setminus L'_{j1} \) and \( L_{i2} \setminus L'_{j2} \) is nonempty. This check has to be repeated for every \( i, j \). The process of finding the intersection of two languages is explained in Lemma B.4 and to be able to find the strings in the intersection and the quotient, you need to follow the steps in Lemma B.5.

### 3.3.2 One-mapping CE Problem for Monadic Systems

One-mapping case of the CE problem is decidable for monadic and convergent string rewriting systems. We can show it using a construction similar to the two-mappings case. However it will be a slightly simpler approach since we only have two input strings \( \alpha \) and \( \beta \) as opposed to four. The algorithm for the one-mapping case also runs in polynomial time as explained below:
Theorem 3.5. The following CE problem is decidable in polynomial time for monadic, finite and convergent string rewriting systems.

Input: A string-rewriting system $R$ on an alphabet $\Sigma$, and irreducible strings $\alpha, \beta \in \Sigma^*$.

Question: Do there exist distinct irreducible strings $X, Y \in \text{IRR}(R)$ such that $\alpha X \downarrow_R \alpha Y$ and $\beta X \downarrow_R \beta Y$?

Proof. This follows from Lemma C.9, since we only need to check whether there are strings $X', Y', V, \gamma_1, \gamma_2$ such that $X'$ and $Y'$ are distinct, $\gamma_1$ and $\gamma_2$ belong to $MP(\alpha)$ and $MP(\beta)$ respectively, and one of the following symmetric cases hold:

(a) $X'V, Y'V \in RF(\alpha, \gamma_1)$ and
(b) $X', Y' \in RF(\beta, \gamma_2)$

or

(c) $X', Y' \in RF(\alpha, \gamma_1)$ and
(d) $X'V, Y'V \in RF(\beta, \gamma_2)$

First of all there are only polynomially many strings in $MP(\alpha)$ and $MP(\beta)$. The two cases can be checked in polynomial time by Lemma B.5.
CHAPTER 4
Conclusion

In this dissertation we studied the fixed point problem (FP), common term problem (CT) and common equation problem (CE) for convergent string rewriting systems. These problems, which are dual to unification problem, are investigated for length-reducing, dwindling and monadic subclasses of finite and convergent string rewriting systems.

<table>
<thead>
<tr>
<th></th>
<th>Convergent</th>
<th>Length-reducing</th>
<th>Dwindling</th>
<th>Monadic</th>
</tr>
</thead>
<tbody>
<tr>
<td>FP</td>
<td>undecidable</td>
<td>NP-complete</td>
<td>NP</td>
<td>P</td>
</tr>
<tr>
<td>CT</td>
<td>undecidable</td>
<td>undecidable</td>
<td>undecidable</td>
<td>P</td>
</tr>
<tr>
<td>CE</td>
<td>undecidable</td>
<td>undecidable</td>
<td>undecidable</td>
<td>P</td>
</tr>
</tbody>
</table>

Table 4.1: Complexity results of the problems in String Rewriting Systems.

As the astute reader will note, the exact complexity of the fixed point problem for dwindling and convergent systems is not yet clear. All we know is that it is in NP [5]. Settling this issue is clearly an important task for the future.

Furthermore, for the sake of brevity (and clarity), we only discussed string rewriting systems in this paper. Our future work will include the investigation of these problems for general term rewriting systems.

We also plan to work on these problems for the following theories:

i. boolean algebra,

ii. linear algebra,

iii. abelian groups with homomorphisms.
Another idea for future work is to explore whether the common equation problem has applications in security. Also, we will look into a generalization of the Common Equation problem where more than two substitutions will be considered.
The following term rewriting system $R_{lin}^1$ specifies a fragment of linear arithmetic using successor and predecessor operators:

\[
\begin{align*}
    x - 0 & \rightarrow x \\
    x - x & \rightarrow 0 \\
    s(x) - y & \rightarrow s(x - y) \\
    p(x) - y & \rightarrow p(x - y) \\
    x - p(y) & \rightarrow s(x - y) \\
    x - s(y) & \rightarrow p(x - y) \\
    p(s(x)) & \rightarrow x \\
    s(p(x)) & \rightarrow x
\end{align*}
\]

We checked convergence of these rules on RRL (Rewrite Rule Laboratory) [21].
Lemma B.1. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and $a \in \Sigma$. Then there exists a DFA $M' = (Q', \Sigma, \delta', q'_0, F')$ that recognizes $L(M) \circ \{a\}$ such that $|F'| \leq |F|$ and $|Q'| \leq |Q| + |F|$.

Proof. The concatenation of a letter with a DFA can be easily achieved by adding extra transactions from each final state $q_{f_i}$ to a new state $p_i$ for the symbol $a$. However, that turns the DFA $M$ into a non-deterministic finite automaton (NFA). We claim that there exists a DFA $M'$ with $|F|$ as the upper bound for accepting states.

Consider Figure B.1: $q_{f_1}$ through $q_{f_n}$ are the $n$ final states of the given DFA $M$. Suppose $\delta(q_{f_i}, a) = r_i$. Then the subset construction gives us the new transition $\delta'(\{q_{f_i}\}, a) = \{r_i, p_i\}$. The new accepting states for the new DFA $M'$ will be $\{r_i, p_i\}$, such that $1 \leq i \leq n$.

![Diagram](image)

Figure B.1: DFA $M$ concatenation with a single letter $a$.

Besides, if the transitions for the letter $a$ from two earlier accepting states have the same destination state, we can combine the new accepting states that were created. Thus in Figure B.2, the state $\{r_1, p_1\}$ can be assigned to $\delta'(\{q_{f_n}\}, a)$, avoiding needless duplication.

Thus the number of final states, $|F'|$ for the DFA $M'$ is less than or equal to the original...
number of final states, $|F|$, in DFA $M$.

Total number of states $|Q'|$ for $M'$ is bounded by the number of the final states $|F|$ in $M$ as well as the number of total states, $|Q|$, in $M$. Therefore, the number of states for $M'$ can be less than or equal to the both of the factors, i.e., $|Q'| \leq |Q| + |F|$.

\[ \begin{array}{c}
\text{(a) } M \\
\includegraphics[width=0.4\textwidth]{example1.png} \\
\text{(b) } M'
\end{array} \]

Figure B.2: DFA $M'$ can have less than $|F|$ states.
Lemma B.2. Concatenation of a deterministic finite automaton (DFA) with a single string has the time complexity $O(|F| \cdot |Z| \cdot |\Sigma|)$, where $|F|$ is the number of final states in the DFA, $|Z|$ is the length of the string and $|\Sigma|$ is the size of the alphabet.

Proof. Recall the previous lemma proved that the number of states in the new DFA after the concatenation of one letter is at most $|Q| + |F|$ and that the number of (new) accepting states is at most $F$. Thus repeatedly applying this operation will result in a DFA with at most $|Q| + |Z| \cdot |F|$ states and at most $|F|$ accepting states. The number of new edges will be at most $|F| \cdot |Z| \cdot |\Sigma|$. Thus the overall complexity is polynomial in the size of the original DFA.

Lemma B.3. Let $M_1$ and $M_2$ be DFAs. Then a multiple-entry DFA (MEFA) for $L(M_2) \setminus L(M_1)$ can be computed in polynomial time.

Proof. Let $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ and let $L_{\text{quo}} = L(M_2) \setminus L(M_1)$. A string $y$ belongs to $L_{\text{quo}}$ if and only if there exists a string $x$ and states $p \in Q_1$, $p' \in F_1$ and $q \in F_2$ such that $\delta^*_2(q_{02}, x) = q$, $\delta^*_1(q_{01}, x) = p$ and $\delta^*_1(p, y) = p'$.

Let $P$ be the product transition system of the two automata, i.e., $P = (Q_1 \times Q_2, \Sigma, \delta, (q_{01}, q_{02}))$ where $\delta$ is defined as

$$\delta((r, r'), c) = (\delta_1(r, c), \delta_2(r', c))$$
for all $c \in \Sigma$, $r \in Q_1$ and $r' \in Q_2$. We can assume that states which are not reachable from $(q_{01}, q_{02})$ have been removed. This can be done in time linear in the size of the transition graph. Now in terms of the transition system we can say that a string $y$ belongs to $L_{\text{quo}}$ if and only if there exists a string $x$ and states $p \in Q_1$, $p' \in F_1$, $q \in F_2$ and $q' \in Q_2$ such that

$$\delta^*((q_{01}, q_{02}), x) = (p, q) \text{ and } \delta^*((p, q), y) = (p', q')$$

We can now convert $P$ into a multiple-entry DFA (MEFA). In the above case, $(p, q)$ has to be one of the initial states of the new MEFA, and $(p', q')$ one of its final states. Therefore, the states that are reachable from $(q_{01}, q_{02})$ that are in $F_2 \times Q_1$ will be the initial states of the MEFA and $F_1 \times Q_2$ will be the final states of the MEFA.

**Lemma B.4.** Given two MEFAs, we can check whether their intersection is empty in polynomial time.

**Proof.** Consider two MEFAs $A_1 = (Q_1, \Sigma, \delta_1, Q_{s1}, F_{A1})$ and $A_2 = (Q_2, \Sigma, \delta_2, Q_{s2}, F_{A2})$. $Q_{s1}$ and $Q_{s2}$ may include more than one initial state. A string $w$ is accepted by both MEFAs if and only if there exist states $q_{\text{init}}^1 \in Q_{s1}$ and $q_{\text{init}}^2 \in Q_{s2}$ such that $\delta_1^*(q_{\text{init}}^1, w) \in F_{A1}$ and $\delta_2^*(q_{\text{init}}^2, w) \in F_{A2}$.

To find such a string $w$, we take the product transition system of the two MEFAs, named as $T$, i.e., $T = (Q_1 \times Q_2, \Sigma, \delta, (Q_{s1} \times Q_{s2}))$ where

$$\delta((r, r'), c) = (\delta_1(r, c), \delta_2(r', c))$$

for all $c \in \Sigma$, $r \in Q_1$ and $r' \in Q_2$. A string $w$ is accepted by both of the MEFAs $A_1$ and $A_2$ if and only if there exist states $p, q, p', q'$ such that $p \in Q_{s1}$, $q \in Q_{s2}$, $p' \in F_{A1}$ and $q' \in F_{A2}$, and

$$\delta^*((p, q), w) = (p', q')$$

We can now apply depth-first search (DFS) to check, in time linear in the size of $T$, if there exists a path from some state in $Q_{s1} \times Q_{s2}$ to a state in $F_{A1} \times F_{A2}$. \qed
Lemma B.5. The following problem is decidable in polynomial time:

**Input:** DFAs $M$ and $N$.

**Question:** Do there exist strings $x, y, z$ such that $x \neq y$, $x, y \in \mathcal{L}(M)$, and $xz, yz \in \mathcal{L}(N)$?

**Proof.** Suppose there exist strings $x, y, z$ such that $x \neq y$, $x, y \in \mathcal{L}(M)$, and $xz, yz \in \mathcal{L}(N)$. We call the triple $(x, y, z)$ a solution. Thus, we have two cases:

(i) Both $x$ and $y$ start from an initial state $q_0$ and reach the same state, $q$, in $N$, i.e.,

$$\exists q : \delta^*(q_0^N, x) = \delta^*(q_0^N, y) = q \quad \text{and} \quad \delta^*(q, z) \in F^N.$$ 

(ii) $x$ and $y$ reach different states, say $q'$ and $q''$, in $N$, i.e.,

$$\exists q', q'' : \delta^*(q_0^N, x) = q' \neq q'' = \delta^*(q_0^N, y) \quad \text{and} \quad \delta^*(q', z) \in F^N \land \delta^*(q'', z) \in F^N.$$ 

Let $A = (Q, \Sigma, \delta, s, F)$ be a DFA and $p$ be a state in $A$. By $A^F=\{p\}$, we denote a replication of $A$, with the sole difference of $p$ being the only accepting state of $A$. Thus $N^F=\{q\}$ denotes a replication of $N$, with $q$ being the accepting state of $N$. Then, we classify these states of $N$ which are not dead states into GREEN, ORANGE and BLUE states. Note that confirming the status of $q$ being a dead state can be done in linear time w.r.t. to the size of graph.

- **GREEN states:** $\{q \mid |\mathcal{L}(N^F=\{q\}) \cap \mathcal{L}(M)| > 1\}$.

  The state $q$ mentioned in case (i) is a GREEN state.

- **ORANGE states:** $\{q \mid |\mathcal{L}(N^F=\{q\}) \cap \mathcal{L}(M)| = 1\}$.

  Suppose that case (i) does not apply, i.e., there are no GREEN states in $N$. Then case (ii) must apply and the states $q'$ and $q''$ must be ORANGE states; in other words, the intersection of $\mathcal{L}(M)$ individually with the two DFAs, $\mathcal{L}(N^F=\{q'\})$ and $\mathcal{L}(N^F=\{q''\})$ gives us exactly 1 string for each. Note also that $x$ and $y$ are two strings in $\mathcal{L}(M)$ which are not equal to each other since $q' \neq q''$. 

33
• **BLUE** states: \( \{ q \mid |\mathcal{L}(N^F=q) \cap \mathcal{L}(M)| = 0 \} \).

![Diagram of Green States](image1)

![Diagram of Orange States](image2)

Figure B.4: Different Structures of Green and Orange States.

The algorithm for finding the triple \((x, y, z)\) is constructed as follows. First, we identify the green and orange states. If there exists a green state, then we have a solution. Otherwise we explore whether there exists a \(z\) such that \(\delta^*(q', z) \in F^N \land \delta^*(q'', z) \in F^N\) for orange states \(q'\) and \(q''\), i.e., we check whether

\[
\{ z \mid \exists (q', q''): q', q'' \text{ are orange states } \land \delta^*(q', z) \in F^N \land \delta^*(q'', z) \in F^N \}
\]

is empty.

Given orange states \(q'\) and \(q''\), we use DFA intersection to check whether there is a string \(z\) that takes both to an accepting state. Let \(N_s=q'\) denote a replication of \(N\), with the difference of \(q'\) being the initial state of \(N\). \(N_s=q''\) is similar to the \(N_s=q'\), but this time \(q''\) is the initial state. After creating these two DFAs, we can find if there exists a string \(z\) by intersecting the DFA \(N_s=q'\) with \(N_s=q''\). This process may have to be repeated for every tuple \((q', q'')\) of orange states.

\(\square\)
APPENDIX C

Appendix C

This corollary is a slightly simpler construction than the one in [19] for the reader’s ease of understanding.

**Corollary C.1.** There is a finite and convergent string system for which the common term (CT) problem is decidable, while the common equation (CE) problem is undecidable.

We show that CE is undecidable by a reduction from GPCP. (See Section 3.2.1.) For notational convenience, we represent an instance of the GPCP as a 3-tuple

\[
\left\langle \left[ \frac{x}{y} \right], S, \left[ \frac{u}{v} \right] \right\rangle
\]

where \((x,y)\) is the start domino, \((u,v)\) is the end domino and \(S\) the set of intermediate dominos.

**Theorem C.1** ([22]). There exist strings \(\alpha, \beta,\) and a set of tuples of strings \(S\) such that following problem is undecidable:

**Input:** Strings \(x_0, y_0.\)

**Question:** Does the GPCP \(\left\langle \left[ \frac{x_0}{y_0} \right], S, \left[ \frac{\alpha}{\beta} \right] \right\rangle\) have a solution?

Let \(\left\langle \left[ \frac{x_0}{y_0} \right], \{(x_i, y_i)\}_{i=1}^{n}, \left[ \frac{x_{n+1}}{y_{n+1}} \right] \right\rangle\) be an instance of the GPCP, i.e., \(\{(x_i, y_i)\}_{i=1}^{n}\) is the set of intermediate dominos and \((x_0, y_0), (x_{n+1}, y_{n+1})\), the start and end dominos respectively, where the strings are over the alphabet \(\Sigma = \{a, b\}\). Let \(\hat{\Sigma} = \{a, b\} \cup \{c_0, \ldots, c_n\} \cup \{c_1, c_2, $, #_1, #_2\}\) be the new alphabet for the instance of CE.

From the given instance, we construct a string-rewriting system \(R_1\) with the following rules:
Lemma C.1. $R_1$ is convergent.

Lemma C.2. GPCP has a solution if and only if there exist strings $w_1, w_2$ such that $x_0 \hat{c}_1 w_1 \rightarrow^! \#_1 w_2$ and $y_0 \hat{c}_2 w_1 \rightarrow^! \#_2 w_2$.

Proof. We prove the “if” direction first. Assume GPCP has a solution. Let $i_1, \ldots, i_k$ be the indices of the intermediate dominoes used, i.e., there are $k + 2$ dominoes in all including the start and end dominoes. This will result in $x_0 x_{i_1}, \ldots, x_{i_k} x_{n+1} = y_0 y_{i_1}, \ldots, y_{i_k} y_{n+1}$. Let $w_1 = c_{i_1} \ldots c_{i_k} c_{n+1}$ and $w_2 = x_0 x_{i_1} \ldots x_{i_k} x_{n+1}$, then

\[
\begin{align*}
x_0 \hat{c}_1 c_{i_1} \ldots c_{i_k} c_{n+1} & \rightarrow x_0 x_{i_1} \hat{c}_1 \ldots c_{i_k} c_{n+1} \rightarrow^! \#_1 x_0 x_{i_1} \ldots x_{i_k} x_{n+1} \\
y_0 \hat{c}_2 c_{i_1} \ldots c_{i_k} c_{n+1} & \rightarrow y_0 y_{i_1} \hat{c}_2 \ldots c_{i_k} c_{n+1} \rightarrow^! \#_1 y_0 y_{i_1} \ldots y_{i_k} y_{n+1}
\end{align*}
\]

For the “only if” direction suppose there exist strings $w_1$ and $w_2$ such that $x_0 \hat{c}_1 w_1 \rightarrow^! \#_1 w_2$ and $y_0 \hat{c}_2 w_1 \rightarrow^! \#_2 w_2$. Without loss of generality, we can assume that $w_1$ and $w_2$ are irreducible. The observation of the rules on the both sides shows that $\#_1 w_2$ and $\#_2 w_2$ can be derived if and only if $c_{n+1}$ occurs in $w_1$ since only the rules with $c_{n+1}$ has a \( \hat{c} \) on the left-hand side (LHS) and a \# symbol on the RHS.

Thus, we can write $w_1$ as $w_1 = w'_1 c_{n+1} w''_1$ such that $w'_1 c_{n+1}$ is the shortest prefix of $w_1$ that contains $c_{n+1}$. To be able to apply the rules with \( \hat{c} \) signs, $w'_1$ should be in $\{c_1, \ldots, c_n\}^*$. Let $w'_1 = c_{i_1} \ldots c_{i_k}$ where $k = |w'_1|$.

\[
\begin{align*}
x_0 \hat{c}_1 w'_1 c_{n+1} & \rightarrow^* \#_1 x_0 x_{i_1} \ldots x_{i_k} x_{n+1} \\
y_0 \hat{c}_2 w'_1 c_{n+1} & \rightarrow^* \#_2 y_0 y_{i_1} \ldots y_{i_k} y_{n+1}
\end{align*}
\]
Since $ does not occur on the left hand side of the rules, the string after the $ sign, i.e., $w''_1$, does not take part in the reductions. Thus

\[ x_0 \cdot c_1 \cdot w'_1 \cdot c_{n+1} \cdot w''_1 \rightarrow^{\dagger} #1 \cdot x_0 \cdot x_i \cdot \ldots \cdot x_k \cdot x_{n+1} \cdot $ \quad \text{w''_1} \\
\]

\[ y_0 \cdot c_2 \cdot w'_1 \cdot c_{n+1} \cdot w''_1 \rightarrow^{\dagger} #2 \cdot y_0 \cdot y_i \cdot \ldots \cdot y_k \cdot y_{n+1} \cdot $ \quad \text{w''_1} \\
\]

Thus it must be that $x_0 \cdot x_i \cdot \ldots \cdot x_k \cdot x_{n+1} = y_0 \cdot y_i \cdot \ldots \cdot y_k \cdot y_{n+1}$, which is a solution to the GPCP. 

\[ \square \]

**Theorem C.2.** The CE problem is undecidable for the finite and convergent string rewriting system $R_1$.

**Lemma C.3.** For all $a \in \hat{\Sigma}$ and strings $Z_1$ and $Z_2$, $Z_1 a \downarrow Z_2 a$ if and only if $Z_1 \downarrow Z_2$.

**Proof.** Since $R_1$ is convergent, all we need to prove is that if $Z_1, Z_2 \in \text{IRR}(R_1)$ and $a \in \hat{\Sigma}$, then $Z_1 a \downarrow Z_2 a$ if and only if $Z_1 \downarrow Z_2$.

Let $c \in \hat{\Sigma}$ such that $Z_1 c \downarrow Z_2 c$ and $Z_1 \neq Z_2$ where $Z_1$ and $Z_2$ are irreducible strings. Clearly either $Z_1 c$ or $Z_2 c$ must be reducible. Thus it has to be that $c \in \{c_1, \ldots, c_{n+1}\} \cup \{\#_1, \#_2\}$. We now need to consider three cases:

i. $c \in \{c_1, \ldots, c_n\}$:
   
   The observation of the rules and the set $c$ belongs shows that, $Z_1$ and $Z_2$ should end with $c_1$ or $c_2$ to make $Z_1 c \downarrow Z_2 c$. Let $i$ be an index, such that $0 \leq i \leq n$ and $Z_1$ and $Z_2$ can be written as $Z_1 = Z'_1 c_1$ and $Z_2 = Z'_2 c_1$:

   \[ Z'_1 c_1 \rightarrow Z'_1 x_i c_1 \]
   \[ Z'_2 c_1 \rightarrow Z'_2 x_i c_1 \quad \text{(C.1)} \]

   Since $Z'_1, Z'_2 \in \text{IRR}(R_1)$, so are $Z'_1 x_i$ and $Z'_2 x_i$, since no left-hand sides end with either $a$ or $b$. Hence, $Z'_1 = Z'_2$.

ii. $c = \#_1$:

   Suppose $Z_1 = Z'_1 Z''_1$ and $Z_2 = Z'_2 Z''_2$ such that $Z''_1$ and $Z''_2$ are the longest suffixes of $Z_1$ and $Z_2$ that belong to $\{a, b\}^*$. Thus:

   \[ Z'_1 Z''_1 \#_1 \rightarrow^{\dagger} Z'_1 \#_1 Z''_1 \]
   \[ Z'_2 Z''_2 \#_1 \rightarrow^{\dagger} Z'_2 \#_1 Z''_2 \quad \text{(C.2)} \]
Since \( Z'_1 \#_1 Z''_1 \) and \( Z'_2 \#_1 Z''_2 \) are irreducible, \( Z'_1 = Z'_2 \) and \( Z''_1 = Z''_2 \).

iii. \( c = \#_2 \):

Follows the same construction as in the previous case, with the only difference being \( \#_2 \) instead of \( \#_1 \).

iv. \( c = c_{n+1} \):

Thus,

\[
\begin{align*}
Z'_1 \; c_{n+1} & \rightarrow Z'_1 \; \#_1 \; x_{n+1} \; \$
\end{align*}
\]

\[
\begin{align*}
Z'_2 \; c_{n+1} & \rightarrow Z'_2 \; \#_1 \; x_{n+1} \; $
\end{align*}
\]

Since there are no left-hand sides that end with \( a, b \) or \( \$ \), we have \( Z'_1 \; \#_1 \downarrow Z'_2 \; \#_1 \). By the case (ii) above, we get \( Z'_1 = Z'_2 \).

\[\Box\]

**Theorem C.3.** The CT problem is decidable for the finite and convergent string rewriting system \( R_1 \).
The Figure C.1 illustrates how $\alpha X$ reduces to its normal form $\alpha_1 a X_2$. ($\alpha$ and $X$ are irreducible strings.)

Let $R$ be a convergent monadic SRS. For an irreducible string $\alpha$, let

$$MP(\alpha) = \left\{ w \mid w \in PREF(\alpha) \circ (\Sigma \cup \{\epsilon\}) \right\}$$

$MP$ stands for the term Minimal Product and $PREF$ is the set of prefixes of given string.

**Lemma C.4.** Let $\mu, \omega, X, Y \in IRR(R)$. Then $\mu X \downarrow \omega Y$ if and only if there exist strings $X', Y', W, \gamma$ such that

1. $\gamma \in MP(\mu) \cup MP(\omega)$,
2. $X = X'W$, $Y = Y'W$, and
3. $\mu X' \xrightarrow{\gamma} R \gamma \xrightarrow{\omega Y'}$.

**Proof.** This proof as well as the proof for Lemma C.5 follows from [8] (see Lemma 3.6).
Lemma C.5. Let $\alpha_1, \alpha_2, \beta_1, \beta_2, X, Y \in \text{IRR}(R)$. Then $\alpha_1 X \downarrow \alpha_2 Y$ and $\beta_1 X \downarrow \beta_2 Y$ if and only if there exist strings $X', Y', V, W, \gamma_1, \gamma_2$ such that

1. $\gamma_1 \in \text{MP}(\alpha_1) \cup \text{MP}(\alpha_2)$,
2. $\gamma_2 \in \text{MP}(\beta_1) \cup \text{MP}(\beta_2)$,
3. $X = X'VW, Y = Y'VW$, and
4. either
   
   (a) $\alpha_1 X'V \xrightarrow{1} R \gamma_1 \xleftarrow{1} \alpha_2 Y'V$ and
   
   (b) $\beta_1 X' \xrightarrow{1} R \gamma_2 \xleftarrow{1} \beta_2 Y'$.

   or

   (c) $\alpha_1 X' \xrightarrow{1} R \gamma_1 \xleftarrow{1} \alpha_2 Y'$ and

   (d) $\beta_1 X'V \xrightarrow{1} R \gamma_2 \xleftarrow{1} \beta_2 Y'V$.

Corollary C.2. Let $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \text{IRR}(R)$. Then there exist irreducible strings $X$ and $Y$ such that $\alpha_1 X \downarrow \alpha_2 Y$ and $\beta_1 X \downarrow \beta_2 Y$ if and only if there exist strings $X', Y', V, \gamma_1, \gamma_2$ such that

1. $\gamma_1 \in \text{MP}(\alpha_1) \cup \text{MP}(\alpha_2)$,
2. $\gamma_2 \in \text{MP}(\beta_1) \cup \text{MP}(\beta_2)$,
3. $X'V$ and $Y'V$ are irreducible, and
4. either
   
   (a) $\alpha_1 X'V \xrightarrow{1} R \gamma_1 \xleftarrow{1} \alpha_2 Y'V$ and
   
   (b) $\beta_1 X' \xrightarrow{1} R \gamma_2 \xleftarrow{1} \beta_2 Y'$.

   or

   (c) $\alpha_1 X' \xrightarrow{1} R \gamma_1 \xleftarrow{1} \alpha_2 Y'$ and

   (d) $\beta_1 X'V \xrightarrow{1} R \gamma_2 \xleftarrow{1} \beta_2 Y'V$. 
Lemma C.6. Let $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \text{IRR}(R)$. Then there exist irreducible strings $X$ and $Y$ such that $\alpha_1X \downarrow \alpha_2Y$ and $\beta_1X \downarrow \beta_2Y$ if and only if there exist strings $X', Y', V, \gamma_1, \gamma_2$ such that

1. $\gamma_1 \in \text{MP}(\alpha_1) \cup \text{MP}(\alpha_2)$,
2. $\gamma_2 \in \text{MP}(\beta_1) \cup \text{MP}(\beta_2)$,
3. either
   (a) $X'V \in \text{RF}(\alpha_1, \gamma_1)$, $Y'V \in \text{RF}(\alpha_2, \gamma_1)$, $X' \in \text{RF}(\beta_1, \gamma_2)$ and $Y' \in \text{RF}(\beta_2, \gamma_2)$
   or
   (b) $X' \in \text{RF}(\alpha_1, \gamma_1)$, $Y' \in \text{RF}(\alpha_2, \gamma_1)$, $X'V \in \text{RF}(\beta_1, \gamma_2)$ and $Y'V \in \text{RF}(\beta_2, \gamma_2)$

Lemma C.7. Let $\mu, X, Y \in \text{IRR}(R)$ where $X \neq Y$. Then $\mu X \downarrow \mu Y$ if and only if there exist strings $X'$, $Y'$, $W, \gamma$ such that

1. $\gamma \in \text{MP}(\mu)$,
2. $X = X'W$, $Y = Y'W$, $X' \neq Y'$ and
3. $\mu X' \xrightarrow{r} \gamma \xleftarrow{r} \mu Y'$.

Proof. Let $Z$ be the normal form of $\mu X$ and $\mu Y$. Then there exists strings $\mu_1, \mu_2, \mu_3, \mu_4, X_1, X_2, Y_1, Y_2$ such that $\mu = \mu_1 \mu_2 = \mu_3 \mu_4$, $X = X_1 X_2$, $Y = Y_1 Y_2$ and

$\mu_2 X_1 \rightarrow^{\ast} a,$
$\mu_4 Y_1 \rightarrow^{\ast} b,$ and
$Z = \mu_1 a X_2 = \mu_3 b Y_2.$
where \( a, b \in \Sigma \cup \{ \lambda \} \). If \( X_1 = Y_1 \), then the same reduction can be applied on both sides, i.e., \( \mu_2 = \mu_4 \) and \( a = b \). But the rest of the string \( X_2 \neq Y_2 \) since \( X \neq Y \). Therefore, we conclude that \( X_1 \neq Y_1 \). It can also be seen that \( \mu_1 a, \mu_3 b \in MP(\alpha) \).

We now consider two cases:

(a) \( X_2 \) is a suffix of \( Y_2 \): Let \( Y_2 = Y'_2X_2 \). Then \( \mu_1 a = \mu_3 b Y'_2 \). We can take \( \gamma = \mu_1 a, X' = X_1, Y' = Y_1Y'_2 \) and \( W = X_2 \).

(b) \( Y_2 \) is a suffix of \( X_2 \): Let \( X_2 = X'_2Y_2 \). Then \( \mu_1 aX'_2 = \mu_3 b \). In this case we can take \( \gamma = \mu_3 b, X' = X_1X'_2, Y' = Y_1 \) and \( W = X_2 \).

Lemma C.8. Let \( \alpha, \beta \in IRR(R) \). Then there exist distinct irreducible strings \( X \) and \( Y \) such that \( \alpha X \downarrow \alpha Y \) and \( \beta X \downarrow \beta Y \) if and only if there exist irreducible strings \( X', Y', V, W, \gamma_1, \gamma_2 \) such that

1. \( X' \neq Y' \),
2. \( \gamma_1 \in MP(\alpha) \),
3. \( \gamma_2 \in MP(\beta) \),
4. \( X = X'VW, Y = Y'VW \), and
5. either

   (a) \( \alpha X'V \xrightarrow{\gamma_1} \alpha Y'V \) and
   
   (b) \( \beta X' \xrightarrow{\gamma_2} \beta Y' \).

   or

   (c) \( \alpha X' \xrightarrow{\gamma_1} \alpha Y' \) and

   (d) \( \beta X'V \xrightarrow{\gamma_2} \beta Y'V \).
Proof. Assume that there exist strings \( X', Y', V, \gamma_1, \gamma_2 \) that satisfy the properties above. Let us consider the fifth property. It shows that \( \alpha X'V \Downarrow \alpha Y'V \) as well as \( \beta X'V \Downarrow \beta Y'V \). Now suppose \( X = X'V \) and \( Y = Y'V \). Therefore, we can see that \( \alpha X \Downarrow \alpha Y \) as well as \( \beta X \Downarrow \beta Y \) and \( X \) and \( Y \) are distinct irreducible strings.

Conversely, assume that there exist distinct irreducible strings \( X \) and \( Y \) such that \( \alpha X \Downarrow \alpha Y \) and \( \beta X \Downarrow \beta Y \). We start by considering the case \( \alpha X \Downarrow \alpha Y \) such that \( X \neq Y \). By Lemma C.7, there must be strings \( X', Y', Z \) and \( \gamma_1 \) such that \( X = X'Z, Y = Y'Z \) and \( \alpha X' \xrightarrow{1} R \gamma_1 \xleftarrow{1} R \alpha Y' \) where \( \gamma_1 \in MP(\alpha) \). Similarly, for the case \( \beta X \Downarrow \beta Y \), \( X \) can be written as \( X''Z' \) and \( Y \) can be written as \( Y''Z' \) such that \( \beta X'' \xrightarrow{1} R \gamma_2 \xleftarrow{1} R \beta Y'' \) for some \( \gamma_2 \in MP(\beta) \).

We have to consider two cases depending on whether \( X' \) is a prefix of \( X'' \) or vice versa. It is not hard to see that they correspond to the two cases in condition 5. \( \Box \)

Corollary C.3. Let \( \alpha, \beta \in \text{IRR}(R) \). Then there exist distinct irreducible strings \( X \) and \( Y \) such that \( \alpha X \Downarrow \alpha Y \) and \( \beta X \Downarrow \beta Y \) if and only if there exist irreducible strings \( X', Y', V, \gamma_1, \gamma_2 \) such that

1. \( X' \neq Y' \),
2. \( \gamma_1 \in MP(\alpha) \),
3. \( \gamma_2 \in MP(\beta) \),
4. \( X'V \) and \( Y'V \) are irreducible, and
5. either
   
   (a) \( \alpha X'V \xrightarrow{1} R \gamma_1 \xleftarrow{1} R \alpha Y'V \) and
   
   (b) \( \beta X' \xrightarrow{1} R \gamma_2 \xleftarrow{1} R \beta Y' \).

or
Lemma C.9. Let $\alpha, \beta \in IRR(R)$. Then there exist distinct irreducible strings $X$ and $Y$ such that $\alpha X \downarrow \alpha Y$ and $\beta X \downarrow \beta Y$ if and only if there exist strings $X', Y', V, \gamma_1, \gamma_2$ such that

1. $X' \neq Y'$,
2. $\gamma_1 \in MP(\alpha)$,
3. $\gamma_2 \in MP(\beta)$,
4. either
   1. $X'V, Y'V \in RF(\alpha, \gamma_1)$ and
   2. $X', Y' \in RF(\beta, \gamma_2)$
   or
   1. $X', Y' \in RF(\alpha, \gamma_1)$ and
   2. $X'V, Y'V \in RF(\beta, \gamma_2)$
BIBLIOGRAPHY


