TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

ON SOME MODIFICATIONS AND APPLICATIONS OF THE POST CORRESPONDENCE PROBLEM

Esa Sahla

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
SARJA – SER. AI OSA – TOM. 665 | ASTRONOMICA – CHEMICA – PHYSICA – MATHEMATICA | TURKU 2022

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

ON SOME MODIFICATIONS AND APPLICATIONS OF THE POST CORRESPONDENCE PROBLEM

Esa Sahla

TURUN YLIOPISTON JULKAISUJA – ANNALES UNIVERSITATIS TURKUENSIS
SARJA – SER. AI OSA – TOM. 665 | ASTRONOMICA – CHEMICA – PHYSICA – MATHEMATICA | TURKU 2022

University of Turku

Faculty of Science
Department of Mathematics and Statistics
Mathematics
Doctoral Programme in Exact Sciences

Supervised by

Professor Vesa Halava
Department of Mathematics and Statistics
University of Turku
Finland

Professor Tero Harju
Department of Mathematics and Statistics
University of Turku
Finland

Reviewed by

Assistant Professor Hendrik Jan Hoogeboom
Leiden Institute of Advanced Computer Science
Leiden University
The Netherlands

Dr Hab Olivier Finkel
Equipe de Logique Mathématique
Institut de Mathématiques de Jussieu - Paris Rive Gauche CNRS and Université de Paris
France

Opponent

Doctor Paul Bell
School of Computing and Mathematical Sciences
Liverpool John Moores University
United Kingdom

ISBN 978-951-29-8787-0 (PRINT)
ISBN 978-951-29-8788-7 (PDF)
ISSN 0082-7002 (PRINT)
ISSN 2343-3175 (ONLINE)
Painosalama, Turku, Finland, 2022

UNIVERSITY OF TURKU
Faculty of Science
Department of Mathematics and Statistics
Mathematics
SAHLA, ESA: On Some Modifications and Applications of the Post Correspondence Problem
Doctoral dissertation, 53 pp.
Doctoral Programme in Exact Sciences
February 2022

ABSTRACT

The Post Correspondence Problem was introduced by Emil Post in 1946. The problem considers pairs of lists of sequences of symbols, or words, where each word has its place on the list determined by its index. The Post Correspondence Problem asks does there exist a sequence of indices so that, when we write the words in the order of the sequence as single words from both lists, the two resulting words are equal. Post proved the problem to be undecidable, that is, no algorithm deciding it can exist. A variety of restrictions and modifications have been introduced to the original formulation of the problem, that have then been shown to be either decidable or undecidable. Both the original Post Correspondence Problem and its modifications have been widely used in proving other decision problems undecidable.

In this thesis we consider some modifications of the Post Correspondence Problem as well as some applications of it in undecidability proofs. We consider a modification for sequences of indices that are infinite to two directions. We also consider a modification to the original Post Correspondence Problem where instead of the words being equal for a sequence of indices, we take two sequences that are conjugates of each other. Two words are conjugates if we can write one word by taking the other and moving some part of that word from the end to the beginning. Both modifications are shown to be undecidable.

We also use the Post Correspondence Problem and its modification for injective morphisms in proving two problems from formal language theory to be undecidable; the first problem is on special shuffling of words and the second problem on fixed points of rational functions.

KEYWORDS: Post correspondence problem, undecidability, word shuffling, fixed point

TURUN YLIOPISTO
Matemaattis-luonnontieteellinen tiedekunta
Matematiikan ja tilastotieteen laitos
Matematiikka
SAHLA, ESA: On Some Modifications and Applications of the Post Correspondence Problem
Väitöskirja, 53 s.
Eksaktien tieteiden tohtoriohjelma
Helmikuu 2022

TIIVISTELMÄ

Postin vastaavuusongelman esitteli alun perin Emil Post vuonna 1946. Ongelma koskee merkkijonoista, tai sanoista, koostuvia listapareja, joissa sanojen indeksit merkitsevät niiden paikkaa listassa. Postin vastaavuusongelma kysyy onko olemassa indeksijonoa, jolla kahdesta annetusta listasta näillä indekseillä valitut sanat peräkkäin kirjoitettuina tuottavat saman sanan. Post itse osoitti ongelman ratkeamattomaksi, eli näytti ongelman ratkaisevan algoritmin olemassaolon mahdottomaksi. Alkuperäisen ongelman asetteluun on vuosien saatossa esitetty erilaisia rajoituksia ja muunnelmia, jotka on edelleen osoitettu joko ratkeaviksi tai ratkeamattomiksi. Sekä alkuperäistä Postin vastaavuusongelmaa, että sen muunnelmia, on käytetty muiden päätösongelmien ratkeamattomuuden osoittamiseen.

Tässä väitöskirjassa tarkastellaan joitakin Postin vastaavuusongelman muunnelmia ja sovelluksia muiden ongelmien ratkeamattomuuden osoittamisessa. Tarkastelussa on ongelman muunnelma indeksilistoille, jotka ovat äärettömiä sekä oikealle että vasemmalle. Työssä tarkastellaan myös alkuperäisen vastaavuusongelman muunnelmaa, jossa yhden indeksijonon sijaan sanojen on oltava samat kahdelle indeksijonolle, jotka ovat toistensa konjugaatteja. Sanat ovat keskenään konjugaatteja, jos toisen saa muodostettua toisesta siirtämällä jonkin mittaisen osan sanan lopusta sen alkuun. Molemmat näistä muunnelmista näytetään ratkeamattomiksi.

Lisäksi Postin vastaavuusongelmaa ja sen muunnelmaa injektiivisille morfismeille käytetään todistettaessa kaksi muuta ongelmaa ratkeamattomiksi; ensimmäinen koskee sanojen sekoittamista ja toinen tietynlaisten funktioiden kiintopisteitä.

ASIASANAT: Postin vastaavuusongelma, ratkeamattomuus, sanojen sekoittaminen, kiintopiste

Acknowledgements

I have my deepest gratitude towards my supervisors Professor Vesa Halava and Professor Tero Harju for guiding me throughout my PhD studies. Their teachings, even before my PhD studies, were some of the most memorable I have. Thank you for introducing me to a variety of interesting problems. Some of these problems may be unanswered today, but we will sort them out by tomorrow.

I am also grateful to Assistant Professor Hendrik Jan Hoogeboom and Dr Hab Olivier Finkel, the pre-examiners of my thesis as well as Doctor Paul Bell for agreeing to act as my opponent.

I am grateful to all my previous teachers who showed me the beauty in discrete mathematics. I would like to offer my special thanks to Professor Juhani Karhumäki for introducing me to combinatorics on words and Professor Jarkko Kari for his brilliancies in tilings and automata theory. I wish to thank the entire staff at the department of mathematics and statistic for their hospitality and positivity. While my attendance at the office was rare in recent times, I appreciated the cracking of jokes at break times and the chats we had. Thanks to all of my roommates that kept me company: Bishwesvar, Debangana, Shemunyenge, Markus, Tuomo, Teemu and Juho (and any others I may have forgotten, looks like I changed my office more often than my socks).

I would like to thank the doctoral program MATTI, the Väisälä foundation and the Jenny and Antti Wihuri foundation for the financial support.

I am grateful for the support and love of my mother Erja, father Juha and sister Sanna. You are always home when you are with your family. I am also grateful for the support of my parents-in-law Mirja and Jyri. We have gotten a lot of invaluable help and knowledge from you as well as a little bit of ice cream for the kids.

Lastly, but most sincerely, I am grateful to my wife Taru and our children Hertta and Gösta. Taru, thank you for standing by me when I was doubtful or had a rough day. Hertta and Gösta, thank you for reminding me that laughter and play are a part of every day.

Pöytyä, February 2022
Esa Sahla

Table of Contents

List of Original Publications

This dissertation is based on the following original publications:

I Vesa Halava, Tero Harju and Esa Sahla. A New Proof for Undecidability of the Bi-Infinite Post Correspondence Problem. Fundamenta Informaticae, 2017; 154: 167-176.

II Vesa Halava, Tero Harju and Esa Sahla. On fixed points of rational transductions. Theoretical Computer Science, 2018; 732: 85-88.

III Vesa Halava, Tero Harju and Esa Sahla. On Shuffling a Word with its Letter-to-Letter Substitution. Fundamenta Informaticae, 2020; 175: 201-206.

IV Vesa Halava, Tero Harju and Esa Sahla. On the Conjugate Post Correspondence Problem. Manuscript.

1 Introduction

The chapters in this thesis cover multiple different topics. All of these topics belong to the field of formal language theory but the most prevalent unifying theme is the presence of the Post Correspondence Problem, PCP for short, which we will introduce a bit later. Basically in the PCP we are looking for a solution as a string of symbols (or a word) to an equation where simple rules are applied to each symbol on that string on both sides of the equation. The problem has turned out to be so complex that it can in fact simulate any given computer algorithm. This thesis showcases this property of the Post Correspondence Problem as well as presents some new constructions utilizing it and its computational power.

The theory of computability is an area in theoretical computer science which studies computational problems and, in particular, the question whether a problem can or cannot be solved by known models of computation. In the context of this thesis computable is synonymous with Turing computable i.e., computation that can be carried out by a Turing machine. *Decision problems* are a type of problems where the answer is given as a ”yes” or a ”no” depending on the input to the problem. Valid suitable inputs to these problems are called *instances*. A typical mathematical decision problem asks whether some instance to the problem possesses or does not possess a given property. If there exists an algorithm that *decides* the problem, that is, it answers correctly to all instances of the problem, we call the problem *decidable*. If on the other hand no such algorithm exists, we call the problem *undecidable*. The existence of an algorithm means that one can construct a Turing machine simulating it.

By the above one can show a problem to be decidable by laying an algorithm that decides it or showing that an algorithm exists. To show a problem to be undecidable one must prove that no algorithm deciding that problem exists. Usually showing the undecidability of a given problem utilizes some other problem known in advance to be undecidable. This method of proof is called reduction: an instance of a known undecidable problem is effectively transformed into an instance of the problem at hand. The transformation is such that if the problem at hand were to be decidable then it would also simultaneously decide the undecidable problem, a contradiction that forces the problem at hand to be undecidable as well. This method is used in this thesis to prove undecidability results.

The main problem of interest in this thesis is the Post Correspondence Problem

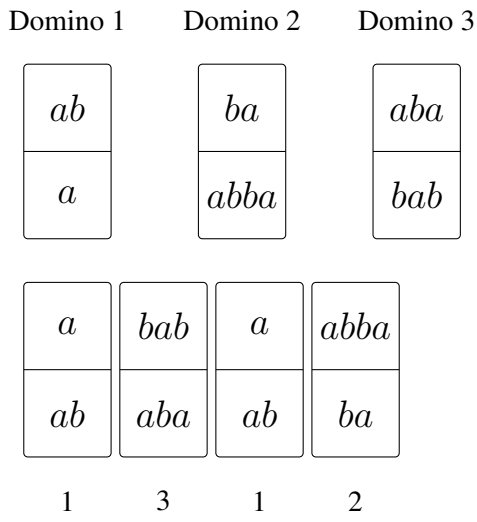

Figure 1. With a set of three dominoes a solution is found by lining them up in the order 1312.

introduced by Emil Post. An instance of the Post Correspondence problem consists of two lists of words $\{u_1, \ldots, u_n\}$ and $\{v_1, \ldots, v_n\}$, and it asks whether there exists a non-empty sequence of indices $i_1, \ldots, i_k$ such that $u_{i_1} \cdots u_{i_k} = v_{i_1} \cdots v_{i_k}$ or not. Another equivalent formulation of the problem is by (monoid) morphisms: given two morphisms $h, g : A^\star \to A^\star$ does there exist a non-empty word $w \in A^\star$ such that $h(w) = g(w)$. The problem is often illustrated with domino pieces having two words on top of each other, one piece for each index on the lists above. In a domino piece for a certain index the word on the top is the word with that index on the first list and the bottom one is similarly from the second list. For each index you can use as many dominoes as you wish. The dominoes are put next to each other so that the tops and bottoms both will spell some word. If the dominoes can be laid next to each other so that the top word is the same as the bottom word, we have a solution to our problem (see the example in Figure 1). This problem can then be varied by giving restrictions to the above-mentioned rules, for example that the domino set has only a certain amount of dominoes or that you must make an infinite strip with them. In this thesis we consider different variants of the Post Correspondence Problem as well as use the PCP to show other problems to be undecidable.

The Post Correspondence Problem belongs to the category of problems that are seemingly simple to solve. Indeed, if we were to search for a solution to a particular instance of the PCP we first choose one domino to begin with. For that piece there is most likely some factor that is left over after we match the words on the top and bottom from the beginning (if there is no such overflow then the single domino is a solution). Having this overflow we then search for a domino which allows us to match this overflow with the (top or bottom word depending on where the overlap

is) word on that domino making sure that everything matches after adding both the top and bottom words. Adding a new domino may then create a new overflow. The process of elimination of overflows continues until there is a perfect match or no continuation is possible. The problem becomes difficult when there are multiple choices for continuation and the overflows increase in length with no signs of shortening. Often there is no way to tell if searching for a solution becomes a lost cause after a while because the possibility of matching perfectly cannot be predicted. Even for "small" instances of the PCP the solutions can be very long: for the 3 pair instance $((aaba, a), (baab, aa), (a, aab))$ the shortest solution is 252 pairs long (Zhao, [1]) and for the 4 pair instance $((a, aa), (aaaa, abab), (aaab, ba), (bab, b))$ the shortest solution consists of 781 pairs (Rahn, [1]).

Emil Post proved undecidability of the PCP originally by a reduction from the assertion (or decision) problem of normal systems which was earlier proved to be an undecidable problem by Post himself. Nowadays, a standard textbook proof for undecidability of the PCP is a straightforward reduction from the halting problem of the Turing machines; see eg. [2] .

As with many other algorithmic problems, the aspect of *bounds* of undecidability are of interest. By a bound we mean some threshold that once crossed turns the problem from decidable to undecidable. With PCP this usually involves the number of words in the lists (i.e., the number of dominoes in the analogy, also called pairs when the words of the same index are paired) given above. It is known that the PCP is decidable for the sets of two pairs; see [3; 4], and undecidable for sets of five pairs by a proof by Neary [5]. Neary used a reduction from *binary tag-systems* to the PCP. The proof uses a similar technique that Matiyasevich and Sénizergues used in their proof for undecidability of the word problem for 3-rule semi-Thue systems in [6] which led to undecidability of the PCP for 7 pairs of words using the construction found by Claus [7] where the word problem of semi-Thue systems is reduced to the PCP. This reduction by Claus is technically very clear and elegant and, therefore, for many variants of the PCP, the simplest known undecidability proofs (usually implying rather good undecidability bounds) are modifications of Claus's construction. In this thesis modifications to Claus's construction are also used in Chapters 3 and 4.

The infinite variant of the PCP asks if there exists a solution that has infinitely many dominoes next to each other so that the resulting top and bottom pairs match no matter how far you go. It is easy to see that if an instance of the PCP has a finite solution then it also has an infinite one by repeating one solution over and over again infinitely many times. The question still remains meaningful as instances can have only infinite solutions as the following trivial example shows: let us have a PCP instance with the pairs (ab, a) and (ab, ba). If the pairs were named 1 and 2 respectively, then the unique infinite solution is $1222\cdots$. The instance has no finite solutions as the same overflow is repeated and there are no alternative ways to continue after the forced beginning with the first pair. This one-way infinite PCP is also

an undecidable problem with the undecidability following directly from the halting problem for Turing machines; see [8] for the first proof. For a fixed set of pairs it is known that the infinite PCP is decidable for instances of two pairs; see [4] for the proof by Halava et al., and undecidable for 8 pairs of words; see [9] for the result by Dong and Liu. In this thesis we consider the problem where the solutions are required to be bi-infinite, i.e., they are infinite in two directions. For the matchings in this case we require that some shift exists so that the images line up. Again a finite solution implies also a bi-infinite one and an ultimately periodic infinite solution implies a bi-infinite solution; say with a unique pair (ab, ba). Still instances admitting only bi-infinite solutions exist, for example in some cases where a starting pair for possible finite solution does not even exist. A trivial two pair instance admitting only bi-infinite solutions is the one with pairs (ab, ba) and (ba, ab). In fact any bi-infinite sequence of these pairs is a solution to the bi-infinite instance but no finite or one-way infinite solutions exist. The first proof for the undecidability of the bi-infinite PCP was given by Ruohonen in [8]. We will look at this problem more in Chapter 3 and give a new construction to show the undecidability.

The structure of the thesis is the following: Chapter 2 is dedicated to preliminaries and notations used and needed later in the thesis. In Chapter 3 we consider the above mentioned bi-infinite variant of the PCP and give a new proof for its undecidability by constructing a special semi-Thue system. A similar construction is then used to show another variant of the PCP, the conjugate PCP, to be undecidable as well. The conjugate PCP asks whether an instance of the PCP has a solution which produces conjugate words under the morphisms, that is, whether the words to be "matched" are of the form uv and vu respectively for some non-empty words u and v. In the last two chapters we use the PCP as a tool to reduce other problems into it to ensure their undecidability. Chapter 4 involves the operation of shuffling of words. Two words are shuffled together by catenating factors from both words starting from the beginning until both words are exhausted. In particular in this chapter we look at a form of self-shuffling where a word is shuffled with itself under a letter-to-letter substitution. In Chapter 5 a new problem is introduced. The problem involves rational transductions and fixed points of injective rational functions. Here another variant of the PCP is needed: the injective PCP, a variant where the morphisms are injective. The injective PCP is known to be undecidable as shown by Ruohonen [10].

2 Preliminaries

In this chapter we introduce some basic notations and results that are needed throughout the thesis. Main part of the chapter consists of the most fundamental things needed about words and formal languages. For a more comprehensive take on these subjects the reader is directed to the vast literature that exists. Great books about formal languages are *Formal Languages* [11] by Salomaa and *Introduction to Formal Language Theory* [12] by Harrison. For books on words the recommended classics are the M. Lothaire books *Combinatorics on Words* [13] and *Algebraic Combinatorics on Words* [14]. Great all-around books are *The Handbook of Formal Languages* [15] Rozenberg and Salomaa, eds., and *Introduction to Automata Theory, Languages, and Computation* [2] by Hopcroft and Ullman, which is a reference to many results in this chapter. Other chapters on this thesis will also have introductory segments where the prerequisites for that particular chapter are given.

2.1 Words

A set of symbols is called an *alphabet*. In this thesis all alphabets will be finite with $|A|$ denoting the number of symbols or *letters* in the alphabet A. In literature the most commonly used alphabets contain symbols from the Latin alphabet or the Arabic numerals with the most common ones being the binary alphabets $\{0, 1\}$ and $\{a, b\}$.

Let A be an alphabet. A (finite) *word* over A is a finite sequence of symbols in A. The length of a word w is the number of symbols in the sequence, denoted by $|w|$. The free monoid $A^\star$ on A with the concatenation operation is the set of all finite sequences of symbols from A, hence the set $A^\star$ denotes all finite words over A. The sequence with zero symbols (and of length zero) is called the *empty word* and denoted by ε. The set of all non-empty words over A is A^+. Thus $A^+ \cup \{\varepsilon\} = A^\star$. For the set of words of length n we use the notation A^n.

In this thesis as a rule (with exceptions) indexing of the letters in a word begins with 1, that is, the first letter of the word has index 1 and the last letter of a word has the index $|w|$. We refer to the letter at index i in word w by $w(i)$. The concatenation of two words $u = a_1 \cdots a_i$ and $v = b_1 \cdots b_j$ is the word $u{\cdot}v = uv = a_1 \cdots a_i b_1 \cdots b_j$ and the power u^n is the word $uu \cdots u$ where the word u appears n times. If two words u, v have exactly the same sequence of letters then they are equal and we can

write $u = v$. If there are non-empty words w, t and for some (possibly empty) words u, v we have that $w = utv$, then t is called a *factor* of w. When t is a factor of w we may denote it by $t \in w$. If t is a factor of w such that $w = tu$ for a word u, then t is called a *prefix* of w. Similarly if $w = vt$ for a word v, then t is called a *suffix* of w. Two words are called prefix-comparable (resp. suffix-comparable) if one of them is the prefix (resp. suffix) of the other. We call two words u, v *conjugates* if there exist factors s, t such that $u = st$ and $v = ts$. Conjugates of a word $w = a_1 \cdots a_n$ can be generated by cyclic k shifts where a letter at the ith position is moved to position $i - k \pmod n$. The cyclic nature of conjugate words is also behind the naming of the circular PCP, a variant of the PCP where the matching is acquired via two words that are conjugates (more on this in Chapter 3).

The deletion of a prefix or suffix u from a word w is denoted by $u^{-1}w$ or wu^{-1} respectively. For example, when $w = abbaba$ and $u = aba$, we have $wu^{-1} = abb$. Later we also use the notation a^{-n} as a shorthand for $(a^n)^{-1}$. The *reversal* of a word $w = a_1a_2 \cdots a_{n-1}a_n$ is denoted by $w^R = a_na_{n-1} \cdots a_2a_1$.

An *infinite word* over A is a sequence of symbols of A indexed by the numbers in $\mathbb{N}$, that is,

$$\omega = \omega(1)\omega(2)\omega(3) \cdots ,$$

where $\omega(i) \in A$ for all $i \in \mathbb{N}$. The set $A^\star$ contains all the words of finite length, hence this set does not include any infinite words. The set of infinite words over alphabet A is denoted by A^ω. For infinite powers of words we use the notation $w^\omega = www \cdots$ for a word w.

A *bi-infinite word* over A is a sequence of symbols of A indexed by the numbers in $\mathbb{Z}$, that is,

$$\omega = \cdots \omega(-3)\omega(-2)\omega(-1)\omega(0)\omega(1)\omega(2)\omega(3) \cdots ,$$

where $\omega(i) \in A$ for all $i \in \mathbb{Z}$. We denote the set of all bi-infinite words over A by $A^\mathbb{Z}$. We use a similar notation for bi-infinite powers: for example $(ab)^\mathbb{Z} = \cdots ababababab \cdots = (ba)^\mathbb{Z}$. For finite and infinite words the equality of words is straightforward: two words are equal if the letters are the same on each index. For bi-infinite words, on the other hand, we can shift one of the words we are comparing and get a letter-to-letter matching.

Let A and B be alphabets. A mapping $h : A^\star \to B^\star$ is a *morphism* if $h(uv) = h(u)h(v)$ for all $u, v \in A^\star$. Morphisms become defined by the images of the letters as the images of words are concatenations of the images of its letters. We call a morphism h *injective* if for all $u, v \in A^\star$, $h(u) = h(v)$ implies $u = v$. We say that a word w has an h*-cover* if w can be expressed as an image under morphism h, in other words, if there exists a word $u \in A^*$ such that $h(u) = w$.

The next example illustrates the use of morphisms as well as the matching of two bi-infinite words.

Example. Let h, g be morphisms defined by

$$\begin{aligned} h(a) &= a & g(a) &= b \\ h(b) &= bab & g(b) &= ab \\ h(c) &= bb & g(c) &= abb \end{aligned}$$

Let us write out the word $(abc)^{\mathbb{Z}} = \cdots abca\mathbf{b}cabc\cdots$, where one letter b is written in bold so that we can follow its image.

Now

$$\begin{aligned} h((abc)^{\mathbb{Z}}) &= \cdots ababbba\mathbf{bab}bbababbb\cdots \\ g((abc)^{\mathbb{Z}}) &= \cdots bababbb\mathbf{ab}abbbababb\cdots \end{aligned}$$

with the images of this bolded b also written in bold. Let the first letters of the bold images be our reference point. With respect to this point the words do not match, but if we shift the bottom word by one position to the left or five positions to the right we have a matching. Here the matching is obvious everywhere because of the periodicity of the bi-infinite word.

2.2 Languages

In formal language theory we are interested in languages that have some specific common properties. These common formalisms or forms of definitions define different families of languages that all have common characteristic properties inside their family. There are typically two ways to define formal languages; with generators or with acceptors. Generators are usually formal grammars with production rules that tell how the words in the language are generated from a simple starting point. Acceptors are automata of some type with no output such that words in a language are the ones accepted by the particular machine. Different types of grammars generate different families of languages. In the 1950's Chomsky described a hierarchy of classes of formal grammars (dubbed the Chomsky hierarchy). The hierarchy is as follows:

- Type-0 grammars: Recursively enumerable languages
- Type-1 grammars: Context-sensitive languages
- Type-2 grammars: Context-free languages
- Type-3 grammars: Regular languages

The lower the type number, the more expressive or general the languages are. In this thesis we are mostly working with questions regarding regular languages, so

on the bottom of the hierarchy. However, in Chapter 4 the *context-free* languages also make an appearance. Note also that studying a decision question and proving it to be undecidable, we actually prove that the sets of YES and NO instances are not recursively numerable languages. Indeed, both YES and NO instances being recursively enumerable languages is equivalent to the problem being decidable; see [2].

Any set of words is called a *language*. Languages can contain a finite or an infinite number of words. In particular the set $A^\star$ is the infinite language containing every word of finite length consisting of letters from A, sometimes called the *universal language* over A. Every language (of finite words) over the alphabet (or over a subset of A) is a subset of $A^\star$. As with alphabets the notation $L^\star$ is defined as the language generated by the language L, that is, $L^\star$ is the set of all finite concatenations of words from L. Note that the empty word is always in $L^\star$ although it may be absent in L. The star operation is often called the *Kleene closure operation*. Other common operations on languages are the set operations union ($L \cup M$) and intersection ($L \cap M$) as well as concatenation (LM).

Regular languages are the languages that can be defined by regular expressions. Equivalently regular languages are those that can be generated by regular grammars or accepted by finite automata (more equivalent definitions also apply). Here we will give a recursive formal definition for regular languages over alphabet A as follows:

1. The empty language $\emptyset$ is a regular language. For each letter $a \in A$ the singleton language $\{a\}$ is a regular language.

2. If L_1 and L_2 are regular languages, then $L_1 \cup L_2$, $L_1 L_2$ and $L_1^\star$ are regular languages.

3. No other language over A is a regular language.

In particular all finite languages are regular. When we are working with formal languages we often want to use some operations on them and the resulting language may not belong to the same family as the original language(s). We say that a family is $closed$ under an operation if by applying that operation to a member or members of the family yields a member of that family. The regular languages are closed under union, concatenation, Kleene star, complementation and intersection. First three of these operations follow from the definition given above. For the complementation ($\overline{L}$ = strings not in L sharing the same alphabet) we recall that regular languages are those accepted by finite automata. Taking a (complete)deterministic finite automaton accepting L and changing the accepting states to non-accepting states, and vice versa, we get the finite automaton accepting $\overline{L}$. Closure under intersection follows now for example from de Morgan's law: $L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$.

The context-free languages are the next class above regular languages in the Chomsky hierarchy and they are unsurprisingly generated by context-free grammars.

The context-free languages also are exactly the languages accepted by pushdown automata. All regular languages are context-free but not all context-free languages are regular. Where a regular language can "keep count" of one thing, context-free languages can do that for two things. For example the language $\{a^n b^n \mid n \geq 0\}$ is context-free but not regular as regular languages cannot count the number of written $a's$ and produce the same number of $b's$. It follows that the closure properties of these two families of languages are not the same. For the operations considered above for regular languages, the context-free languages are closed under union, concatenation and Kleene star, but are not closed under complementation or intersection; see [2]. Even though the intersection of two context-free languages is not necessarily context-free the next result is an useful one (see [2] for a proof):

***Lemma* 1.** *The intersection of a context-free language and a regular language is a context-free language.*

The result becomes obvious while considering pushdown automata. Both regular and context-free languages are also closed under taking morphic and inverse morphic images [2].

Typical decision problems faced in formal language theory are the membership, emptiness and universality problems.

Problem (Membership)**.** *For a language $L \subset A^\star$ satisfying certain properties, and a word $w \in A^\star$, is $w \in L$?*

Problem (Emptiness)**.** *Is a given language L satisfying certain properties the empty language, that is, is $L = \emptyset$?*

Problem (Universality)**.** *Is a given language L over A satisfying certain properties the universal language, that is, is $L = A^\star$?*

To make the above problems algorithmical decision problems the languages L in them must be given effectively, usually either by an accepting automaton or defining grammar. The universality and emptiness problems are connected: asking whether a given language L is empty is equivalent to asking weather the complement language $\overline{L}$ is universal. For regular languages all of the three mentioned decision problems are decidable: membership is decided by running the finite automaton on the input word, emptiness is decided similarly by checking whether an accepting computation exists for the given automaton and universality is decided by constructing an automaton for the complement language and checking its emptiness.

For context free languages the emptiness and membership problems are decidable, while the universality problem is an undecidable problem [2]. The emptiness problem for context-free languages is important for us, and therefore we will give it as the following lemma.

***Lemma* 2.** *The emptiness problem for context-free languages is decidable.*

2.3 The Post Correspondence Problem

We will adopt the following formal definition for the Post Correspondence Problem (PCP for short):

Problem (The Post Correspondence Problem)**.** *Given two morphisms $h, g : A^\star \to B^\star$, does there exist a non-empty word $w \in A^\star$ such that $h(w) = g(w)$?*

A given pair of morphisms (h, g) is an *instance* of the PCP. A word w that satisfies $h(w) = g(w)$ is a *solution* to the instance (h, g).

Numerous variants of the PCP exist where the solutions or the morphisms are restricted by various conditions. We will mention here the variants that are used or referred to later in this thesis. The first variant defined here was already discussed in Chapter 1.

Problem (The Infinite Post Correspondence Problem)**.** *Given two morphisms $h, g : A^\star \to B^\star$, does there exist an infinite word $w \in A^\omega$ such that $h(w) = g(w)$?*

The next variant is the main focus of Chapter 3. It expands the infiniteness of solutions to two directions:

Problem (The Bi-infinite Post Correspondence Problem ($\mathbb{Z}$PCP))**.** *Given two morphisms $h, g\colon A^* \to B^*$, does there exist a bi-infinite word $\omega : \mathbb{Z} \to A$ such that $h(\omega) = g(\omega)$?*

Next we will define the *injective PCP*, first proved to be undecidable by Lecerf in [16]; see also Ruohonen [8], and Karhumäki and Saarela [17].

Problem (The Injective Post Correspondence Problem)**.** *Given two injective morphisms $h, g : A^\star \to B^\star$, does there exist a non-empty word $w \in A^\star$ such that $h(w) = g(w)$?.*

The next variant we present here is the *circular PCP*. Its undecidability was originally also proved by Ruohonen in [8]. In fact the variant proved there was even a stronger one, called the n-permutation PCP, where instead of conjugation the operation between the words $h(u)$ and $g(v)$ is an n-permutation. For the n-permutation PCP see also the proof by Ernvall et al. in [18]. Recall that two words u and v are conjugates if there exist words s and t such that $u = st$ and $v = ts$.

Problem (The Circular Post Correspondence Problem)**.** *Given two morphisms $h, g : A^\star \to B^\star$, does there exist two words $u, v \in A^\star$ with $uv \neq \varepsilon$ such that $h(uv) = g(vu)$?*

Lastly we introduce the *conjugate-PCP*. This variant will be discussed further in Chapter 4.

Problem (Conjugate Post Correspondence Problem)**.** *Given two morphisms $h, g : A^* \to B^*$, does there exist a word $w \in A^+$ such that $h(w) = uv$ and $g(w) = vu$ for some words $u, v \in B^*$?*

Semi-Thue Systems and Claus Instances

One useful family on PCP instances are the *Claus instances*. To define Claus instances we first consider the *semi-Thue systems*, a type of string rewriting systems needed in the definition of the Claus instances.

A *semi-Thue system* T is a rewriting system defined as a tuple (Σ, R) where $\Sigma = \{a_1, a_2, \ldots, a_n\}$ is a finite alphabet, the elements of which are called *generators* of T, and $R \subseteq \Sigma^* \times \Sigma^*$ is a finite relation. The elements of R are called the *rules* of T. The rules are extended to strings over $\Sigma^\star$ by allowing the rewriting of substrings according to R. We write $u \rightarrow_T v$, if there exists a rule $(x, y) \in R$ such that $u = u_1 x u_2$ and $v = u_1 y u_2$ for some words u_1 and u_2. We denote by $\rightarrow_T^*$ the reflexive and transitive closure of $\rightarrow_T$, and by $\rightarrow_T^+$ the transitive closure of $\rightarrow_T$. Note that the index T is usually omitted from the notation, i.e., we shall write $\rightarrow$, when the semi-Thue system studied is clear from the context. If $u \rightarrow^* v$ in T, we say that there is a derivation from u to v in T.

If the relation R is symmetric, then T is a *Thue system* and then T corresponds to a semigroup with generators Σ and relations R.

The following problem is an important and useful one:

Problem (The Word Problem for semi-Thue Systems)**.** *Given a semi-Thue system $T = (\Sigma, R)$ and two words u and v, does there exist a derivation $u \rightarrow_T^* v$?*

The first proofs for undecidability of the word problem of (semi-)Thue systems were given independently by Post and Markov in 1947 [19; 20].

Given an instance (h, g) of the PCP where $h, g : \Sigma^\star \rightarrow \Gamma^\star$ for $\Gamma = \{a_1, \ldots, a_n\}$ we can transform it into an equivalent instance, where the images of the morphisms are over the binary alphabet $\{a, b\}$. For this we define a coding morphism $\phi : \Gamma^\star \rightarrow \{a, b\}^\star$ by $\phi(a_i) = ab^{i+1}a$. The morphism ϕ is injective and it is rather clear that the instance $(\phi h, \phi g)$ has a solution if and only if the instance (h, g) has a solution.

Let $A = \{a, b\}$ and let $d = aba$. An instance (h, g) where $h, g : \Sigma^\star \rightarrow A^\star$ is called a *Claus instance* if $\Sigma = \{a_1, \ldots, a_n\}$ and for all $i = 2, \ldots, n-1$

$$h(a_i) \in (dA)^\star,\ h(a_1) \in (dA)^\star,\ h(a_n) = dd$$
$$g(a_i) \in (Ad)^\star,\ g(a_1) = d,\ g(a_n) \in (Ad)^+ d.$$

The idea of Claus instances is to add the factor $d = aba$ between every letter. Note that in the case of the instance $(\phi h, \phi g)$ the factor aba does not appear in the images. It is clear from the definition that all possible solutions of Claus instances must begin with the symbol a_1 and end with the symbol a_n. Therefore it is easy to see ([7]) that all of the non-empty solutions are of the form $\{a_1 w a_n\}^+$, where $w \in \{a_2, \ldots, a_{n-1}\}$ and $h(a_1 w a_n) = g(a_1 w a_n)$.

The name for Claus instances comes from a construction by V. Claus where from an n rule semi-Thue system a size $n + 4$ PCP instance is constructed such that,

the existence of a solution to the PCP instance is equivalent to the existence of a solution to the word problem of the semi-Thue system. We will outline the original construction by Claus here, as we will use similar ideas in later chapters.

Let $T = \{\Gamma, R\}$ be a semi-Thue system with $\Gamma = \{a, b\}$ and $R = \{t_1, \ldots, t_n\}$ where $t_i = (u_i, v_i)$. We may assume that the rules $t_i \in R$ are encoded into binary words by ϕ, so that $u_1, v_1 \in (ab^2b^\star a)^\star$. Recall that $d = aba$ and let $f = aa$ be a marker that is not an image of ϕ. We define two *desynchronizing morphisms* $l_d, r_d : \{a, b\}^\star \to \{a, b\}^\star$ by

$$l_d(x) = dx,$$
$$r_d(x) = xd$$

for $x \in \{a, b\}$. Now we add two new symbols c, e to our alphabet and define a PCP instance (h, g) with $h, g : (\{a, b, c, e\} \cup R)^\star \to \{a, b\}^\star$. Let $w_0 \to_T^* w$ be an instance of the word problem and set

	h	g	
x	$l_d(x)$	$r_d(x)$,	$x \in \{a, b\}$
t_i	$l_d(v_i)$	$r_d(u_i)$,	$t_i \in R$
c	$l_d(wf)$	d	
e	dd	$r_d(fw_0)d$	

Here the word $w \in \{a, b\}^\star$ is the input word and w_0 is the given fixed word. The instance described is by definition a Claus instance where $a_1 = c$ and $a_n = e$. It can be shown ([7]) that the possible minimal solutions to the instance (h, g) are of the form

$$cw_1fw_2f \cdots w_me$$

where

$$w_i = x_{i_0}t_{i_1}x_{i_1}t_{i_2}x_{i_2} \cdots t_{p_i}x_{p_i}$$

for some words $x_j \in \{a, b\}^\star$ and $t_j \in R$. Also, there is a derivation from w_i to w_{i+1} in the semi-Thue system T for $i = 1, \ldots, m - 1$. From the forms of the solutions to Claus instances we know that the minimal solutions here are of the form cwe, where the word w does not contain the symbols c or e. It follows that an instance of the PCP has a solution if and only if the instance of the word problem has a positive answer. Therefore, the undecidability of the PCP follows from the undecidability of the word problem for semi-Thue systems. The construction is such that for a semi-Thue system with n rules we get a PCP instance of size $n + 4$.

Matiyasevich and Sénizergues showed in [6] that there exists a 3-rule semi-Thue system with an undecidable individual word problem. From this result together with the above construction by Claus we get the following theorem:

***Theorem* 1.** *The PCP is undecidable for Claus instances of size 7.*

3 The Bi-infinite Post Correspondence Problem

In this chapter we study a variant of the PCP called the *bi-infinite Post Correspondence Problem* ($\mathbb{Z}$PCP), where it is asked whether or not there exists a bi-infinite sequence of the indices for a given instance of the PCP such that the words agree. As with most variants of the PCP it is beneficial to define the problem using morphisms. Recall the formal definition of the $\mathbb{Z}$PCP from Chapter 2:

Problem. *Given two morphisms $h, g\colon A^* \to B^*$, does there exist a bi-infinite word $\omega : \mathbb{Z} \to A$ such that $h(\omega) = g(\omega)$?*

As defined in Chapter 2 equality of the images of bi-infinite words is defined in the following way: $h(\omega) = g(\omega)$ if and only if there is a constant $k \in \mathbb{Z}$ such that $h(\omega)(i) = g(\omega)(i + k)$ for all positions $i \in \mathbb{Z}$.

An instance of the $\mathbb{Z}$PCP is therefore a pair of morphisms (h, g) and a bi-infinite word ω is a solution of the instance (h, g) if it satisfies $h(\omega) = g(\omega)$.

Recall that the infinite PCP asks for the existence of an infinite solution in one direction. In the one sided case the beginning of a solution is similar to the regular PCP: the solutions have initial pairs where one of the words is the prefix of the other. Adding a new domino never disrupts the matching of the previous ones. The chosen pair might not lead to a solution but it will not change the matching of previously added symbols.

The case of the $\mathbb{Z}$PCP is more sophisticated. In the $\mathbb{Z}$PCP the images of the letters need not be comparable because the matching is achieved by shifting one of the images by some constant k. Therefore to construct a solution we can take a pair of dominoes where the first word from one pair matches to the second word from the other pair and distance them by some constant k; see figure 2. The figure demonstrates how after every new letter is added to ω we have to check that a covering by images of letters under g exists. Moreover this word $a_1a_2a_3$ providing the cover must be a factor of ω. We see that there are more things to consider than a straightforward matching of letters.

The undecidability of the $\mathbb{Z}$PCP was originally proved by Ruohonen in [8] using linearly bounded automata (LBA). A word is accepted by an LBA if it is accepted by a Turing machine with a bounded length of tape, and the bound being linear to the length of the input. The family of languages accepted with LBA's coincides with

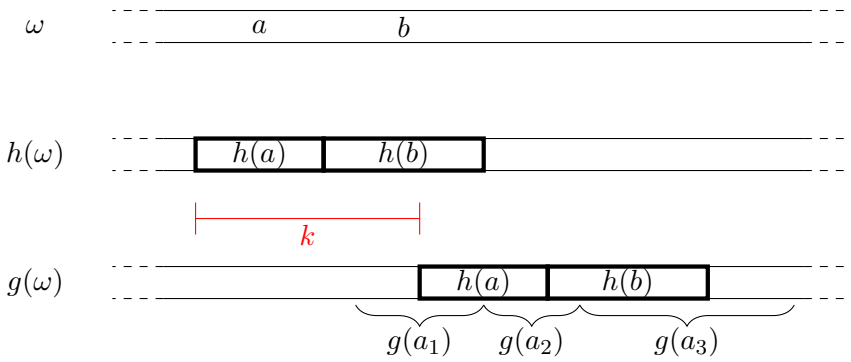

Figure 2. Constructing a pre-solution to the bi-infinite PCP.

context sensitive languages; see [2]. Ruohonen reduced the problem of existence of circular/repetitive configuration of a given LBA into the $\mathbb{Z}$PCP. In the circularity problem it is asked whether or not there exists a configuration α of the given LBA such that if the computation starts from α, after finitely many steps the LBA returns to the configuration α. Configuration here means the content of the tape and the state of the machine. As the input word of the LBA is not fixed, Ruohonen uses an extremely long word containing all reasonable configurations up to certain length determined by the linear bound in his construction.

In this chapter a simpler proof for undecidability of the $\mathbb{Z}$PCP is given. The proof presented is simpler because of the following two reasons:

1. it uses the word problem of the special type of semi-Thue systems in the reduction instead of the LBA, and

2. we are able to fix a word where the solution begins in the constructed instance of the $\mathbb{Z}$PCP.

We use a modification to Claus's construction from Chapter 2. Our modification uses the ideas of Halava and Harju in [21] where it was proved that the infinite PCP is undecidable for nine pairs of words. For the $\mathbb{Z}$PCP we also need the word problem of special type of semi-Thue systems, called deterministic semi-Thue systems.

Let $T = (\Sigma, R)$ be a semi-Thue system such that $\Sigma = A \cup B$ with $A \cap B = \emptyset$. Then T is called *B-deterministic*, if

1. $R \subseteq A^*BA^* \times A^*BA^*$, namely, if the rules contain a unique letter from B on both sides, and

2. for all words $w \in A^*BA^*$, if there exists a rule in R giving $w \to_T w'$, then the rule is unique in T.

Here we shall call a semi-Thue system deterministic, if it is B-deterministic for some B.

In [22] it was proved that the word problem is undecidable for deterministic semi-Thue systems. The construction there is rather involved as the constructed semi-Thue system is also *reversible* and the whole derivation starting from a fixed initial word is remembered in the words of the (unique) derivation. Moreover, it was proved in [22] that the *circular* word problem, asking whether or not there exists a derivation $u \to^+ u$ for a given u in T, is undecidable using the construction in [23] for deterministic semi-Thue systems. We shall now give a new simpler construction for the circular word problem (of the B-deterministic semi-Thue systems), using the halting problem of deterministic Turing machines. This construction is based on the one by Huet and Lankford ([24]) for undecidability of the word problem of the semi-Thue systems.

A Turing machine is a seven-tuple $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \star, F)$ where

- Q is a finite set of states.
- Γ is the tape alphabet.
- $\Sigma \subset \Gamma$ is the input alphabet.
- $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function.
- q_0 is the initial state.
- $\star \in \Gamma$ is the blank symbol.
- $F \subset Q$ is a set of final (accepting) states.

The Turing machine manipulates symbols on an infinite tape according to the rules given by δ. The machine reads the symbol on the tape at its current position and according to the transition function writes a symbol on that position, changes the state, and moves one position to the left or right of that position according to the letter L or R, respectively. A Turing machine is deterministic if for every state and symbol pair in δ there is at most one transition.

Let $\mathcal{M} = (Q, \Sigma, \Gamma, \delta, q_0, \star, F)$ be a deterministic Turing machine with blank symbol $\star$. Let the tape alphabet $\Gamma = \{b_0, b_1, \ldots, b_m\}$ where $b_0 = \star$ and let $Q = \{q_0, q_1, \ldots, q_n\}$. As usual, Q is the set of states, $\Sigma \subset \Gamma$ is the input alphabet, $\delta\colon Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function, q_0 is the initial state and F is the set of final states.

Our first semi-Thue system is $T'_{\mathcal{M}} = (B'_{\mathcal{M}}, \mathcal{R}'_{\mathcal{M}})$ for a Turing machine $\mathcal{M}$. Let

$$B'_{\mathcal{M}} = \{L, R\} \cup \Gamma \cup Q.$$

The symbols L and R are used as the left and right end markers of the configuration and are not to be confused with the left and right move directions of the Turing machine. Without loss of generality, we assume that the set of final states is $F = \{h\}$, that is, the TM has a unique halting state h.

The transitions of $\mathcal{M}$ are transformed into rules of the semi-Thue system $T'_{\mathcal{M}}$ in the following way:
1. For each right move (move with direction R) $(q_i, b_j) \longrightarrow (q_\ell, b_k, R)$ in $\mathcal{M}$, define

$$(q_i b_j,\ b_k q_\ell) \in \mathcal{R}'_{\mathcal{M}},$$

and if $j = 0$, i.e., $b_j = \star$, we also add the rule

$$(q_i R,\ b_k q_\ell R) \in \mathcal{R}'_{\mathcal{M}}.$$

2. For each left move $(q_i, b_j) \longrightarrow (q_\ell, b_k, L)$ in $\mathcal{M}$, we define the rules

$$\begin{aligned}&(b_t q_i b_j,\ q_\ell b_t b_k) \in \mathcal{R}'_{\mathcal{M}} \text{ for all } t = 0, 1, \ldots, m, \text{ and}\\ &(L q_i b_j,\ L q_\ell b_0 b_k) \in \mathcal{R}'_{\mathcal{M}},\end{aligned}$$

and if $j = 0$, i.e., $b_j = \star$, we also add the rules

$$\begin{aligned}&(b_t q_i R,\ q_\ell b_t b_k R) \in \mathcal{R}'_{\mathcal{M}} \text{ for all } t = 0, 1, \ldots, m, \text{ and}\\ &(L q_i R,\ L q_\ell b_0 b_k R) \in \mathcal{R}'_{\mathcal{M}}.\end{aligned}$$

It is rather straightforward to show that for a configuration $\alpha_1 q \alpha_2 \in \Gamma^* Q \Gamma^+$ of TM $\mathcal{M}$,

$$\alpha_1 q \alpha_2 \vdash_{\mathcal{M}} \alpha'_1 q' \alpha'_2 \iff L\alpha_1 q \alpha_2 R \to_{T'_{\mathcal{M}}} L\alpha'_1 q' \alpha'_2 R.$$

Moreover, $T'_{\mathcal{M}}$ is clearly Q-deterministic as $\mathcal{M}$ is deterministic.

As the halting problem for Turing machines is undecidable (even for the empty input word) we have

***Lemma* 3.** *The existence of derivations of the form $Lq_0 \star R \to^* LuhvR$ for some $u, v \in \Gamma^*$ is undecidable for Q-deterministic semi-Thue systems.*

Proof. If the existence of such a derivation would be decidable then we could also decide the halting problem for Turing machines on the empty input word, which is well known to be undecidable. □

Next we add the deletion rules to $T'_{\mathcal{M}}$ for the circular derivation, that is, we define two more "state" symbols h_l, h_r and add the following rules to $\mathcal{R}'_{\mathcal{M}}$ for all $a \in \Gamma$:

$$(ah, h_l),\ (ah_l, h_l),\ (Lh,\ Lh_r),\ (Lh_l,\ Lh_r), (h_r a,\ h_r).$$

The deletion rules have the effect that from every halting configuration there is a derivation

$$LuhvR \rightarrow^+ Lh_rR.$$

and by adding a rule

$$(Lh_rR, Lq_0\star R), \tag{1}$$

we have that

$$Lq_0\star R \rightarrow^+_{T'_{\mathcal{M}}} Lq_0\star R.$$

if and only if TM $\mathcal{M}$ halts on empty input. Note that adding these rules preserves the determinism of $T'_{\mathcal{M}}$

***Theorem* 2.** *The circular word problem is undecidable for the deterministic semi-Thue systems.*

Next we define our second semi-Thue system $T_{\mathcal{M}} = (B_{\mathcal{M}}, \mathcal{R}_{\mathcal{M}})$. Assume that $\mathcal{M}$ is a TM and $T'(\mathcal{M}) = (B'_{\mathcal{M}}, \mathcal{R}'_{\mathcal{M}})$ is a semi-Thue system defined as above. Let $B_{\mathcal{M}} = B'_{\mathcal{M}} \cup \{\overline{L}, \overline{R}\} \cup \overline{\Gamma} \cup \overline{Q}$, where $\overline{\Gamma}$ and $\overline{Q}$ are copies of the sets Γ and Q, respectively, where the elements are overlined. We define $\mathcal{R}_{\mathcal{M}}$ in the following steps. For our main theorem in the next section, we need to define rules for the overlined copies of the letters in the alphabet $B_{\mathcal{M}}$. We will make overlined copies of the rules in the following way:

1. We denote $S = \mathcal{R}'_{\mathcal{M}} \setminus \{(Lh_rR, Lq_0\star R)\}$. Clearly $S \subseteq \mathcal{R}_{\mathcal{M}}$.
2. For all the rules $(u, v) \in S$ defined above we also add the overlined rules $(\overline{u}, \overline{v}) \in \mathcal{R}_{\mathcal{M}}$.
3. Finally we add the rules

$$(Lh_rR, \overline{Lq_0\star R}) \text{ and } (\overline{Lh_rR}, Lq_0\star R)$$

to $\mathcal{R}_{\mathcal{M}}$ and denote these rules by t_h and $\overline{t_h}$, respectively. For $Lq_0\star R$ there exists a unique rule $(Lq_0\star R, v) \in \mathcal{R}_{\mathcal{M}}$ and we denote it by t_0.

It now follows that the TM $\mathcal{M}$ halts on empty input if and only if there is a derivation $Lq_0\star R \rightarrow^+ \overline{Lq_0\star R} \rightarrow^+ Lq_0\star R$ in $T_{\mathcal{M}}$. From the previous lemma we get

***Lemma* 4.** *The existence of derivations of the form $Lq_0\star R \rightarrow^+ \overline{Lq_0\star R} \rightarrow^+ Lq_0\star R$ is undecidable for Q-deterministic semi-Thue systems of the form of $T_{\mathcal{M}}$.*

3.1 The New Proof For Undecidability

In this section we construct an instance (h, g) of $\mathbb{Z}$PCP using the semi-Thue system $T_{\mathcal{M}}$ defined above. The idea behind the construction is that the morphisms simulate the rules of the semi-Thue system: if there is a single step derivation $xuy \rightarrow xvy$ for a rule $t = (u, v)$, then $h(xty)$ corresponds to xuy and $g(xty)$ corresponds to xvy (we

will actually not have these exact images, but desynchronized copies of them using special symbols). Then by forcing the initial configuration to appear in the solution we end up with a word containing all configurations reached by $T_{\mathcal{M}}$ when started on the initial configuration, as we shall see.

For the semi-Thue system $T_{\mathcal{M}} = (B_{\mathcal{M}}, \mathcal{R}_{\mathcal{M}})$ we can construct an equivalent system $\widehat{T_{\mathcal{M}}} = (\widehat{B_{\mathcal{M}}}, \widehat{\mathcal{R}_{\mathcal{M}}})$ that has an alphabet of size four (see [21]): when we denote $B_{\mathcal{M}} = \{a_1, a_2, \ldots, a_k, \overline{a_1}, \overline{a_2}, \ldots, \overline{a_k}\}$ and define a coding $\varphi : B_{\mathcal{M}}^* \to \{a, b, \overline{a}, \overline{b}\}^\star$ with $\varphi(a_i) = ab^i a$ and $\varphi(\overline{a_i}) = \overline{ab^i a}$ for $a_i \in B_{\mathcal{M}}$. Then the new set of rules is $\widehat{\mathcal{R}_{\mathcal{M}}} = \{(\varphi(u), \varphi(v)) \mid (u, v) \in \mathcal{R}_{\mathcal{M}}\}$. Now it is easy to see that the semi-Thue system $\widehat{T_{\mathcal{M}}} = (\widehat{B_{\mathcal{M}}}, \widehat{\mathcal{R}_{\mathcal{M}}})$ with $\widehat{B_{\mathcal{M}}} = \{a, b, \overline{a}, \overline{b}\}$ has an undecidable circular word problem if and only if $T_{\mathcal{M}}$ has. This follows from the fact that $w \to_{T_{\mathcal{M}}} w'$ if and only if $\varphi(w) \to_{\widehat{T_{\mathcal{M}}}} \varphi(w')$.

Note that the new system $\widehat{T_{\mathcal{M}}}$ is no longer Q-deterministic. However, we can extend the notion of B-determinism of semi-Thue systems from a set of symbols B into a set of words $\varphi(B)$ in a natural way. It follows that $\widehat{T_{\mathcal{M}}}$ is $\varphi(Q)$-deterministic.

Define the morphisms l_s and r_s, the left and right desynchronizing morphisms, respectively, for a word s by

$$l_s(a) = sa \text{ and } r_s(a) = as$$

for any letter a. In the following we use the notation a^{-k} for erasing k instances of letter a. For example $a^{-2}w$ is the word where the word aa is erased from the beginning of w assuming that w does have a prefix aa and wb^{-1} is the word where b is erased from the end of w assuming that w has the suffix b.

Let us now have the semi-Thue system $\widehat{T_{\mathcal{M}}}$ constructed from $T_{\mathcal{M}}$ via encoding φ. In what follows, the set of rules $\widehat{\mathcal{R}_{\mathcal{M}}}$ is considered also as an alphabet. From now on the rules t_i (and the special rules t_h, t_0) are from this set of rules and are not to be confused with the non-encoded rules $\mathcal{R}_{\mathcal{M}}$. From $T_{\mathcal{M}}$ we define the morphisms $h, g : (\{a_1, a_2, b_1, b_2, \overline{a_1}, \overline{a_2}, \overline{b_1}, \overline{b_2}, \#, \overline{\#}, \#_1, \#_2\} \cup \widehat{\mathcal{R}_{\mathcal{M}}})^* \to \{a, b, d, e, f, \#\}^*$ according to the following table:

	h	g	
x_1	dxd	$xee,$	$x \in \{a, b\}$
x_2	ddx	$xee,$	$x \in \{a, b\}$
t_i	$d^{-1}l_{d^2}(v_i)$	$r_{e^2}(u_i),$	$t_i \notin \{t_0, t_h\}$
t_0	$d^{-1}l_{d^2}(v_0)$	$r_{e^2}(u_0)e^{-2}f^3$	
t_h	$dr_{e^2}(v_h)e^{-2}ff$	$r_{e^2}(u_h)$	
$\#$	$dd\#d$	$\#ee$	
$\overline{x_1}$	xee	$xdd,$	$\overline{x} \in \{\overline{a}, \overline{b}\}$
$\overline{x_2}$	exe	$xdd,$	$\overline{x} \in \{\overline{a}, \overline{b}\}$
$\overline{t_i}$	$e^{-2}l_{e^2}(v_i)e$	$r_{d^2}(u_i),$	$\overline{t_i} \notin \{\overline{t_0}, \overline{t_h}\}$
$\overline{t_0}$	$e^{-2}l_{e^2}(v_0)e$	$r_{d^2}(u_0)d^{-2}f^3$	
$\overline{t_h}$	$r_{d^2}(v_h)d^{-2}f$	$r_{d^2}(u_h)$	
$\overline{\#}$	$e\#ee$	$\#dd$	
$\#_1$	$f\#ee$	$\#ee$	
$\#_2$	$ff\#d$	$\#dd$	

These morphisms are modifications of the morphisms introduced in [21] where the size of the domain alphabet is increased by adding overlined copies of all symbols. For the letters of $\widehat{B_{\mathcal{M}}}$ we have two copies of each letter indexed with 1 and 2. The purpose of this is to force exactly one re-writable term in configurations whose leftmost symbols are indexed with 1 and rightmost symbols are indexed with 2. In addition we add two new symbols $\#_1$ and $\#_2$ that aid in locating the transition between computation cycles. In the images we have three desynchronizing symbols d, e and f, where d and e are doing the main desynchronizing and f is added to aid the change in cycles. It is clear from the images of letters that if $h(\omega) = g(\omega)$ then ω must be bi-infinite. Using these morphisms we now state:

Lemma 5. *The semi-Thue system $\widehat{T_{\mathcal{M}}}$ has a cyclic computation if and only if there exists a bi-infinite word ω such that $h(\omega) = g(\omega)$.*

Proof. For clarity we omit the encoding φ from the following considerations and keep in mind that all of the symbols are actually encoded.

Assume that $\widehat{T_{\mathcal{M}}}$ has a cyclic computation. As the semi-Thue system is $\varphi(Q)$-deterministic we have the unique derivation

$$Lq_0\star R \to^+ \overline{Lq_0\star R} \to^+ Lq_0\star R.$$

We now code this computation into a bi-infinite word

$$\omega = (\alpha_0 t_0 \beta_0 \# \alpha_1 t_1 \beta_1 \# \cdots \# \alpha_h t_h \beta_h \#_1 \overline{\alpha_0 t_0 \beta_0 \# \alpha_1 t_1 \beta_1 \# \cdots \# \alpha_h t_h \beta_h} \#_2)^{\mathbb{Z}},$$

where $\alpha_i \in \{a_1, b_1\}^*, \beta_i \in \{a_2, b_2\}^*$ and $t_j \in \widehat{\mathcal{R}_{\mathcal{M}}}$ is the jth rule used in the derivation. We note that

$$dh(\alpha_i t_i \beta_i)dd = l_{d^2}(\alpha_i v_i \beta_i)dd = ddr_{d^2}(\alpha_{i+1} u_{i+1} \beta_{i+1}) = ddg(\overline{\alpha_{i+1} t_{i+1} \beta_{i+1}})$$

(similarly for overlined symbols), which means that the matching is acquired configuration to configuration from the consecutive configurations in the derivation that are overlined (resp. non-overlined). We set $\alpha_0 = \beta_0 = \alpha_h = \beta_h = \varepsilon$, the empty word. Also $t_h = (Lh_rR, \overline{Lq_0 \star R})$, $\overline{t_h} = (\overline{Lh_rR}, Lq_0 \star R)$ and $t_0 = (Lq_0 \star R, v)$ for some unique v as before.

From the construction of h and g we have that $h(\omega) = g(\omega)$, where the images are shifted by a constant k in relation to each other in order get a matching. Here the size of k depends on the length of the derivation as well as the lengths of the words in the derivation.

Assume then that we have a solution ν of the $\mathbb{Z}PCP$ instance (h, g).

As it is in the proof for the one-way infinite PCP, the images of the rule symbols t_i under h split the desynchronizing words (dd or ee) on either side of the image (on one side there is a single desynchronizing symbol and none on the other). Now from the form of g (either dd or ee appears between each pair of letters with the exception of the factor fff) it follows that the possible letters appearing before and after the rule symbol must be indexed with 1 and 2, respectively. This guarantees that between two $\#$-symbols there is exactly one re-writing symbol t_i as before and after a $\#$-symbol the letters (if any) must be indexed with 2 and 1, respectively.

The introduction of the new symbol e eliminates the trivial solutions that we would get with the morphisms in the proof of undecidability for one-way infinite PCP. It is also clear that we cannot have a solution using only overlined or non-overlined symbols because of the different desynchronizing symbols under h and g.

From the consideration above it follows that a solution must have symbols that are not overlined and symbols that are. From the forms of h and g follows that the only way to get from non-overlined symbols to overlined ones is via transition t_h, swapping d and e. Because $h(t_h)$ ends in fff and f appears in the image under g only in factor f^3, we have that t_h is followed by $\#_1$ in ν (and preceded by a letter that is not overlined and whose image ends in d). Now $h(t_h\#_1) = dr_{e^2}(v_h)e^{-2}f^3\#e^2$ and the image is not in conflict with the form of the images under g, in particular $h(t_h\#_1) = dg(t_0\#)$. Similarly, to swap from e to d we must have $\overline{t_h}\#_2$ in ν (preceded by an overlined letter whose image ends in e^2 as there are no letters whose image ends in d^2 under h) as $h(\overline{t_h}\#_2)d = r_{d^2}(v_h)d^{-2}f^3\#d^2 = g(\overline{t_0\#})$. This swap needs to happen infinitely often as after finitely many swaps the desynchronizing symbols to the right of the last swap are different in the images of h and g.

By these considerations we can deduce that the transition symbol t_h must appear in ν infinitely many times both as non-overlined and overlined versions. Because $h(t_h\#_1) = dg(t_0\#)$ and because u_0 appears desynchronized only in the image of $g(t_0)$ we have that t_0 must be in ν as many times as t_h. Similarly $\overline{t_0}$ must be in ν. Now by the desynchronizing property of h and g, between consecutive symbols t_0 and t_h in ν are only symbols t_i that are not overlined. Similarly between consecutive

symbols $\overline{t_0}$ and $\overline{t_h}$ are only overlined transition symbols.

The image $h(t_0)$ is a word $d^{-1}l_{d^2}(\alpha u'\beta)$ where by the $\varphi(Q)$-determinism of $\widehat{T_{\mathcal{M}}}$ there is a unique rule $t' = (u', v') \in \widehat{\mathcal{R}_{\mathcal{M}}}$. The only way to get a copy of $\alpha u'\beta$ desynchronized with d in $g(\nu)$ is to have $\overline{\alpha t'\beta}$ in ν. This in turn means that there is a word $\alpha' t''\beta'$ in ν such that $h(\overline{\alpha t'\beta}) = l_{e^2}(\alpha v'\beta)ee = eer_{e^2}(\alpha' u''\beta') = eeg(\alpha' t''\beta')$ for a unique rule $t'' = (u'', v'')$. The case is similar for $h(\overline{t_0})$ which shows that $\alpha t'\beta$ is in ν and so forth. Inductively we can see that ν contains all configurations and transitions done by $\widehat{T_{\mathcal{M}}}$ when started on the word $Lq_0 \star R$. It remains to be shown that the computation that begins with t_0 actually ends up using the rule t_h.

Because $h(\nu) = g(\nu)$, there is the smallest constant k such that shifting, say $g(\nu)$, k positions to the left, we have a matching. Now consider a factor u of ν containing a factor $t_0 \cdots t_h$ and all the symbols before and after this factor in ν having d in their image under h. Let us take u such that $|u|$ is as small as possible. The image $h(u)$ is desynchronized with d with the exception of the first and last symbols also containing the symbol e. To get a matching for the factor of u desynchronized with d we also must have a corresponding factor u' having the same factor desynchronized with d under g. Note that u and u' do not overlap in ν. If we assumed that the shifting of $g(\nu)$ is to the left, then u' is to the right of u. Because the image $g(\nu)$ is shifted to the left so is the factor $g(u)$. It follows that we have also u' to the left of u. Inductively we have that u and u' appear infinitely often in ν and are always the same distance from each other. The steps above are visualized in figure 3. Moreover the words separating u and u' must also be equal everywhere, that is we can write $\nu = (u\alpha u'\beta)^{\mathbb{Z}}$ for some words α and β. As the word $u\alpha u'\beta$ is finite it also contains finitely many configurations. Earlier we concluded that ν contains all configurations when started on input $Lq_0 \star R$. It follows from this and the determinism of $\widehat{T_{\mathcal{M}}}$ that after a finite number of derivation steps the rule t_h is used.

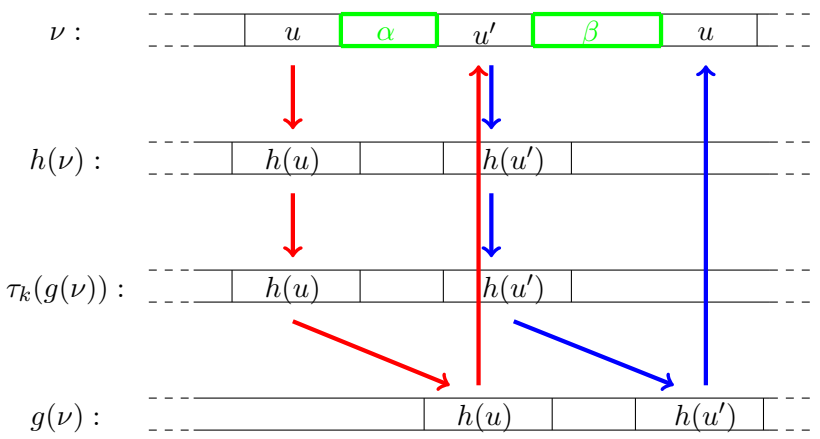

Figure 3. Visualization of how factors repeat in the solution. The arrows follow the steps described in the text.

We have concluded that there is a derivation $Lq_0 \star R \rightarrow \overline{Lq_0 \star R}$. A similar reasoning shows that continuing the derivations we will be using rule $\overline{t_h}$. It follows that there is a derivation $Lq_0 \star R \rightarrow^* \overline{Lq_0 \star R} \rightarrow^* Lq_0 \star R$ that is, $\widehat{T_{\mathcal{M}}}$ has a cyclic computation.

□

Combining the previous lemmas 4 and 5 with the fact that $\widehat{T_{\mathcal{M}}}$ has a cyclic computation if and only if $T_{\mathcal{M}}$ has, we have proved:

***Theorem* 3.** *The bi-infinite Post Correspondence Problem is undecidable.*

In the construction of the semi-Thue system $T_{\mathcal{M}}$ the addition of overlined letters was done to force a cyclic computation. The morphisms h and g were constructed in a way that the change between non-overlined and overlined letters is only possible in solutions containing the "description" of a cyclic computation in the semi-Thue system. In Chapter 4 this same idea is used in the context of another variant of the PCP.

4 The Conjugate Post Correspondence Problem

Two words x and y are conjugates if they can be written in the form $x = uv$ and $y = vu$ for some words u and v. Recall from Chapter 2 that the problem whether there exist conjugate words with the same image under a pair of morphisms is known as the circular Post Correspondence Problem, or CPCP for short. More formally the CPCP asks for given morphisms h and g, whether there exist words u and v with $uv \neq \varepsilon$ such that $h(uv) = g(vu)$. Here we give a new variant of the problem by requiring that the images are conjugate words with the same pre-image. We call this problem the *conjugate-PCP* and give it the following formal definition:

Problem (Conjugate Post Correspondence Problem)**.** *Given two morphisms $h, g : A^* \to B^*$, does there exist a word $w \in A^+$ such that $h(w) = uv$ and $g(w) = vu$ for some words $u, v \in B^*$?*

The conjugate-PCP was originally proved undecidable by Ruohonen in [8]. In this chapter we shall give a new simpler proof based on the semi-Thue system introduced in Chapter 3.

The behaviour of the instances of the conjugate-PCP differ vastly from the more traditional variants of the PCP where a valid presolution (prefix of a possible solution) can be verified by a matching of the images. Working out a possible solution to a conjugate-PCP instance is much less intuitive and more similar to the $\mathbb{Z}$PCP from Chapter 3.

For example, let us have some morphisms h, g and a guess that a solution w begins with the letter a. Then the situation is the following:

$$\begin{aligned} h(w) &= \underbrace{h(a) \quad \cdots \quad g(a) \quad \cdots}_{u \qquad\qquad v} \\ g(w) &= \underbrace{g(a) \quad \cdots \quad h(a) \quad \cdots}_{v \qquad\qquad u} \end{aligned}$$

The validity of the presolution to the conjugate-PCP beginning with a cannot be verified because no matching needs to happen between $h(a)$ and $g(a)$. Moreover the factorization of the images to u and v need not be unique even for minimal solutions:

Example. Let $h, g : \{a, b\}^* \to \{a, b\}^*$ be morphisms defined by

$$\begin{aligned} h(a) &= aba, \;\; g(a) = bab, \\ h(b) &= b, \quad\;\; g(b) = a. \end{aligned}$$

Now ab is a minimal solution for the conjugate-PCP instance (h, g) having two factorizations: $u = a, v = bab$ or $u = aba, v = b$.

The construction

We recall shortly the construction of the semi-Thue system $T_{\mathcal{M}}$ in Chapter 3. The construction used the structure of a given Turing machine $\mathcal{M}$. We can simplify our presentation by acknowledging the existence of such a system and declaring that our new system has the same properties.

Let $T = (\Sigma, \mathcal{R})$ be our semi-Thue system with the following properties:

1. $\Sigma = A \cup \overline{A} \cup B \cup \overline{B}$ with pairwise disjoint alphabets $A, \overline{A}, B, \overline{B}$. Notably $A = \{a, b, L, R\}$ where L, R are markers for the left and right border of the word, respectively.

2. T is $(B \cup \overline{B})$-deterministic in the following way:

   (i) $\mathcal{R} \subseteq (A^*BA^* \times A^*BA^*) \cup (\overline{A^*BA^*} \times \overline{A^*BA^*}) \cup (A^*BA^* \times \overline{A^*BA^*}) \cup (\overline{A^*BA^*} \times A^*BA^*)$.

   (ii) If there is a rule t_i in $\mathcal{R}$ where all symbols are non-overlined, then the corresponding overlined rule $\overline{t_i}$, where all symbols are overlined is also in $\mathcal{R}$, and vice versa.

   (iii) For all words $w \in (A \cup \overline{A})^*(B \cup \overline{B})(A \cup \overline{A})^*$, if there is a rule in $\mathcal{R}$ giving $w \to_T w'$ then the rule is unique.

   (iv) There is a single rule from $A^*BA^* \times \overline{A^*BA^*}$ and a single rule from $\overline{A^*BA^*} \times A^*BA^*$, moreover these rules are such that they re-write everything between the L and R markers, namely if there are rules giving $u \to_T \overline{w_0}$ and $\overline{u} \to_T w_0$ for a $u \in A^*BA^*$ then the rules are $(u, \overline{w_0})$ and $(\overline{u}, w_0)$, respectively.

3. T has an undecidable circular word problem. In particular it is undecidable whether T has a circular derivation $w_0 \to_T^* w_0$ where $w_0 \in A^*BA^*$ is the word appearing in rules of 2(iv). Note that w_0 and u in case 2(iv) are fixed words from the construction of the semi-Thue system T_M for a TM M, and $w_0 \neq u$.

The special $(B \cup \overline{B})$-determinism of T can be interpreted as derivations being in two different phases: the normal phase and the overlined phase. Transitioning between phases is via the unique rules from 2(iv). It is straightforward to see that all derivations do not go through phase changes and that the phase is changed more than once if and only if T has a circular derivation. The system considered is now clear from the context and we write the derivations omitting the index T simply as $\to$.

We now add a few additional rules to T: we remove the unique rule $(u, \overline{w_0})$ and replace it with one extra step by introducing rules (u, s) and $(s, \overline{w_0})$ where s is a new symbol for the intermediate step. The corresponding overlined rules $(\overline{u}, \overline{s})$ and $(\overline{s}, w_0)$ are added also to replace the rule $(\overline{u}, w_0)$. These new rules are needed in identifying the border between words u and v, and adding them has no effect on the behaviour of T.

By Theorem 2 we have the following lemma.

Lemma 6. *Assume that the semi-Thue system T is constructed as above. Then T has an undecidable individual circular word problem for the word w_0.*

We now reduce the individual circular word problem of the system T to the conjugate-PCP.

Let $\mathcal{R} = \{t_0, t_1, \ldots, t_{h-1}, t_h\}$, where the rules are pairs $t_i = (u_i, v_i)$. We denote by l_x and r_x the left and right desynchronizing morphisms defined by

$$l_x(a) = xa, \qquad r_x(a) = ax$$

for all words x. In the following we consider the elements of $\mathcal{R}$ as letters. Denote by A_j the alphabet A where letters are given subscripts $j = 1, 2$. Define morphisms $h, g : (A_1 \cup A_2 \cup \overline{A_1} \cup \overline{A_2} \cup \{\#, \overline{\#}, I\} \cup \mathcal{R})^* \to \{a, b, d, e, f, \#, \$, £\}^*$ according to the following table:

	h	g	
I	$\$l_{d^2}(w_0\#)d$	$£ee,$	
x_1	dxd	$xee,$	$x \in \{a, b\}$
x_2	ddx	$xee,$	$x \in \{a, b\}$
t_i	$d^{-1}l_{d^2}(v_i)$	$r_{e^2}(u_i),$	$t_i \notin \{t_{h-1}, t_h\}$
t_{h-1}	$dsff$	$r_{e^2}(u\#)$	
t_h	$f\$£l_{e^2}(w_0\#)ee$	$sfff£\$dd$	
$\#$	$dd\#d$	$\#ee$	
$\overline{x_1}$	xee	$xdd,$	$\overline{x} \in \{\overline{a}, \overline{b}\}$
$\overline{x_2}$	exe	$xdd,$	$\overline{x} \in \{\overline{a}, \overline{b}\}$
$\overline{t_i}$	$e^{-2}l_{e^2}(v_i)e$	$r_{d^2}(u_i),$	$\overline{t_i} \notin \{\overline{t_{h-1}}, \overline{t_h}\}$
$\overline{t_{h-1}}$	sf	$r_{d^2}(u\#)$	
$\overline{t_h}$	$ff£$	$sfff\$$	
$\overline{\#}$	$e\#ee$	$\#dd$	

Here the re-writing rules are of the form $t_i = (u_i, v_i)$, for some u_i, v_i. The following rules play important roles:
$t_{h-1} = (u, s)$, where u is the unique word such that $(u, \overline{w_0}) \in R$, and
$t_h = (s, \overline{w_0})$.

We begin by examining the form of the images of h and g. The morphisms are modifications of the ones in Chapter 3 with slight alterations made such that it is possible to have (finite) solutions to the conjugate-PCP instance with easily identifiable borders between the factors u and v using special symbols \$ and £. The symbols d, e and f function as desynchronizing symbols. The desynchronizing symbols d and e make sure that in the solution w the factors that will represent the configurations of the semi-Thue system T are of the correct form, that is of the form where the determinism is kept intact. This follows from the forms of h and g: under g all images are desynchronized by either e^2 (non-overlined letters) or d^2 (overlined letters). To get similarly desynchronized factors in the image under h we note that in the pre-image the words between two $\#$-symbols (similarly for overlined symbols $\overline{\#}$) are of the form $\alpha t \beta$ where $\alpha \in \{a_1, b_1\}$, $\beta \in \{a_2, b_2\}$ and $t \in \mathcal{R}$ (with end markers L and R omitted from α and β). The symbol f is not really used in desynchronizing but making sure that the change between phases is carried out correctly.

The following lemma is useful in our proof:

***Lemma* 7.** *The images $h(w)$ and $g(w)$ are conjugates if and only if $h(w_1)$ and $g(w_2)$ are conjugates for all conjugates w_1 and w_2 of w.*

Proof. If $h(w_1)$ and $g(w_2)$ are conjugates for all conjugates w_1 and w_2 of w then $h(w)$ and $g(w)$ are conjugates.

Assume then that $h(w)$ and $g(w)$ are conjugates and let w_1 and w_2 be conjugates of w. We may now write $w_1 = xwx^{-1}$ and $w_2 = ywy^{-1}$. Denote $wx^{-1} = w'$ and $wy^{-1} = w''$. Now $h(w_1) = h(xw') = h(x)h(w')$ is a conjugate of $h(w')h(x) = h(w'x) = h(w)$ and $g(w_2) = g(yw'') = g(y)g(w'')$ is a conjugate of $g(w'')g(y) = g(w''y) = g(w)$. By our assumption also $h(w_1)$ and $g(w_2)$ are conjugates. □

Next we will show that a circular derivation beginning from word w_0 exists in T if and only if there is a solution to the conjugate-PCP instance (h, g). We will prove the claim in the following two lemmata.

***Lemma* 8.** *If there is a circular derivation in T beginning from w_0, then there exists a non-empty word w such that $h(w) = uv$ and $g(w) = vu$ for some words u and v.*

Proof. Assume that a circular derivation exists. The derivation is of the form $w_0 = \alpha_1 u_1 \beta_1 \to \alpha_1 v_1 \beta_1 = \alpha_2 u_2 \beta_2 \to \cdots \to u \to s \to \overline{w_0} = \overline{\alpha_1 u_1 \beta_1} \to \cdots \overline{u} \to \overline{s} \to w_0$. This derivation can be coded into a word

$$w = Iw_1\#w_2\#w_3\#\cdots\#t_{h-1}t_h\overline{w_1\#w_2\#w_3\#\cdots\#t_{h-1}t_h},$$

where $w_i = \alpha_i t_i \beta_i$ for each i and where we recall that $t_i = (u_i, v_i)$ is the unique rewriting rule used in each derivation step. The rules t_{h-1} and t_h appear right before the transition to overlined derivation as they correspond to the final and intermediate steps before the transition. Let us consider the images of w under the morphisms h and g:

$$h(w) = \$l_{d^2}(w_0\#\alpha_1 v_1\beta_1\#\alpha_2 v_2\beta_2\#\cdots\#s)fff\$\pounds l_{e^2}(w_0\#\alpha_1 v_1\beta_1\cdots\#s)fff\pounds)$$

and

$$g(w) = r_{e^2}(\pounds\alpha_1 u_1\beta_1\#\alpha_2 u_2\beta_2\#\cdots\#u\#)sfff\pounds r_{d^2}(\$\alpha_1 u_1\beta_1\cdots\#u\#)sfff\$).$$

These images are indeed very similar. The images match at all positions that do not contain a desynchronizing symbol (d or e) or a special symbol (\$ or £). Thus, if we erased all of these non-matching symbols we would have images that are equal (and of the form q^2 for a word q). Also the non-matching symbols are such that d is always matched with e and \$ is always matched with £. It is quite clear that the factors in both $h(w)$ and $g(w)$ beginning and ending in the same special symbol are the same, that is, the factors $u = \$\cdots\$$ and $v = \pounds\cdots\pounds$ appearing in both images are equal. It follows that $h(w) = uv$ and $g(w) = vu$, which proves our claim. □

***Lemma* 9.** *If there exists a non-empty word w such that $h(w) = uv$ and $g(w) = vu$ for some words u and v, then there is a circular derivation in T beginning from w_0.*

Proof. Firstly we show that the factor f^3 must appear in $h(w)$ and hence $t_{h-1}t_h$ or $\overline{t_{h-1}t_h}$ is a factor in w. Assume the contrary: there is no factor f^3 in $h(w)$.

From the construction of g we know that also $h(w)$ is desynchronized so that between each letter there is either a factor d^2 or e^2. Conjugation of $g(w)$ does not break this property except possibly for the beginning and the end of $h(w)$. Note $h(w)$ could start and end in a single desynchronizing symbol.

Take now the first letter c of w. We can assume that it is a non-overlined letter as the considerations are similar for the overlined case. The letter c cannot be t_{h-1} as it would have to be followed by t_h: f^2 does not appear as a factor under g without f^3, and $t_{h-1}\overline{t_h}$ produces f^4, which is uncoverable by g. From the construction of h we see that the letters following c must also be non-overlined, otherwise the desynchronization would be broken. Thus the desynchronizing symbol is the same for all these following letters. But as we can see from the form of the morphisms h and g, we have a different desynchronizing symbols under g for c and its successors. It is clear that $h(g)$ must contain both d and e and so w must have both non-overlined and overlined letters. If there is a change in the desynchronizing symbol in $h(w)$ then it

contradicts the form of the images under g. Hence we must have the factor $t_{h-1}t_h$ in w to make the transition without breaking the desynchronization.

The images of the factor $t_{h-1}t_h$ are as follows:

$$h(t_{h-1}t_h) = dsfff\$£l_{e^2}(w_0\#)ee$$

and

$$g(t_{h-1}t_h) = r_{e^2}(u\#)sfff\$£dd.$$

As we can see the desynchronizing symbols do not match. Hence we also must have the overlined copy of this factor in w, that is, the factor $\overline{t_{h-1}t_h}I$, the images of which are as follows (the letter I is a forced continuation to the overlined factor to account for the special symbols \$ and £):

$$h(\overline{t_{h-1}t_h}I) = sfff£\$l_{d^2}(w_0\#)d$$

and

$$g(\overline{t_{h-1}t_h}I) = r_{d^2}(u\#)sfff\$£ee.$$

Either one of these factors has one swap between symbol d and e. From the above we concluded that we need an even number of these swaps as for every factor $t_{h-1}t_h$ we must also have the factor $\overline{t_{h-1}t_h}I$ and vice versa. It is possible that $h(w)$ ends in the letter f. In this case the swap happens "from the end to the beginning", i.e., the prefix of a factor doing the swap is at the end of w and the remaining suffix is at the beginning of w. The following proposition shows that we can in fact restrict ourselves to the case where the factors $t_{h-1}t_h$ and $\overline{t_{h-1}t_h}$ are intact, that is, the swap does not happen from the end to the beginning of $h(w)$ as a result of the conjugation between $h(w)$ and $g(w)$.

Proposition. *It may be assumed that the first and last symbols of $h(w)$ are \$ and £.*

Proof of proposition. If $h(w)$ is not of the desired form then it has £\$ as a factor (by above the symbols from $\overline{t_{h-1}t_h}I$ are in w). Images of the letters under h do not have £\$ as a factor so there is a factorization $w = w_1w_2$ such that $h(w_1)$ ends in £ and $h(w_2)$ begins with \$ (w_1 ends in $\overline{t_h}$ and w_2 begins with I). By Lemma 7, $h(w)$ and $g(w)$ are conjugates if and only if $h(w_2w_1)$ and $g(w_2w_1)$ are, where now $h(w_2w_1)$ has \$ as the first symbol and £ as the last symbol.

□

Now by the above proposition we may assume that w begins with I and ends with $\overline{t_h}$. From this it also follows that when $h(w) = uv$ and $g(w) = vu$ the word u has \$ as the first and the last symbol and v has £ as the first and the last symbol.

It follows that $w = I \cdots t_h \cdots \overline{t_h}$, where the border between u and v is in the image $h(t_h)$:

$$\begin{aligned} h(w) &= \underbrace{\$l_{d^2}(w_0\#)d \quad \cdots \quad f\$\pounds l_{e^2}(w_0\#)ee}_{u} \quad \cdots \quad \underbrace{ff\pounds}_{v} \\ g(w) &= \underbrace{\pounds ee \quad \cdots \quad sfff\pounds\$dd}_{v} \quad \cdots \quad \underbrace{sfff\$}_{u} \end{aligned}$$

Here the border between u and v need not be in the image of the same instance of t_h. Nevertheless we know by above that in the image under g the word u begins with $\$l_{d^2}(w_0\#)d$. To get this image as a factor of $g(w)$ we must have $t_h\overline{\alpha_1 t_1 \beta_1 \#}$ in w, where $t_1 = (u_1, v_1)$ is the first rewriting rule used and $w_0 = \alpha_1 u_1 \beta_1$. Now

$$h(t_h\overline{\alpha_1 t_1 \beta_1 \#}) = f\$\pounds l_{e^2}(w_0\#\alpha_1 v_1 \beta_1 \#)ee$$

which shows that

$$I\alpha_1 t_1 \beta_1 \# \alpha_2 t_2 \beta_2 \# \in w \tag{2}$$

where by the $(B \cup \overline{B})$-determinism of T rule $t_2 \in \mathcal{R}$ is the unique rule and $\alpha_1, \alpha_2 \in \{a_1, b_1\}^* \cup \{\varepsilon\}$ and $\beta_1, \beta_2 \in \{a_2, b_2\}^* \cup \{\varepsilon\}$ are unique words such that $g(\alpha_2 t_2 \beta_2) = r_{e^2}(\alpha_2 u_2 \beta_2) = r_{e^2}(\alpha_1 v_1 \beta_1)$. Again,

$$h(I\alpha_1 t_1 \beta_1 \# \alpha_2 t_2 \beta_2 \#) = \$l_{d^2}(w_0\#\alpha_1 v_1 \beta_1 \# \alpha_2 v_2 \beta_2 \#)d$$

which as also a factor of $g(w)$ gives that

$$t_h\overline{\alpha_1 t_1 \beta_1 \# \alpha_2 t_2 \beta_2 \# \alpha_3 t_3 \beta_3 \#} \in w \tag{3}$$

for a unique $t_3 \in \mathcal{R}$ and $\alpha_3 \in \{a_1, b_1\}^* \cup \{\varepsilon\}$, $\beta_3 \in \{a_2, b_2\}^* \cup \{\varepsilon\}$.

It is easy to see that the words given by this procedure beginning with I or t_h (as in 2 and 3, respectively) contain derivations of the system T starting from w_0 where configurations are represented as words between #-symbols and consecutive configurations in these words are also consecutive in T (as is explained in the beginning of the proof), that is, we get from the former to the latter by a single derivation step.

From the finiteness of w it follows that long enough factors of w with the forms as in 2 and 3 represent cyclic computations: the configuration/symbol s is reached eventually and from there we have the rule (s, w_0) which starts a new cycle. We conclude that T must have a cyclic computation starting from configuration w_0.

□

Lemmas 6, 8 and 9 together yield our main theorem:

***Theorem* 4.** *The conjugate-PCP is undecidable.*

This result does not generalize using this same construction by say, adding more desynchronizing symbols and border markers for each element in the permutation. The generalization of the conjugate-PCP would be the following problem:

Problem (Image Permutation Post Correspondence Problem)**.** *Given two morphisms $h, g : A^* \to B^*$, does there exist a word $w \in A^+$ and an n-permutation σ such that $h(w) = u_1 u_2 \cdots u_n$ and $g(w) = u_{\sigma(1)} u_{\sigma(2)} \cdots u_{\sigma(n)}$ for some words $u_1, \ldots, u_n \in B^*$?*

The reason the construction does not work for the general n-permutations is that allowing more factors to be permuted can make solutions that do not describe TM computations. This is because of special cases for different values of n and σ, but also by the fact that the permutated factors may be single letters. In fact any solution w that produces Abelian equivalent words $h(w)$ and $g(w)$ also has a permutation that makes one of the words into the other. A "simple" proof using the techniques in this chapter is for now deemed unlikely, and some other approach may prove to be more fruitful.

As a related result we note that PCP instances where one of the morphisms is a permutation of the other are undecidable. Indeed, it was shown by Halava and Harju in [25] that the PCP is undecidable for instances $(h, h\pi)$, where $h : A^* \to B^*$ is a morphism and $\pi : A^* \to A^*$ is a permutation.

5 A Problem In Word Shuffling

In formal language theory the study on the shuffling operator goes back at least to the 1960s; see Ginsburg and Spanier [26]. The shuffle operation appears in different types of applicational problems like concurrency of processes ([27; 28]) and multi-point communication ([29]), to mention a few. The applicability to these problems led to a lot of theoretical research on shuffling in the 70's and 80's (see for example [30; 31]). Since then, the operation has occurred in many contexts including formal languages, computability, and process algebras; see e.g. [32; 33]. Therefore, it is fair to say that shuffling is an important topic in formal language and automata theory; see [34].

Let A be an alphabet. The *shuffle* operation $\sqcup\!\sqcup$ on words over A is defined recursively as follows: for words $u, v \in A^*$ and letters $a, b \in A$,

$$au \sqcup\!\sqcup bv = a(u \sqcup\!\sqcup bv) \cup b(au \sqcup\!\sqcup v)\,,$$
$$\varepsilon \sqcup\!\sqcup u = u \sqcup\!\sqcup \varepsilon = \{u\}.$$

Note that a shuffle of two words is always a set of words. For languages $L_1, L_2 \in A^*$, the shuffle is defined by

$$L_1 \sqcup\!\sqcup L_2 = \bigcup_{u \in L_1, v \in L_2} u \sqcup\!\sqcup v\,.$$

We give the following well-known results as propositions, the proofs can be found in [35].

***Proposition* 1.** *For two regular languages L_1 and L_2, also $L_1 \sqcup\!\sqcup L_2$ is regular.*

***Proposition* 2.** *The family of the context-free languages is not closed under the shuffle operation, but $L \sqcup\!\sqcup R$ is context-free for context-free L and regular R.*

***Proposition* 3.** *It is undecidable for context-free languages L whether there are context-free languages L_1 and L_2 with $L_1, L_2 \neq \{\varepsilon\}$ such that $L = L_1 \sqcup\!\sqcup L_2$.*

The shuffle of equal length words u and v is called *perfect*, denoted by $\sqcup\!\sqcup_p$, if the letters of the two words alternate starting from the first letter of u (in the literature also the term *interleaving* is used to refer to the perfect shuffle, and is denoted by $I(u, v)$). Formally, if $u = a_1 a_2 \cdots a_n$ and $v = b_1 b_2 \cdots b_n$ then $u \sqcup\!\sqcup_p v =$

$a_1b_1a_2b_2\cdots a_nb_n$. The perfect shuffle can also be defined for words that are not of equal length by simply catenating the remaining suffix of the longer word as the suffix of the shuffle word when all the letters of the shorter word have been exhausted. Note also that the prefect shuffle of two words is a single word. Indeed $u \sqcup\!\sqcup_p v \in u \sqcup\!\sqcup v$. In our considerations the equal length definition is sufficient as we are only shuffling words of equal length.

In Henshall et al. [36] the authors considered the following four self-shuffles: $w \sqcup\!\sqcup w$, $w \sqcup\!\sqcup w^R$, $w \sqcup\!\sqcup_p w$ and $w \sqcup\!\sqcup_p w^R$. In fact these notions were extended in [36] to languages rather than single words. In this chapter we consider a special type of shuffle which can be thought of as a type of a self-shuffle where we look at words that are shuffled with their own images under a letter-to-letter morphism ϕ. In particular, we are interested in the existence of such words in the context of regular languages: given a regular language R and a morphism ϕ can we find a word w so that R contains a word in $w \sqcup\!\sqcup \phi(w)$?

We prove that, for given a regular language R and a letter-to-letter morphism ϕ, it is undecidable whether or not there exists a word w such that $w \sqcup\!\sqcup \phi(w) \cap R \neq \emptyset$. This result was already presented by Engelfriet and Rozenberg in [37], where the authors showed that it is undecidable whether or not $M \cap L_{\{0,1\}} = \emptyset$, where M is an arbitrary regular language and $L_{\{0,1\}} = \bigcup_{w\in\{0,1\}^\star} w \sqcup\!\sqcup \overline{w}$. We present a more direct proof of this result in this chapter. In [37] overlined letters serve the same purpose as the substitution ϕ here. In the rest of the chapter overlined symbols are not to be confused with the ones in the problem formulation from [37].

5.1 Perfect shuffle

In the case of the perfect shuffle, the above problem is rather straightforward. Indeed, for a given word w, the language $w \sqcup\!\sqcup_p \phi(w)$ is a singleton, and the language

$$L = \bigcup_{w\in A^*} w \sqcup\!\sqcup_p \phi(w)$$

is regular. To see this, let h be the morphism defined by $h(a) = a\phi(a)$ for all $a \in A$. For a word $w = a_1a_2\cdots a_n$ we have

$$h(w) = h(a_1a_2\cdots a_n) = a_1\phi(a_1)a_2\phi(a_2)\cdots a_n\phi(a_n).$$

We then have $L = h(A^*)$. It follows that $L \cap R$ is regular for all regular R, and we can effectively check the emptiness of $L \cap R$.

The case of perfect shuffle with the image of the reversed word (or image of the word reversed, as $\phi(w^R) = \phi(w)^R$ for letter to letter morphisms) can also be seen to be decidable by modifying the results in [36]. It was shown there that the language $L' = \bigcup_{w\in L} w \sqcup\!\sqcup_p w^R$ is context-free for all regular languages L. In a similar fashion

it can be shown that also the language $L'' = \bigcup_{w \in A^*} w \sqcup_p \phi(w^R)$ is context-free; see the proof in [36]. From Lemma 1 it follows that $L'' \cap R$ is context-free for all regular languages R, and by Lemma 2 the emptiness is decidable for these languages.

Therefore, we have

Theorem 5. *Given a regular language R and a letter-to-letter morphism ϕ, the following problems are decidable:*

1. *does there exist a word w such that $w \sqcup_p \phi(w) \cap R \neq \emptyset$.*
2. *does there exist a word w such that $w \sqcup_p \phi(w^R) \cap R \neq \emptyset$.*

5.2 Regular shuffle

As opposed to the perfect shuffle, regular shuffle has a lot more freedom in terms of the lengths of the factors alternating between the words. This freedom is indeed enough to make our problem undecidable.

In the proof of Theorem 6 we reduce our shuffle problem to the Post Correspondence Problem.

Theorem 6. *Given a regular language R and a letter-to-letter morphism ϕ, it is undecidable whether or not there exists a word w such that $w \sqcup \phi(w) \cap R \neq \emptyset$.*

Proof. Let (h_1, h_2) be an instance of the Post Correspondence Problem (PCP) with $h_1, h_2 : A^* \to A^*$.

We build a new alphabet using the letters from A. For each letter $a \in A$, we will have two colours: colour $\underline{a}$ and colour $\overline{a}$. Now for each letter $b \in A$, we will take copies of that letter coloured with $\underline{a}$ and $\overline{a}$ for all $a \in A$ denoted by $b^{\underline{a}}$ and $b^{\overline{a}}$, respectively. We denote the alphabets of these coloured letters, coloured by $\underline{a}$ and $\overline{a}$, respectively, by $A_{\underline{a}}$ and $A_{\overline{a}}$. The sets of underlined and overlined colours are $\underline{\mathcal{C}} = \{\underline{a} \mid a \in A\}$ and $\overline{\mathcal{C}} = \{\overline{a} \mid a \in A\}$, respectively. Let also

$$A_{\underline{\mathcal{C}}} = \bigcup_{a \in A} A_{\underline{a}} \text{ and } A_{\overline{\mathcal{C}}} = \bigcup_{a \in A} A_{\overline{a}}$$

denote the full sets of underlined and overlined coloured letters, respectively. Let $e\colon (A_{\underline{\mathcal{C}}} \cup A_{\overline{\mathcal{C}}})^* \to A^*$ be the *eraser morphism* that deletes the colours from the coloured letters returning them as letters in A. For a word $w \in A^*$, we denote by $w^{(\underline{a})}$ the word w with the colouring $\underline{a}$ of each of its letters so that, $e(w^{(\underline{a})}) = w$. If a word w is coloured with (possibly) multiple colours from $\underline{\mathcal{C}}$ without specifying the colouring we write it as $w^{(\underline{\mathcal{C}})}$, which is equivalent to $w \in A^*_{\underline{\mathcal{C}}}$ (and similarly for $\overline{\mathcal{C}}$).

Now, define $R = L_1 \cap L_2$, where

$$L_1 = \{uv \mid \exists a \in A\colon h_1(a) = e(u), u \in A^*_{\underline{\mathcal{C}}},\ v \in A^*_{\overline{a}}\}^+ \tag{4}$$

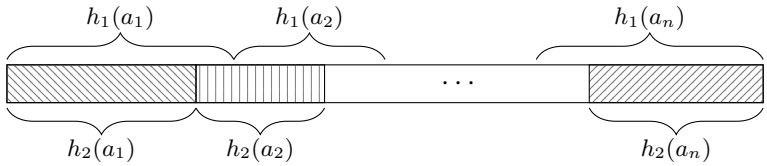

Figure 4. Colouring of *e(w)*.

and

$$L_2 = \left(A^*_{\underline{\mathcal{C}}} \cdot \{w \mid \exists a \in A\colon h_2(a) = e(w),\ w \in A^*_{\overline{a}}\}\right)^+. \tag{5}$$

The words in L_1 are of the form $\omega = u_1 v_1 \cdots u_k v_k$, where each u_i is an image of a single letter from A under h_1, say b_i, with any colouring of its letters from the set $\underline{\mathcal{C}}$, and v_i is a word coloured with $\overline{b_i}$ where $h_1(b_i) = e(u)$.

In L_2 the words are of the form $\sigma = s_1 t_1 \cdots s_l t_l$ where the factors s_i have any colouring from the set $\underline{\mathcal{C}}$, and t_i are images of single letters from A coloured with overlined colours corresponding to their pre-images.

It is clear that the language $R = L_1 \cap L_2$ is regular as an intersection of two regular languages.

Let then $\phi : A_{\underline{\mathcal{C}}} \to A_{\overline{\mathcal{C}}}$ be a letter-to-letter substitution that recolours the underlined words: for all $a, b \in A$, let

$$\phi(b^{(\underline{a})}) = b^{(\overline{a})}.$$

***Claim* 1.** *There exists a word $w \in A_{\underline{\mathcal{C}}}$ such that $w \sqcup\!\sqcup \phi(w) \cap R \neq \emptyset$ if and only if there exists a word $v = a_1 a_2 \cdots a_n$ such that $e(w) = h_1(v)$ and $h_1(v) = h_2(v)$.*

Proof. Assume first that such a word v exists and thus

$$e(w) = h_1(v) = h_1(a_1)h_1(a_2)\cdots h_1(a_n).$$

Language R is over $A_{\underline{\mathcal{C}}} \cup A_{\overline{\mathcal{C}}}$ which means that w has a colouring, which is defined in the following way:

Take $e(w) = h_1(v)$. By the assumptions $e(w)$ also has a h_2-cover. Colour each letter of w with the corresponding colour of the letter that covers it under h_2 (see Figure 4 for the coloring under morphism h_2). For example, if an occurrence of the letter a is covered by $h_2(a_k)$ then a will be of colour $\underline{a_k}$. Therefore with this colouring, we have

$$w = h_1(a_1)^{(\underline{\mathcal{C}})} h_1(a_2)^{(\underline{\mathcal{C}})} \cdots h_1(a_n)^{(\underline{\mathcal{C}})} = h_2(a_1)^{(\underline{a_1})} h_2(a_2)^{(\underline{a_2})} \cdots h_2(a_n)^{(\underline{a_n})}$$

and

$$\phi(w) = h_2(a_1)^{(\overline{a_1})} h_2(a_2)^{(\overline{a_2})} \cdots h_2(a_n)^{(\overline{a_n})}.$$

It follows that the word

$$h_1(a_1)^{(\underline{\mathcal{C}})}h_2(a_1)^{(\overline{a_1})}h_1(a_2)^{(\underline{\mathcal{C}})}h_2(a_2)^{(\overline{a_2})}\cdots h_1(a_n)^{(\underline{\mathcal{C}})}h_2(a_n)^{(\overline{a_n})}$$

belongs to both $w \sqcup\!\sqcup \phi(w)$ and R, which proves the claim from right to left.

Let us prove the other direction. Assume that there exists a word w such that $w \sqcup\!\sqcup \phi(w) \cap R$ is not empty, and choose a word $\tau \in w \sqcup\!\sqcup \phi(w) \cap R$. From the construction of $R = L_1 \cap L_2$ it follows that $\tau = u_1^{(\underline{\mathcal{C}})}v_1^{(\overline{\mathcal{C}})}u_2^{(\underline{\mathcal{C}})}v_2^{(\overline{\mathcal{C}})}\cdots u_k^{(\underline{\mathcal{C}})}v_k^{(\overline{\mathcal{C}})}$, for some u_i, v_i and $k > 0$. Moreover for each u_i with $i \in \{1, \ldots, k\}$ the word $e(u_i)$ is an image of a letter from A under the morphism h_1. Denote by a_i the letter corresponding to u_i, that is, $e(u_i) = h_1(a_i)$.

Since $\tau \in L_1$ each v_j is coloured with the overlined colour corresponding to a_j determined by u_j. Also, since $\tau \in L_2$, for each v_j we actually have that $e(v_j) = h_2(a_j)$. It follows that

$$\tau = h_1(a_1)^{(\underline{\mathcal{C}})}h_2(a_1)^{(\overline{a_1})}h_1(a_2)^{(\underline{\mathcal{C}})}h_2(a_2)^{(\overline{a_2})}\cdots h_1(a_k)^{(\underline{\mathcal{C}})}h_2(a_k)^{(\overline{a_k})}$$

for some letters $a_1, a_2, \ldots, a_k \in A$.

On the other hand $\tau \in w \sqcup\!\sqcup \phi(w)$. Here ϕ is length preserving with domain $A_{\underline{\mathcal{C}}}$ and range $A_{\overline{\mathcal{C}}}$. This implies that

$$w = h_1(a_1)^{(\underline{\mathcal{C}})}h_1(a_2)^{(\underline{\mathcal{C}})}\cdots h_1(a_k)^{(\underline{\mathcal{C}})}$$

and

$$\phi(w) = h_2(a_1)^{(\overline{a_1})}h_2(a_2)^{(\overline{a_2})}\cdots h_2(a_k)^{(\overline{a_k})}.$$

All that ϕ does is change the colouring of the letters. Hence in terms of the underlying alphabet A the words do not change. This leads to

$$e(w) = e(\phi(w))$$

which implies

$$h_1(a_1a_2\cdots a_k) = h_2(a_1a_2\cdots a_k)$$

and proves Claim 1. □

Now, by Claim 1, a word w exists if and only if the instance (h_1, h_2) of the PCP has a solution. From this it follows that the existence of a word w is also undecidable. □

The above is a direct proof for our result using a reduction to the PCP. In [37] the result is a corollary of a number of other results on the representation of recursively enumerable languages. It is debatable whether or not the proof in this chapter is more simple. Nevertheless the reduction to the PCP is an interesting one.

The last remaining case for shuffling a word with its letter-to-letter substitution is the following:

Problem. *For a regular language R and a morphism ϕ, does there exist a word w such that $w \sqcup\!\sqcup \phi(w^R) \cap R \neq \emptyset$?*

This case remains open for now. We conjecture that it is also an undecidable problem.

6 The Fixed Point Problem For Injective Rational Functions

The problem of interest in this chapter is the *fixed point problem* of functions. For functions on finitely generated word semigroups, the problem is formulated in the following way:

Problem (The fixed point problem for word semigroups)**.** *Let A be a finite alphabet and $f\colon A^* \to A^*$ be a function. Does there exist a word w such that $f(w) = w$.*

We shall prove that the fixed point problem is undecidable for injective rational functions. Rational functions are functions that are realized by rational transductions, i.e., functions that are defined by finite transducers (defined formally in Section 6.1 below). As a corollary we also acquire a result for computable (recursive) functions over natural numbers. Our proof applies the undecidability of the injective PCP introduced in Chapter 2.

For general functions the undecidability of the existence of a fixed point follows easily from the basic computability results on Turing machines. We can see this by giving a modification of the classical diagonal argument or by the halting problem, i.e., by defining a function f in a way that it checks for any input i whether the ith Turing machine halts on its code, we can transform f into a function h having a fixed point i if and only if the ith Turing machine halts on input i. In this construction the function h is clearly noncomputable. In this chapter we construct a function which is computable and has undecidable fixed point problem.

In automata theory the fixed point problem has been studied for cellular automata; see [38; 39]. There it is shown that it is undecidable whether a cellular automaton converges to a pattern from a given finite configuration or from any configuration. In particular, it is undecidable whether there is a configuration that is fixed by the given cellular automaton.

6.1 Rational Transductions

A finite transducer is in essence a finite nondeterministic automaton with an output. Formally a transducer is a 6-tuple

$$\mathcal{T} = (Q, A, B, E, q, F)$$

where

- Q is the set of states with an initial state $q \in Q$ and the set of final states $F \subset Q$,
- A and B are the input and output alphabets, respectively,
- E is a finite relation of transitions $E \subset Q \times A^* \times B^* \times Q$.

We also write $q \xrightarrow{u,v} p$ for $(q, u, v, p) \in E$, where $u \in A^*$ is the input word and $v \in B^*$ is the output word. Finite transducers can be presented (similarly to finite automata) as directed graphs with edges labelled with ordered pairs (u, v) stating the input and output of the transition between states.

A computation of a transducer is a sequence of transitions

$$\alpha : p_1 \xrightarrow{u_1,v_1} p_2 \xrightarrow{u_2,v_2} \cdots \xrightarrow{u_n,v_n} p_{n+1},$$

where $u_1u_2 \cdots u_n$ is called the input word and $v_1v_2 \cdots v_n$ is called the output word. We can simply write $p \xrightarrow{u,v} q$ if there exists a computation α from p to q with input u and output v. A computation is accepting if it begins at the initial state and ends in a final state.

The relation *realized* by a transducer $\mathcal{T}$ is

$$\tau = \{(u, v) \mid \text{there is an accepting computation in } \mathcal{T} \text{ with input } u \text{ and output } v\}.$$

A relation realized by a finite transducer is called a *rational transduction*. A rational transduction $\tau \subset A^* \times B^*$ may also be considered as a function $\tau : A^* \to 2^{B^*}$, where $\tau(u) = \{v \mid (u, v) \in \tau\}$. The *domain* of τ is defined to be the set $\mathrm{dom}(\tau) = \{w \mid \tau(w) \neq \emptyset\}$.

In the rest of the chapter we are mostly interested in *rational (partial) functions*. These are rational transductions that are functions $\tau\colon \mathrm{dom}(\tau) \to B^*$, i.e., $\tau(w)$ is single valued for all $w \in \mathrm{dom}(\tau)$. A rational partial function τ is a *rational function* if $\mathrm{dom}(\tau) = A^*$.

The family of the rational transductions is closed under compositions; see e.g. Berstel [40]. For closure properties of rational transductions we give the following theorem:

Theorem 7. *Assume τ_1 and τ_2 are rational transductions. Then*

1. *the union $\tau_1 \cup \tau_2$, and*
2. *the composition $\tau_1\tau_2 = \{(u, v) \mid \exists w\colon (u, w) \in \tau_1$ and $(w, v) \in \tau_2\}$ are rational transductions.*

The following two theorems give us characterizations of the rational transductions. Our first characterization (Theorem 8; see [41]) by Nivat is given using a regular language, intersection, morphism and inverse morphism. The second characterization (Theorem 9; see [42]) by Latteux and Leguy gives a purely morphic characterization of the rational transductions except for the marker function.

***Theorem* 8** (Nivat)**.** *A mapping $\tau\colon A^* \to 2^{B^*}$ is a rational transduction if and only if there exist a regular set R and two morphisms g and h such that*

$$\tau(w) = g(h^{-1}(w) \cap R)\,.$$

***Theorem* 9** (Latteux-Leguy)**.** *A mapping $\tau\colon A^* \to 2^{B^*}$ is a rational transduction if and only if there exist morphisms h_i and a marking μ (with $\mu(w) = wa$ for some fixed letter a) such that*

$$\tau = h_4 h_3^{-1} h_2 h_1^{-1} \mu\,.$$

There are many results on the form of the compositions for various different kinds of transductions. It was shown by Turakainen [43] that the compositions of morphisms and inverse morphisms *without* the markings, are exactly the rational transductions realized by simple transducers, i.e., transducers that have a unique final state that is also the (unique) initial state.

***Theorem* 10.** *The family of rational transductions realized by simple transducers is equal to the set of all compositions*

$$\tau = h_4 h_3^{-1} h_2 h_1^{-1},$$

where h_i is a morphism for all $i = 1, 2, 3, 4$.

6.2 The Fixed Point Problem

We shall employ the undecidability of the injective PCP introduced in Chapter 2 and have it at work in our next theorem.

***Theorem* 11.** *The PCP is undecidable for injective morphisms.*

Let $A_1 = \{1, 2, \ldots, k-1\}$ for some $k \in \mathbb{N}$ and choose $A = A_1 \cup \{0\}$. Assume now that $g, h\colon A_1^* \to B^*$ are injective morphisms.

Since the regular languages are closed under taking morphic images, inverse morphic images and taking complements, the languages

$$R = g^{-1}h(A_1^+) \text{ and } \overline{R} = A^+ \setminus g^{-1}h(A_1^+)$$

are both regular. Let $\oplus$ and $\ominus$ be new symbols. Then also the marked language $\oplus R \cup \ominus \overline{R}$ is regular.

The mapping

$$\chi_{g^{-1}h}(w) = \begin{cases} \oplus w, & \text{if } w \in R, \\ \ominus w, & \text{if } w \in \overline{R}\,. \end{cases}$$

is rational. Indeed, we can apply Theorem 8 to the morphisms g', h' and the regular language $\oplus R \cup \ominus\overline{R}$, where g' and h' are identity functions except that $h'(\oplus) = \varepsilon = h'(\ominus)$.

Next we shall define transductions $\mu_{h^{-1}g}$ from $\oplus R$ to A^+ and μ_C from $\ominus\overline{R}$ to A^+. We begin with $\mu_{h^{-1}g}$.

A word w belongs to $R = g^{-1}h(A_1^+)$ if and only if $w = g^{-1}h(u)$ for $u = h^{-1}g(w)$. (Recall that g and h are injective.) The image $\mu_{h^{-1}g}(w)$ for $w \in \oplus R$ is then defined by

$$\mu_{h^{-1}g}(\oplus w) = u, \quad \text{if } g(w) = h(u).$$

Obviously, this mapping is a rational function as g and h are injective.

The transduction μ_C from $\ominus\overline{R}$ is defined by

$$\mu_C(\ominus w) = w01.$$

Clearly, μ_C is an injective rational function. Now also $\mu_{h^{-1}g} \cup \mu_C$ is a rational function from $\{\oplus, \ominus\}A^+$ to A^+, and it is injective.

Finally, consider the composition $\tau_{(g,h)} = (\mu_{h^{-1}g} \cup \mu_C) \circ \chi_{g^{-1}h}$. By Theorem 7 it is a rational transduction. By the above constructions,

$$\tau_{(g,h)}(w) = \begin{cases} h^{-1}g(w), & \text{if } w \in R, \\ w01, & \text{if } w \in \overline{R}. \end{cases}$$

***Lemma* 10.** *The mapping $\tau_{(g,h)}$ is an injective rational function from A^+ to A^+.*

Proof. We need to show that $\tau_{(g,h)}$ is injective, that is, if $\tau_{(g,h)}(w_1) = \tau_{(g,h)}(w_2)$, then $w_1 = w_2$. There are three cases:

(1) If $w_1, w_2 \in R$ $(= g^{-1}h(A_1^+))$, then $\tau_{(g,h)}(w_1) = h^{-1}g(w_1) = h^{-1}g(w_2) = \tau_{(g,h)}(w_2)$ if and only if $g(w_1) = g(w_2)$ by injectivity of h. By injectivity of g this is possible if and only if $w_1 = w_2$.

(2) For $w_1 \in R$ and $w_2 \in \overline{R}$, we have $\tau_{(g,h)}(w_1) \neq \tau_{(g,h)}(w_2)$ as $\tau_{(g,h)}(w_2)$ contains symbol 0.

(3) For $w_1, w_2 \in \overline{R}$, $\tau_{(g,h)}(w_1) = w_101 = w_201 = \tau_{(g,h)}(w_2)$ if and only if $w_1 = w_2$.

This proves the claim. □

***Lemma* 11.** *Let $\tau_{(g,h)}$ be as in the above. Then $\tau_{(g,h)}(w) = w$ if and only if w is a solution of the instance (g, h) of the injective PCP.*

Proof. It follows by the above construction that $\tau_{(g,h)}(w) = w$ if and only if $w \in g^{-1}h(A_1^+)$ and $h^{-1}g(w) = w$, in other words, if and only if $w \in A_1^+$ and $g(w) = h(w)$. □

Therefore, we have proved our main theorem.

***Theorem* 12.** *The fixed point problem is is undecidable for injective rational functions.*

6.3 Fixed points in computable functions over $\mathbb{N}$

Next we transform our result on fixed point of rational transduction into a result on computable functions over the natural numbers. For that we shall use the injective mapping $\sigma\colon A^* \to \mathbb{N}$ where a word $w \in A$ is considered as a k-adic number. More formally, for a word $w = a_n \cdots a_0 \in A^*$, with each a_i in $A = \{0, 1, \ldots, k-1\}$,

$$\sigma(w) = \sigma(a_n a_{n-1} \cdots a_0) = \sum_{i=0}^{n} a_i k^i.$$

It is clear that σ is injective. It is also clear that the functions σ, σ^{-1} and $\tau_{(g,h)}$ (as constructed in the previous section) are computable (by Turing machines or by any equivalent model of computability). As a corollary to Theorem 12 we get the following result by considering the functions $f = \sigma\tau_{(g,h)}\sigma^{-1}\colon \mathbb{N} \to \mathbb{N}$ for injective morphisms g and h.

***Theorem* 13.** *The fixed point problem is undecidable for injective computable functions over* $\mathbb{N}$.

Indeed, as the class of the Turing computable functions is equal to the class of recursive functions, we have proved that the fixed point problem is undecidable for injective recursive functions over $\mathbb{N}$.

List of References

[1] PCP at home. URL https://web.archive.org/web/20090309162234/http://www.theory.informatik.uni-kassel.de/~stamer/pcp/pcpcontest_en.html.

[2] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Co., Reading, Mass., 1979. ISBN 0-201-02988-X.

[3] A. Ehrenfeucht, J. Karhumäki, and G. Rozenberg. The (generalized) Post correspondence problem with lists consisting of two words is decidable. *Theoret. Comput. Sci.*, 21(2):119–144, 1982. ISSN 0304-3975. doi: 10.1016/0304-3975(89)90080-7. URL https://doi.org/10.1016/0304-3975(89)90080-7.

[4] Vesa Halava, Tero Harju, and Juhani Karhumäki. Decidability of the binary infinite Post correspondence problem. *Discrete Appl. Math.*, 130(3):521–526, 2003. ISSN 0166-218X. doi: 10.1016/S0166-218X(03)00330-5. URL https://doi.org/10.1016/S0166-218X(03)00330-5.

[5] Turlough Neary. Undecidability in binary tag systems and the Post correspondence problem for five pairs of words. In *32nd International Symposium on Theoretical Aspects of Computer Science*, volume 30 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 649–661. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2015.

[6] Yuri Matiyasevich and Géraud Sénizergues. Decision problems for semi-Thue systems with a few rules. *Theoret. Comput. Sci.*, 330(1):145–169, 2005. ISSN 0304-3975. doi: 10.1016/j.tcs.2004.09.016. URL http://dx.doi.org/10.1016/j.tcs.2004.09.016.

[7] V. Claus. Some remarks on PCP(k) and related problems. *Bull. EATCS*, 12:54–61, 1980.

[8] Keijo Ruohonen. On some variants of Post's correspondence problem. *Acta Inform.*, 19(4):357–367, 1983. ISSN 0001-5903. doi: 10.1007/BF00290732. URL http://dx.doi.org/10.1007/BF00290732.

[9] Jing Dong and Qinghui Liu. Undecidability of infinite post correspondence problem for instances of size 8. *RAIRO Theor. Inform. Appl.*, 46(3):451–457, 2012. ISSN 0988-3754. doi: 10.1051/ita/2012015. URL https://doi.org/10.1051/ita/2012015.

[10] Keijo Ruohonen. Reversible machines and Post's correspondence problem for biprefix morphisms. *Elektron. Informationsverarb. Kybernet.*, 21(12):579–595, 1985. ISSN 0013-5712.

[11] Arto Salomaa. *Formal languages*. ACM Monograph Series. Academic Press [Harcourt Brace Jovanovich, Publishers], New York-London, 1973.

[12] Michael A. Harrison. *Introduction to formal language theory*. Addison-Wesley Publishing Co., Reading, Mass., 1978. ISBN 0-201-02955-3.

[13] M Lothaire. *Combinatorics on Words*. Cambridge University Press, Cambridge, 1997. ISBN 9780511566097.

[14] M. Lothaire. *Algebraic combinatorics on words*, volume 90 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2002. ISBN 0-521-81220-8. doi: 10.1017/CBO9781107326019. URL https://doi.org/10.1017/CBO9781107326019. A collective work by Jean Berstel, Dominique Perrin, Patrice Seebold, Julien Cassaigne, Aldo De Luca, Steffano Varricchio, Alain Lascoux, Bernard Leclerc, Jean-Yves Thibon, Veronique

Bruyere, Christiane Frougny, Filippo Mignosi, Antonio Restivo, Christophe Reutenauer, Dominique Foata, Guo-Niu Han, Jacques Desarmenien, Volker Diekert, Tero Harju, Juhani Karhumaki and Wojciech Plandowski, With a preface by Berstel and Perrin.

[15] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages. Vol. 1*. Springer-Verlag, Berlin, 1997. ISBN 3-540-60420-0. doi: 10.1007/978-3-642-59126-6. URL https://doi.org/10.1007/978-3-642-59126-6. Word, language, grammar.

[16] Y. Lecerf. Récursive insolubilité de l'équation générale de diagonalisation de deux monomorphismes de monoïdes libres $\phi x = \psi x$. *Comptes Rendus de l'Académie des Sciences*, 257:2940–2943, 1963.

[17] Juhani Karhumäki and Aleksi Saarela. Noneffective regularity of equality languages and bounded delay morphisms. *Discrete Math. Theor. Comput. Sci.*, 12(4):9–17, 2010.

[18] Mari Ernvall, Vesa Halava, and Tero Harju. On the n-permutation post correspondence problem. *Theoretical Computer Science*, 601:15–20, oct 2015. doi: 10.1016/j.tcs.2015.07.021.

[19] Emil Leon Post. Recursive unsolvability of a problem of Thue. *J. Symbolic Logic*, 12:1–11, 1947. ISSN 0022-4812. doi: 10.2307/2267170. URL http://dx.doi.org/10.2307/2267170.

[20] A. Markoff. On the impossibility of certain algorithms in the theory of associative systems. *C. R. (Doklady) Acad. Sci. URSS (N.S.)*, 55:583–586, 1947.

[21] Vesa Halava and Tero Harju. Undecidability of infinite Post correspondence problem for instances of size 9. *Theor. Inform. Appl.*, 40(4):551–557, 2006. ISSN 0988-3754. doi: 10.1051/ita:2006039. URL https://doi.org/10.1051/ita:2006039.

[22] Vesa Halava and Tero Harju. New proof for the undecidability of the circular PCP. *Acta Inform.*, 50(5-6):331–341, 2013. ISSN 0001-5903. doi: 10.1007/s00236-013-0183-5. URL http://dx.doi.org/10.1007/s00236-013-0183-5.

[23] Vesa Halava and Tero Harju. Word problem for deterministic and reversible semi-Thue systems. *Semigroup Forum*, 88(2):468–478, 2014. ISSN 0037-1912. doi: 10.1007/s00233-013-9550-3. URL http://dx.doi.org/10.1007/s00233-013-9550-3.

[24] G. Huet and D. Lankford. On the uniform halting problem for term rewriting systems. *Rapport Laboria 283, INRIA*, 1978. URL http://www.ens-lyon.fr/LIP/REWRITING/TERMINATION/Huet_Lankford.pdf.

[25] Vesa Halava and T. Harju. Some new results on post correspondence problem and its modifications. *Bull. EATCS*, 73:131–141, 2001.

[26] Seymour Ginsburg and Edwin H. Spanier. Mappings of languages by two-tape devices. *Journal of the ACM*, 12(3):423–434, jul 1965. doi: 10.1145/321281.321294.

[27] M. Nivat A. Arnold. *Collo. AFCET Les Mathématiques del'informatique*, chapter Comportements de processus, pages 35–68. 1982.

[28] W. F. Ogden, W. E. Riddle, and W. C. Round. Complexity of expressions allowing concurrency. In *Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages - POPL '78*. ACM Press, 1978. doi: 10.1145/512760.512780.

[29] Kazuo Iwama. Unique decomposability of shuffled strings. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing - STOC '83*. ACM Press, 1983. doi: 10.1145/800061.808768.

[30] Kazuo Iwama. The universe problem for unrestricted flow languages. *Acta Informatica*, 19(1), apr 1983. doi: 10.1007/bf00263929.

[31] Matthias Jantzen. The power of synchronizing operations on strings. *Theoretical Computer Science*, 14(2):127–154, 1981. doi: 10.1016/0304-3975(81)90054-2.

[32] Jean Berstel, Luc Boasson, Olivier Carton, Jean-Éric Pin, and Antonio Restivo. The expressive power of the shuffle product. *Information and Computation*, 208(11):1258–1272, nov 2010. doi: 10.1016/j.ic.2010.06.002.

[33] Masami Ito. *Algebraic Theory of Automata and Languages*. WORLD SCIENTIFIC, apr 2004. doi: 10.1142/4791.

[34] M. Beek and J. Kleijn. Shuffles and synchronized shuffles: A survey. In *Discrete Mathematics and Computer Science*, 2014.

[35] C. Câmpeanu, K. Salomaa, and S. Vágvölgyi. Shuffle decompositions of regular languages. *Internat. J. Found. Comput. Sci.*, 13(6):799–816, 2002. ISSN 0129-0541. doi: 10.1142/S0129054102001461. URL https://doi.org/10.1142/S0129054102001461.

[36] D. Henshall, N. Rampersad, and J. Shallit. Shuffling and unshuffling. *Bulletin of the EATCS*, (107):131–142, 2012.

[37] J. Engelfriet and G. Rozenberg. Fixed point languages, equality languages, and representation of recursively enumerable languages. *Journal of the ACM*, 27(3):499–518, jul 1980. doi: 10.1145/322203.322211.

[38] Karel Culik II, Jan Pachl, and Sheng Yu. On the limit sets of cellular automata. *SIAM Journal on Computing*, 18(4):831–842, aug 1989. doi: 10.1137/0218057.

[39] Jarkko Kari. The nilpotency problem of one-dimensional cellular automata. *SIAM Journal on Computing*, 21(3):571–586, jun 1992. doi: 10.1137/0221036.

[40] Jean Berstel. *Transductions and Context-Free Languages*. Vieweg+Teubner Verlag, 1979. doi: 10.1007/978-3-663-09367-1.

[41] Maurice Nivat. Transductions des langages de chomsky. *Annales de l'institut Fourier*, 18(1):339–455, 1968. doi: 10.5802/aif.287.

[42] M. Latteux and J. Leguy. On the composition of morphisms and inverse morphisms. In *Automata, languages and programming (Barcelona, 1983)*, volume 154 of *Lecture Notes in Comput. Sci.*, pages 420–432. Springer, Berlin, 1983. doi: 10.1007/BFb0036926. URL https://doi.org/10.1007/BFb0036926.

[43] P. Turakainen. A machine-oriented approach to compositions of morphisms and inverse morphisms. *Bulletin of the EATCS*, (20):162–166, 1983.

TURUN
YLIOPISTO
UNIVERSITY
OF TURKU

Painosalama, Turku, Finland 2022

ISBN 978-951-29-8787-0 (PRINT)
ISBN 978-951-29-8788-7 (PDF)
ISSN 0082-7002 (PRINT)
ISSN 2343-3175 (ONLINE))