TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik

Technische Berichte

Technical Reports

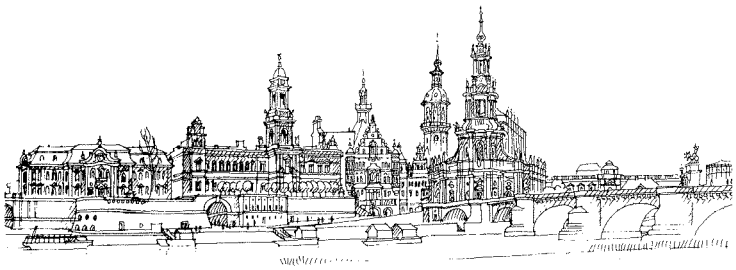
ISSN 1430-211X

TUD / FI 98 / 07 - Mai 1998

Daniel Kirsten

Grundlagen der Programmierung
Institut für Softwaretechnik I

Some Undecidability Results related to the Star Problem in Trace Monoids



Technische Universität Dresden
Fakultät Informatik
D-01062 Dresden
Germany

URL: http://www.inf.tu-dresden.de/

Some Undecidability Results related to the Star Problem in Trace Monoids*

Daniel Kirsten
Department of Computer Science
Dresden University of Technology
D-01062 Dresden, Germany

Daniel.Kirsten@inf.tu-dresden.de
http://www.inf.tu-dresden.de/~dk11

May 29, 1998

Abstract

This paper deals with decision problems related to the star problem in trace monoids, which means to determine whether the iteration of a recognizable trace language is recognizable. Due to a theorem by Richomme from 1994 [30, 31], we know that the Star Problem is decidable in trace monoids which do not contain a C4-submonoid. The C4 is (isomorphic to) the Cartesian Product of two free monoids over doubleton alphabets. It is not known, whether the Star Problem is decidable in C4 or in trace monoids containing a C4. In this paper, we show undecidability of some related problems: Assume a trace monoid which contains a C4. Then, it is undecidable whether for two given recognizable languages K and L, we have $K \subseteq L^*$, although we can decide $K^* \subseteq L$. Further, we can not decide recognizability of $K \cap L^*$ as well as universality and recognizability of $K \cup L^*$.

*This work has been supported by the postgraduate program "Specification of discrete processes and systems of processes by operational models and logics" of the German Research Community (Deutsche Forschungsgemeinschaft).

1 Introduction

Free partially commutative monoids, also called trace monoids, were introduced by CARTIER and FOATA in 1969 [4]. In 1977, MAZURKIEWICZ proposed trace monoids as a potential model for concurrent processes [21], which marks the beginning of a systematic study of trace monoids by mathematicians and theoretical computer scientists, see e.g., [7, 8, 9]. A part of the research in trace theory deals with examinations of well-known classic results for free monoids in the framework of traces.

One main stream in trace theory is the study of recognizable trace languages, which can be considered as an extension of the well studied concept of regular languages in free monoids. A major step in this research is OCHMAŃSKY's PhD thesis from 1984 [28]. Some of the results concerning regular languages in free monoids can be generalized to recognizable languages in trace monoids. However, there is one major difference: The iteration of a recognizable trace language does not necessarily yield a recognizable language. This fact raises the so called star problem: Given a recognizable language L, is L^* recognizable? In general, it is not known whether the star problem is decidable. The main result after a stream of publications dealing with this problem is a theorem stated by RICHOMME in 1994, saying that the star problem is decidable in trace monoids which do not contain a particular submonoid called C4 [30, 31]. It is not known whether the star problem is decidable in trace monoids with a C4-submonoid. It is even unknown for finite trace languages.

In this paper, we consider some decision problems for recognizable trace languages which are related to the star problem. If we have two recognizable languages K and L in a trace monoid with a C4-submonoid, then it is undecidable whether K is a subset of L^* and whether $K \cup L^*$ yields the complete monoid. Further, recognizability of $K \cup L^*$ and $K \cap L^*$ is undecidable.

The paper is organized as follows. After this introduction, I explain some concepts from algebra and formal language theory. In the first subsection, we get familiar with some basic notions from algebra, formal language theory and trace theory. In the subsequent ones, we deal with recognizable sets, rational sets and relations between them. Then, we discuss some decision problems concerning recognizable and rational trace languages and their solutions as far as known.

In Section 3, we establish a method to define two recognizable trace languages from a given instance of POST's Correspondence Proplem. We examine properties of these languages and properties of the iteration of one of these languages. In Section 4, we use these properties to develop the main results. In Section 5 we show some additional results which may become important in future papers. In Section 6, we compare the new results to known results and discuss their possible relations to the star problem.

2 Formal Definitions

2.1 Monoids, Languages and Traces

I briefly introduce the basic notions from algebra and trace theory. Unless I do not state precise sources, I consider the concepts and notions as well-known.

By $\mathbb{N}$, we denote the set of natural numbers including zero, i.e., $\mathbb{N} = \{0, 1, 2, \ldots\}$.

A *monoid* $(\mathbb{M}, \oplus)$ is an algebraic structure consisting of a set $\mathbb{M}$, and a binary operation $\oplus$ fulfilling the following two axioms: Firstly, the operation $\oplus$ is associative, i.e., for every elements k, l, m in $\mathbb{M}$, we have $(k \oplus l) \oplus m = k \oplus (l \oplus m)$. Secondly, $\mathbb{M}$ contains an element $\lambda_{\mathbb{M}}$, such that for every m in $\mathbb{M}$, we have $\lambda_{\mathbb{M}} \oplus m = m \oplus \lambda_{\mathbb{M}} = m$. $\mathbb{M}$ is called the *underlying set*, the operation $\oplus$ is called the *monoid operation* or *monoid product*, and the element $\lambda_{\mathbb{M}}$ is called the *neutral element*. We drop the symbol $\oplus$ and denote the monoid product by juxtaposition. We drop the index at the neutral element $\lambda_{\mathbb{M}}$ as long as no confusion arises. We use the symbol $\mathbb{M}$ to denote both the monoid and its underlying set. A monoid is called *finite* iff $\mathbb{M}$ is a finite set.

For every natural number n, we define the n-fold monoid product as follows: For every m in $\mathbb{M}$, m^0 yields the neutral element $\lambda_{\mathbb{M}}$, and further, for every natural number n, m^{n+1} denotes $m^n m$.

We extend the monoid product to subsets of $\mathbb{M}$. If K and L are two subsets of $\mathbb{M}$, the set KL contains all elements kl for some k in K and l in L. We extend the n-fold monoid product to sets. We define $L^0 := \{\lambda_{\mathbb{M}}\}$, and for every natural number n, $L^{n+1} := L^n L$. We call a subset L of $\mathbb{M}$ *closed under monoid product* iff LL is a subset of L.

For a subset L of a monoid $\mathbb{M}$, we define the non-empty iteration L^+ as the union of the sets L^1, L^2, L^3,... Hence, L^+ is the least subset of $\mathbb{M}$ which contains L and is closed under monoid product. We denote the iteration of L by L^* and define it by $L^* := L^+ \cup \{\lambda_{\mathbb{M}}\}$.

We call a subset of G of $\mathbb{M}$ a set of *generators* of $\mathbb{M}$ iff firstly, it holds $G^* = \mathbb{M}$, and secondly, there is no proper subset K of G with $K^* = \mathbb{M}$. We call a monoid $\mathbb{M}$ *finitely generated* iff $\mathbb{M}$ has a finite set of generators.

Assume two monoids $\mathbb{M}$ and $\mathbb{M}'$. Their Cartesian Product is the monoid denoted by $\binom{\mathbb{M}}{\mathbb{M}'}$ and defined in the following way: The underlying set is the Cartesian Product of the underlying sets of $\mathbb{M}$ and $\mathbb{M}'$. The monoid operation is defined componentwise, i.e., for every pair of elements $\binom{m_1}{m_1'}$ and $\binom{m_2}{m_2'}$, their product yields $\binom{m_1 m_2}{m_1' m_2'}$. The products $m_1 m_2$ and $m_1' m_2'$ are the monoid products in $\mathbb{M}$ and $\mathbb{M}'$, respectively. The neutral element of the Cartesian Product of $\mathbb{M}$ and $\mathbb{M}'$ is $\binom{\lambda_{\mathbb{M}}}{\lambda_{\mathbb{M}'}}$.

Again, assume two monoids $\mathbb{M}$ and $\mathbb{M}'$. We call a function $h : \mathbb{M} \to \mathbb{M}'$ a *homomorphism* iff h preserves the monoid product, i.e., for every two elements k and l in the monoid $\mathbb{M}$, we have $h(k)h(l) = h(kl)$. A homomorphism h is called a *monoid homomorphism* iff h preserves the neutral element, i.e., $h(\lambda_{\mathbb{M}}) = \lambda_{\mathbb{M}'}$. There are homomorphisms which do not preserve the neutral element. We follow the ususal terminology in the literature. Whenever we use the term homomorphism, we really mean monoid homomorphism. We call the homomorphism h a *non-erasing* homomorphism iff for every element m in $\mathbb{M}$, we have $h(m) = \lambda_{\mathbb{M}'}$ only if m is the neutral element of $\mathbb{M}$.

We extend the homomorphism h. If L is a subset of $\mathbb{M}$, we define $h(L)$ as the set of all k in $\mathbb{M}'$ such that for some m in L, we have $h(m) = k$. We denote the *inverse* of h by h^{-1}. We define it on subsets of $\mathbb{M}'$, if K is a subset of $\mathbb{M}'$, $h^{-1}(K)$ yields the set of all m in $\mathbb{M}$, such that $h(m)$ belongs to K. We call h an *isomorphism* iff for every element k of $\mathbb{M}'$, the set $h^{-1}(\{k\})$ is a singleton. Then, we can regard h^{-1} as a homomorphism from $\mathbb{M}'$ to $\mathbb{M}$. Iff an isomorphism between $\mathbb{M}$ and $\mathbb{M}'$ exists, we call the monoids *isomorphic*.

Assume a set G of generators of $\mathbb{M}$, and assume two homomorphisms h_1 and h_2. If h_1 and h_2 coincide on G, then h_1 and h_2 coincide on the whole monoid $\mathbb{M}$, i.e., if for every m in G, we have $h_1(m) = h_2(m)$, then for every m in $\mathbb{M}$, we have $h_1(m) = h_2(m)$, too.

Assume two monoids $\mathbb{M}$ and $\mathbb{M}'$, once more. We say that $\mathbb{M}$ is a *submonoid* of $\mathbb{M}'$ iff $\mathbb{M}$ is a subset of $\mathbb{M}'$ and the identity function from $\mathbb{M}$ to $\mathbb{M}'$ is a monoid homomorphism, i.e., the monoid product of $\mathbb{M}$ is the monoid product of $\mathbb{M}'$ restricted to elements of $\mathbb{M}$.

By an *alphabet*, we mean a finite set of symbols. Its elements are called *letters*. Assume an alphabet Σ. We denote the *free monoid over* Σ by Σ^*. Its underlying set is the set of all words (strings) consisting of letters of Σ, the monoid product is the concatenation, and the neutral element is the empty string. Obviously, Σ is the set of generators of Σ^*. For every word w in Σ^*, we call the number of letters of w the *length* of w, and denote it by $|w|$. For every letter a in Σ and every word w in Σ^*, we denote the number of occurrences of a in w by $|w|_a$.

Cartier and Foata introduced the concept of the free partially commutative monoids in 1969 [4]. In 1977 Mazurkiewicz considered this concept as a potential model for concurrent systems [21]. Since then, free partially commutative monoids are examined by both mathematicians and theoretical computer scientists. For a general overview, I recommend the surveys [7, 8, 9].

Assume an alphabet Σ. We call a binary relation I over Σ an *independence relation* iff I is irreflexive and symmetric. For every pair of letters a and b with aIb, we say that a and b are *independent*, otherwise a and b are *dependent*. We call the pair (Σ, I) an *independence alphabet*.

We call two words w_1, w_2 in Σ^* equivalent iff we can transform w_1 into w_2 by finitely many exchanges of independent adjacent letters which we denote by $w_1 \sim_I w_2$. For instance, if a and c are independent letters, *baacbac*, *bacabac* and *bcaabca* are mutually equivalent words.

The relation $\sim_I$ is an equivalence relation. For every word w in Σ^*, we denote by $[w]_I$ the equivalence class of w. Moreover, $\sim_I$ is a congruence relation. This means, for every words w_1, w_1', w_2, w_2' in Σ^* with $w_1 \sim_I w_2$ and $w_1' \sim_I w_2'$, we have $w_1 w_1' = w_2 w_2'$. Therefore, we can define a monoid with the sets $[w]_I$ as elements. For any words w_1 and w_2, we define the product of $[w_1]_I$ and $[w_2]_I$ by $[w_1 w_2]_I$. We denote this monoid by $\mathbb{M}(\Sigma, I)$ and call it the *trace monoid* over Σ and I. Its elements, i.e., the equivalence classes $[w]_I$, are called *traces* and its subsets are called *trace languages* or shortly *languages*. The function $[\,]_I$ is a homomorphism from the free monoid Σ^* to $\mathbb{M}(\Sigma, I)$. As long as no confusion arises, we omit the index I at $[\,]_I$.

If I is the empty relation over Σ, the trace monoid $\mathbb{M}(\Sigma, I)$ is the free monoid. If I is the biggest irreflexive relation over Σ, i.e., two letters a and b are independent iff a and b are different, then the trace monoid $\mathbb{M}(\Sigma, I)$ is the free commutative monoid over Σ. Opposed to this very brief introduction, we formally define P3 and C4.

Lemma 2.1 Assume two disjoint alphabets Σ_1 and Σ_2, and assume the independence relation $I := \Sigma_1 \times \Sigma_2 \cup \Sigma_2 \times \Sigma_1$. The trace monoid $\mathbb{M}(\Sigma_1 \cup \Sigma_2, I)$ is isomorphic to the monoid $\binom{\Sigma_1^*}{\Sigma_2^*}$. An isomorphism maps every letter a of Σ_1 to $\binom{a}{\lambda}$, and every letter b of Σ_2 to $\binom{\lambda}{b}$. □

This lemma is an application of a method by MAZURKIEWICZ to transform arbitrary trace monoids into (sub)monoids of Cartesian Products of free monoids [20, 21]. Iff one of the alphabets Σ_1 and Σ_2 is a doubleton, and the other one is a singleton, we denote by P3 both the monoid $\binom{\Sigma_1^*}{\Sigma_2^*}$ and the independence alphabet $(\Sigma_1 \cup \Sigma_2, I)$ with I from Lemma 2.1. Iff both alphabets are doubletons, we accordingly use the notion C4. The notions P3 and C4 abbreviate *path of 3 letters* and *cycle of 4 letters*, respectively. Whenever we deal with P3 or C4, we regard the homomorphism $[\,]$ as a homomorphism from $(\Sigma_1 \cup \Sigma_2)^*$ to $\binom{\Sigma_1^*}{\Sigma_2^*}$.

Assume two independence alphabets (Σ_1, I_1) and (Σ_2, I_2). We say that (Σ_1, I_1) is a *subalphabet* of (Σ_2, I_2) iff Σ_1 is a subset of Σ_2, and I_1 is the restriction of I_2 to the letters of Σ_1, i.e., we have $I_1 = I_2 \cap (\Sigma_1 \times \Sigma_1)$. Then, the monoid $\mathbb{M}(\Sigma_1, I_1)$ is a submonoid of $\mathbb{M}(\Sigma_2, I_2)$. For instance, P3 is a subalphabet and a submonoid of C4.

Assume an independence alphabet (Σ, I). A trace t in $\mathbb{M}(\Sigma, I)$ is called *connected* iff for every non-empty traces t_1 and t_2 with $t = t_1 t_2$, there is a letter a occurring in t_1 and there is a letter b occurring in t_2, such that a and b are dependent. A trace language L is called *connected* iff every trace in L is connected. A trace $\binom{u}{v}$ in P3 or C4 is connected iff u or v is the empty word λ.

2.2 Rational Sets

Rational expressions and rational sets were introduced by KLEENE in 1956 [19]. I give a brief definition, I appreciate, e.g., [2, 10] for deeper understanding.

Definition 2.2 Assume a monoid $\mathbb{M}$. The set of *rational expressions* over $\mathbb{M}$, denoted by $\mathrm{REX}(\mathbb{M})$, is the least set which contains the symbol Ω, every element a of $\mathbb{M}$, and for every $r, r_1, r_2 \in \mathrm{REX}(\mathbb{M})$, $\mathrm{REX}(\mathbb{M})$ also contains (r^*), $(r_1 \cup r_2)$ and $(r_1 r_2)$. □

Rational expressions define rational languages.

Definition 2.3 Assume a monoid $\mathbb{M}$. Every rational expression r over $\mathbb{M}$ defines a language $L(r) \subseteq \mathbb{M}$ in the following way

- $L(\Omega) := \emptyset$, and for every $a \in \mathbb{M}$, $L(a) := \{a\}$,

- $L(r_1 \cup r_2) := L(r_1) \cup L(r_2)$, and $L(r_1 r_2) := L(r_1)L(r_2)$,
- $L(r^*) := L(r)^*$.

A language $L \subseteq \mathbb{M}$ is called a *rational* language iff there is a rational expression r such that $L(r) = L$. RAT($\mathbb{M}$) denotes the class of all rational languages over $\mathbb{M}$. □

We omit parentheses by assuming that the star has the highest priority, followed by the monoid operation. We further omit outermost parentheses, and parentheses superfluous by associativity of set union and monoid operation, e.g., we denote $(r \cup (r_1 \cup (r_2(r_3^*))))$ by $r \cup r_1 \cup r_2 r_3^*$. We use some usual convenient abbreviations, where n is any natural number: We use r^+ to abbreviate rr^*, r^0 to denote $\lambda_{\mathbb{M}}$ and r^{n+1} to abbreviate $r^n r$. Further, we write $r^{\geq n}$ to denote $r^n r^*$, and $r^{>n}$ to denote $r^{n+1}r^*$. Accordingly, we use $r^{\leq n}$ and $r^{<n+1}$ to abbreviate $r^0 \cup r^1 \cup \ldots \cup r^n$. For convenience, we allow to write $r^{<0}$ by treating it as the rational expression Ω.

If $S = \{s_1, \ldots, s_n\}$ is a finite subset of $\mathbb{M}$, we use S to denote the rational expression $s_1 \cup \ldots \cup s_n$. Hence, if Σ is an alphabet, then we regard Σ as a rational expression, its language consists of the letters in Σ.

Obviously, for every monoid $\mathbb{M}$, the class RAT($\mathbb{M}$) contains the empty set and every finite subset of $\mathbb{M}$, and it is closed under union, monoid operation and iteration. For every finitely generated monoid $\mathbb{M}$, e.g., for every trace monoid, RAT($\mathbb{M}$) contains $\mathbb{M}$ itself. Now, we show that rational languages are closed under homomorphisms.

Lemma 2.4 Assume two monoids $\mathbb{M}$ and $\mathbb{M}'$ and a homomorphism $h : \mathbb{M} \to \mathbb{M}'$. For every language $L \in \mathrm{RAT}(\mathbb{M})$, we have $h(L) \in \mathrm{RAT}(\mathbb{M}')$. Moreover, if h is computable, then there is an algorithm which transforms every rational expression $r \in \mathrm{REX}(\mathbb{M})$ into a rational expression $r' \in \mathrm{REX}(\mathbb{M})$, such that $h(L(r)) = L(r')$. □

Proof: We show the construction of r', such that $h(L(r)) = L(r')$. We extend h to a function $h' : \mathrm{REX}(\mathbb{M}) \to \mathrm{REX}(\mathbb{M}')$ to construct r' "top down". For every monoid element m, we set $h'(m) := h(m)$, and we set $h'(\Omega) := \Omega$. For every rational expressions r_1, r_2 and r, we define $h'(r_1 \cup r_2) := h'(r_1) \cup h'(r_2)$, $h'(r_1 r_2) := h'(r_1)h'(r_2)$ and $h'(r^*) := h'(r)^*$. We can proceed the verification $h(L(r)) = L(h'(r))$ inductively by using the fact that h is a homomorphism. □

Generally, RAT($\mathbb{M}$) is not closed under inverse homomorphisms, i.e., using the terminology above, for some r' in REX($\mathbb{M}'$), the set $h^{-1}(L(r'))$ does not necessarily yield a rational language. We will establish a suitable example in the next subsection. We show one helpful lemma concerning rational expressions in finite monoids.

Lemma 2.5 Assume a *finite* monoid $\mathbb{M}$. There is an algorithm which for every rational expression $r \in \mathrm{REX}(\mathbb{M})$ computes the set $L(r)$. □

Proof: We sketch a recursive algorithm. We can obviously compute the language of rational expressions which are just monoids elements or Ω. The algorithm evaluates expressions $r_1 r_2$, $r_1 \cup r_2$ or r^* recursively by firstly computing $L(r_1)$ and $L(r_2)$, resp. $L(r)$, and computing their product, union or iteration, respectively. If $\mathbb{M}$ is a finite monoid, there are algorithms computing product, union and iteration of subsets of $\mathbb{M}$. □

2.3 Recognizable Sets

As long as computers are finite devices, but there are possibly arbitrary big data, we have the problem to process data of arbitrary size by finite devices. The concept of recognizability describes a formal method how to use finite machines to deal with infinite objects. It originates from Mezei and Wright from 1967 [26]. There are numerous equivalent definitions. I introduce it as far as we use it in this paper, for a more general overview I recommend [2, 10]. I took most of the contents of this section from there.

Definition 2.6 Assume a monoid $\mathbb{M}$. An $\mathbb{M}$-*automaton* is a triple $\mathcal{A} = [Q, h, F]$, where Q is a finite monoid, h is a homomorphism $h : \mathbb{M} \to Q$ and F is a subset of Q. The *language* of an $\mathbb{M}$-automaton $\mathcal{A}$ is defined by $L(\mathcal{A}) = h^{-1}(F)$. □

We call Q the underlying monoid of $\mathcal{A}$, and the elements of Q the states of $\mathcal{A}$. We further call h the homomorphism of $\mathcal{A}$, and F the set of accepting states of $\mathcal{A}$. If $L(\mathcal{A}) = L$, then we say that $\mathcal{A}$ defines L or $\mathcal{A}$ is an $\mathbb{M}$-automaton for L. The $\mathbb{M}$-automata define recognizable languages over $\mathbb{M}$. We call a subset L of M a *recognizable* language over $\mathbb{M}$ iff there is an $\mathbb{M}$-automaton $\mathcal{A}$, such that $L = L(\mathcal{A})$. We denote the class of all recognizable languages over $\mathbb{M}$ by REC($\mathbb{M}$).

Definition 2.6 shows a common way to define recognizability in arbitrary monoids. Following Courcelle [6], we call the triple $[Q, h, F]$ an $\mathbb{M}$-automaton. The following theorem is a classic one, you find the proof, e.g., in [2, 10].

Theorem 2.7 Assume a monoid $\mathbb{M}$. The class REC($\mathbb{M}$) contains the empty set $\emptyset$, $\mathbb{M}$ itself and it is closed under union, intersection, complement and inverse homomorphisms. Moreover, we can construct $\mathbb{M}$-automata for the empty set and $\mathbb{M}$, and there are algorithms which construct for every two $\mathbb{M}$-automata $\mathcal{A}_1$ and $\mathcal{A}_2$ $\mathbb{M}$-automata for the sets $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$, $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$ and $\mathbb{M} \setminus L(\mathcal{A}_1)$. □

There are monoids, having finite subsets which are not recognizable. There are further monoids, in which the product of two recognizable subsets is not always a recognizable set. And further, the iteration of a recognizable set is not always recognizable. But, we have the following theorem for trace monoids:

Theorem 2.8 Assume a trace monoid $\mathbb{M}(\Sigma, I)$. The class REC($\mathbb{M}(\Sigma, I)$) contains all finite subsets of $\mathbb{M}(\Sigma, I)$, and it is closed under monoid product and under iteration of *connected* trace languages. □

Recognizability of finite trace languages is obvious. The proof of the closure under monoid product originates from Fliess [12]. Closure under iteration of connected trace languages is due to Ochmański [28], Clerbout and Latteux [5], and Métivier [23]. In [27], you find a recent survey on recognizable trace languages, it contains neat little proofs of the assertions in Theorem 2.8. The following theorem allows us to generalize some undecidability results.

Theorem 2.9 Assume an independence alphabet (Σ_2, I_2) with a subalphabet (Σ_1, I_1). For every language L in $\mathbb{M}(\Sigma_1, I_1)$, L is a recognizable language over $\mathbb{M}(\Sigma_1, I_1)$ iff L is a recognizable language over $\mathbb{M}(\Sigma_2, I_2)$. Moreover, there is an algorithm which computes for every $\mathbb{M}(\Sigma_1, I_1)$-automaton an $\mathbb{M}(\Sigma_2, I_2)$-automaton for the same language. Further, there is an algorithm which computes for every $\mathbb{M}(\Sigma_2, I_2)$-automaton for some language L in $\mathbb{M}(\Sigma_1, I_1)$ an $\mathbb{M}(\Sigma_1, I_1)$-automaton for L. □

Proof: We observe that $\mathbb{M}(\Sigma_1, I_1)$ is a submonoid of $\mathbb{M}(\Sigma_2, I_2)$.

We firstly assume an $\mathbb{M}(\Sigma_2, I_2)$-automaton $\mathcal{A} = [Q, h, F]$ for some language L over $\mathbb{M}(\Sigma_1, I_1)$. We define an $\mathbb{M}(\Sigma_1, I_1)$-automaton $\mathcal{A}'$ for L. The underlying monoid and the set of accepting states of $\mathcal{A}'$ are Q and F from $\mathcal{A}$. We define the homomorphism h' of $\mathcal{A}'$ as the restriction of the homomorphism h of $\mathcal{A}$ to the traces in $\mathbb{M}(\Sigma_1, I_1)$. It is straightforward to verify that $\mathcal{A}'$ is an $\mathbb{M}(\Sigma_1, I_1)$-automaton for L.

For the other direction, we assume an $\mathbb{M}(\Sigma_1, I_1)$-automaton $\mathcal{A} = [Q, h, F]$ for L, and we define an $\mathbb{M}(\Sigma_2, I_2)$-automaton $\mathcal{A}'$ for L. The monoid $\mathbb{M}(\Sigma_2, I_2)$ is an extension of the monoid $\mathbb{M}(\Sigma_1, I_1)$. Consequently, we extend $\mathcal{A}$ to obtain $\mathcal{A}'$. We perform this by adding a garbage state q' to the underlying monoid of $\mathcal{A}$ and adjusting the homomorphism of $\mathcal{A}$.

We define the underlying monoid Q' of $\mathcal{A}'$ by $Q' := Q \cup \{q'\}$, such that q' does not already belong to Q. We extend the monoid product in Q to define the monoid product in Q'. For every

pair of elements k, l in Q', the product kl in Q' yields the same result as the product in Q, as long as neither k nor l is the new element q'. Otherwise, the monoid product yields q'.

We extend the homomorphism h of $\mathcal{A}$ to the homomorphism $h' : \mathbb{M}(\Sigma_2, I_2) \to Q'$ of $\mathcal{A}'$. For every trace t in $\mathbb{M}(\Sigma_2, I_2)$ we define $h'(t)$ in the following way. If t contains a letter which does not belong to Σ_1, the trace t is a trace in $\mathbb{M}(\Sigma_2, I_2) \setminus \mathbb{M}(\Sigma_1, I_1)$, and we set $h'(t) := q'$. If t consists of letters from Σ_1, it is a trace in $\mathbb{M}(\Sigma_1, I_1)$, and we set $h'(t) = h(t)$.

We use the set F of accepting states of $\mathcal{A}$ as set of accepting states of $\mathcal{A}'$. We have to verify that $\mathcal{A}' = [Q', h', F]$ is an $\mathbb{M}(\Sigma_2, I_2)$-automaton for L. We have to show that Q' is a finite monoid, that h' is a homomorphism and that $L(\mathcal{A}')$ really yields L. These assertions are straightforward. □

We need a theorem by MEZEI concerning recognizable sets in Cartesian Products. It is not published by the author himself, but, it is widely known as MEZEI's Theorem, you find it in, e.g., [2, 10].

Theorem 2.10 Assume two monoids $\mathbb{M}$ and $\mathbb{M}'$. A set L is recognizable in $\binom{\mathbb{M}}{\mathbb{M}'}$ iff there is a natural number n, and further, there are recognizable sets $L_1, \ldots, L_n \subseteq \mathbb{M}$ and recognizable sets $L'_1, \ldots, L'_n \subseteq \mathbb{M}'$, such that:

$$L = \binom{L_1}{L'_1} \cup \ldots \cup \binom{L_n}{L'_n}$$

Moreover, there is an algorithm which computes for every $\binom{\mathbb{M}}{\mathbb{M}'}$-automaton the number n, $\mathbb{M}$-automata for $L_1, \ldots, L_n$ and $\mathbb{M}'$-automata for $L'_1, \ldots, L'_n$, and vice versa. □

2.4 Relations between Recognizable Sets and Rational Sets

The following theorem is due to KLEENE [19].

Theorem 2.11 Assume a finite alphabet Σ. A language $L \subseteq \Sigma^*$ is rational iff L is recognizable, i.e., $\mathrm{RAT}(\Sigma^*) = \mathrm{REC}(\Sigma^*)$. Moreover, there are algorithms which construct for every rational expression r a Σ^*-automaton for $L(r)$, and vice versa. □

KLEENE stated his theorem using the term *regular* instead of recognizable. We better avoid this term. You find the proof in teaching books concerning formal language theory, e.g., [17, 35]. We trivially conclude that the class of rational languages of a free monoid over a finite alphabet is closed under intersection and complement. We must not generalize it to arbitrary monoids, but in finitely generated monoids, we have one direction due to MCKNIGHT [22, 2, 10].

Theorem 2.12 Assume a monoid $\mathbb{M}$. It holds $\mathrm{REC}(\mathbb{M}) \subseteq \mathrm{RAT}(\mathbb{M})$ iff $\mathbb{M}$ is finitely generated. Moreover, there is an algorithm which constructs for every $\mathbb{M}$-automaton $\mathcal{A}$ a rational expression r with $L(r) = L(\mathcal{A})$. □

Example 2.13 We consider the alphabets $\Sigma = \{a, b, c\}$, $\Sigma_1 = \{a, c\}$, $\Sigma_2 = \{b\}$ and the monoids Σ^* and $\mathrm{P3} = \binom{\Sigma_1^*}{\Sigma_2^*}$. We define the language L in Σ^* by the rational expression $r = (ab)^*$. Hence, L is rational and recognizable. We apply the homomorphism $[\,]$ on L, we get $[L] = \left\{ \binom{a^n}{b^n} \,\middle|\, n \in \mathbb{N} \right\}$. We show in two ways that $[L]$ is not recognizable. If $[Q, F, h]$ is a P3-automaton for $[L]$, there are two different natural numbers i and j, such that $h\binom{a^i}{\lambda} = h\binom{a^j}{\lambda}$. So, we have $h\binom{a^i}{\lambda} h\binom{\lambda}{b^i} = h\binom{a^j}{\lambda} h\binom{\lambda}{b^i}$, i.e., $h\binom{a^i}{b^i} = h\binom{a^j}{b^i}$. Hence, either both or none of $\binom{a^i}{b^i}$ and $\binom{a^j}{b^i}$ belong to $[L]$, which is a contradiction. Another way to show that $[L]$ is not recognizable is to use the closure properties of recognizable sets. By applying the inverse homomorphism $[\,]^{-1}$ om $[L]$, we get $[[L]]^{-1} = \{w \mid |w|_a = |w|_b\,,\ |w|_c = 0\}$. This language is not recognizable. If we assume that $[[L]]^{-1}$ is recognizable, its intersection with the recognizable language defined by a^*b^* would also be recognizable. But, this intersection yields $\{a^n b^n \mid n \in \mathbb{N}\}$, which is not recognizable.

However, $[L]$ is rational. We can use Lemma 2.4 to show that $[L] = L(\binom{a}{b}^*)$. On the other hand, $[[L]]^{-1}$ is not recognizable as shown, and by Theorem 2.11, $[[L]]^{-1}$ is not rational.

To sum up, L and the application of $[\,]$ show that the class of recognizable languages is not closed under homomorphisms, $[L]$ and the application of $[\,]^{-1}$ show that the rational languages are not closed under inverse homomorphisms. Furthermore, $[L]$ is an example for a rational language in P3, which is not recognizable. The singleton $\{\binom{a}{b}\}$ is an example for a recognizable language with a non-recognizable iteration.

Now, we show that rational languages in P3 are not closed under intersection. We define $L_1 = \binom{a}{b}^*\binom{c}{\lambda}^*$ and $L_2 = \binom{a}{\lambda}^*\binom{c}{b}^*$. We call their intersection $L' = \left\{ \binom{a^n c^n}{b^n} \;\middle|\; n \in \mathbb{N} \right\}$. We assume L' is rational. We apply a homomorphism $h_{ac} : \mathrm{P3} \to \Sigma_1^*$ to L'. It simply erases the letter b. We get $h_{ac}(L') = \{a^n c^n \mid n \in \mathbb{N}\}$, which is also rational, because homomorphisms preserve rationality. This is a contradiction, hence L' is not rational.

Consequently, the class of rational languages of P3 is not closed under complement. At least one of the languages L_1, L_2 or $(\mathrm{P3} \setminus L_1) \cup (\mathrm{P3} \setminus L_2)$ is a rational language with a non-rational complement, while the reader is invited to figure out which of them. □

The preceding examples are widely used in the literature. The divergence between recognizabilty and rationality yields some technical problems. Firstly, the term regular is imprecise in this area, it is used as a synonym for two different concepts. Secondly, we cannot simply use rational expressions as a convenient method to define recognizable languages. Later, we use Mezei's Theorem as a crucial tool to define recognizable languages. We state the following decidability result.

Theorem 2.14 Assume some monoid $\mathbb{M}$. There is an algorithm which decides for every rational expression r and every automaton $\mathcal{A}$ whether it holds $L(r) \subseteq L(\mathcal{A})$. □

Proof: Let $\mathcal{A} = [Q, h, F]$. For every element m of $\mathbb{M}$, we have $m \in L(\mathcal{A})$ iff $h(m) \in F$. Hence, for every subset K of $\mathbb{M}$, we have $K \subseteq L(\mathcal{A})$, iff $h(K) \subseteq F$. Consequently, $L(r) \subseteq L(\mathcal{A})$ holds, iff $h(L(r)) \subseteq F$. Thus, by Lemma 2.4 we can construct a rational expression r' over Q, such that $h(L(r)) = L(r')$. Therefore, we can decide $h(L(r)) \subseteq F$ by deciding $L(r') \subseteq F$. We can compute the finite set $L(r')$ by Lemma 2.5, such that $L(r') \subseteq F$ is decidable. Finally, $L(r) \subseteq L(\mathcal{A})$ is decidable. □

I have not seen this result in the literature in this form, but the proof is natural and straightforward, that it is surely already published. We could see this theorem as a corollary to results from Mezei and Wright, because rational sets are special cases of equational sets defined in [26].

Now, we extend Theorem 2.9. The extended version allows us to boil down some undecidability results to C4.

Definition 2.15 Assume two natural numbers k, l with $k > 0$ and $l > 0$, and assume two alphabets $\Gamma_1 = \{a_1, \ldots, a_k\}$ and $\Gamma_2 = \{b_1, \ldots, b_l\}$. Assume further two alphabets $\Sigma_1 = \{a, c\}$ and $\Sigma_2 = \{b, d\}$. The *canonical homomorphisms* $g_1 : \Gamma_1^* \to \Sigma_1^*$ and $g_2 : \Gamma_2^* \to \Sigma_2^*$ are defined in the following way: For every letter a_i in Γ_1, we set $g_1(a_i) = ac^i$, and for every letter b_i in Γ_2, we set $g_2(b_i) = bd^i$.

The *canonical homomorphism* $h : \binom{\Gamma_1^*}{\Gamma_2^*} \to \mathrm{C4}$ is defined componentwise from g_1 and g_2, i.e., for every trace $\binom{u}{v}$ in $\binom{\Gamma_1^*}{\Gamma_2^*}$, we have $h\binom{u}{v} = \binom{g_1(u)}{g_2(v)}$. □

The method to code free monoids over arbitrary finite alphabets into free monoids over doubletons is widely used in the literature. The extension to Cartesian Products of free monoids is from [13]. The canonical codes g_1 and g_2 are not unique, e.g., we could define for every $a_i \in \Gamma_1$, $g_1(a_i) := ca^i$. Hence, h is not unique.

The homomorphisms g_1 and g_2 are injective. Consequently, h is an injective morphism. Just assume two traces $\binom{u_1}{u_2}$ and $\binom{v_1}{v_2}$ in $\binom{\Gamma_1^*}{\Gamma_2^*}$. If we have $h\binom{u_1}{u_2} = h\binom{v_1}{v_2}$, we conclude that $g_1(u_1) = g_1(v_1)$

and $g_2(u_2) = g_2(v_2)$, i.e., $u_1 = v_1$ and $u_2 = v_2$ such that we have $\binom{u_1}{u_2} = \binom{v_1}{v_2}$. The homomorphism h further preserves recognizability.

Theorem 2.16 Assume two alphabets Γ_1 and Γ_2 and further a canonical homomorphism $h : \binom{\Gamma_1^*}{\Gamma_2^*} \to$ C4. A language $L \subseteq \binom{\Gamma_1^*}{\Gamma_2^*}$ is recognizable iff $h(L)$ is a recognizable language in C4. Moreover, there is an algorithm which computes for every $\binom{\Gamma_1^*}{\Gamma_2^*}$-automaton $\mathcal{A}$ a C4-automaton for $h(L(\mathcal{A}))$. □

Proof: We firstly show that recognizability of $h(L)$ implies recognizability of L. The canonical homomorphism h is injective. Hence, we have $L = h^{-1}(h(L))$. Inverse homomorphisms preserve recognizability due to Theorem 2.7 such that $h^{-1}(h(L))$ is recognizable.

To prove the other direction, we take over the notions Σ_1, Σ_2, g_1 and g_2 from Definition 2.15. We assume a $\binom{\Gamma_1^*}{\Gamma_2^*}$-automaton $\mathcal{A}$ for L. We use Theorem 2.10 to decompose L. We obtain a natural number n and automata for languages $L_1, \ldots, L_n$ and $L'_1, \ldots, L'_n$ in Γ_1^* and Γ_2^*, respectively. Then, we can write $h(L)$ as

$$h(L) = \begin{pmatrix} g_1(L_1) \\ g_2(L'_1) \end{pmatrix} \cup \ldots \cup \begin{pmatrix} g_1(L_n) \\ g_2(L'_n) \end{pmatrix}$$

For every i with $1 \leq i \leq n$, we construct a Σ_1^*-automaton for $g_1(L_i)$ as follows. Outgoing from an automaton for L_i, we use Theorem 2.11 to construct a rational expression for L_i. Then, we use Theorem 2.4 to obtain a rational expression for $g_1(L_i)$. And by Theorem 2.11 again, we get a Σ_1^*-automaton for $g_1(L_i)$. By the same way, we construct for every i with $1 \leq i \leq n$ a Σ_2^*-automaton for $g_2(L'_i)$.

From these automata, we can construct a C4-automaton for $h(L)$ using Theorem 2.10. □

2.5 Some Decidability Problems for Trace Languages

The following questions concerning the gap between the classes of recognizable and rational languages in trace monoids arise:

Complement Problem: Can we decide whether the complement of the language of a rational expression is a rational language?

Recognizability Problem: Can we decide whether the language of a rational expression is a recognizable language?

Star Problem: Can we decide whether the iteration of a recognizable language yields a recognizable language?

Finite Power Property Problem: Can we decide whether a recognizable language has the finite power property, i.e., is there a natural number n, such that $L^* = L^{\leq n}$?

By FPPP, we abbreviate the finite power property problem. By asking "can we decide", we ask for easy characterizations of these trace monoids where the four above questions are decidable. SAKAROVITCH answered the first two questions in 1987 and 1992.

Theorem 2.17 Assume a trace monoid $\mathbb{M}(\Sigma, I)$.
The following three assertions are equivalent:

- (Σ, I) does not contain an P3-subalphabet.
- The rational languages of $\mathbb{M}$ form an (effective) Boolean algebra.
- We can decide whether the language of a rational expression yields a recognizable language. □

The equivalence of the first two assertions is proved in [3, 1, 32], the third assertion is added in [33]. $\mathbb{M}(\Sigma, I)$ does not contain an P3-submonoid iff for every three different letters a, b, c of Σ, aIb and bIc imply aIc. If the class of the rational languages of a monoid $\mathbb{M}$ is closed under complement, it is also closed under intersection. Then, RAT($\mathbb{M}$) is a Boolean algebra.

The star problem and the FPPP are open, today. During the recent 14 years, many papers have dealt with these two questions. However, only partial results have been achieved, in general, both problems have remained unsolved. I give just a brief survey about their history. The star problem in the free monoid is trivial due to KLEENE, and it is decidable in free commutative monoids due to GINSBURG and SPANIER [14, 15]. BRZOZOWSKI raised the FPPP in the free monoid in 1966, and it took more than ten years till SIMON and HASHIGUCHI independently showed its decidability [34, 16]. In 1984, OCHMAŃSKY examined recognizable trace languages in his PhD thesis and stated the star problem. During the eighties, OCHMAŃSKY [28], CLERBOUT and LATTEUX [5] and MÉTIVIER [23] independently proved that the iteration of a connected recognizable trace language yields a recognizable trace language. In 1992, SAKAROVITCH found the solution of the recognizability problem shown in Theorem 2.17. This solution trivially implies the decidability of the star problem in trace monoids which do not contain a P3-submonoid. The attempt to extend SAKAROVITCH's characterization to the star problem failed, just in the same year, GASTIN, OCHMAŃSKY, PETIT and ROZOY showed the decidability of the star problem in P3 [13].

During the subsequent years, MÉTIVIER and RICHOMME developed these ideas. They showed decidability of the FPPP of connected trace languages and decidability of the star problem for trace languages containing at most four traces as well as for finite sets containing at most two connected traces [24, 25]. They further proved that decidability of the star problem in any trace monoid implies decidability of the FPPP in any trace monoid [24, 25]. Finally, RICHOMME proved the following theorem.

Theorem 2.18 Assume a trace monoid $\mathbb{M}(\Sigma, I)$. If the monoid $\mathbb{M}(\Sigma, I)$ does not contain a C4-submonoid, then the star problem and the FPPP are decidable. □

The main ideas of the proof are in [31], the complete proof is in [30]. Please note that the *if* in the theorem has just one *f*.

These are just the main partial solutions. Really, there are much more details, study of examples, research on sufficient and necessary conditions...We make the following informal observations by examining this history: Obviously, concerning the iteration of languages, non-connected languages seem to be more complicated than connected ones. The free monoids, monoids without a P3-submonoid, the monoid P3 and the C4 seem to form a list of monoids with increasing difficulty.

Further, rational languages are more complicated than recognizable languages, e.g., in P3 it is undecidable whether the complement of a rational language is rational. Consequently, we have a two dimensional hierarchy of difficulties! Now, we should cut this discussion and work on the goals of this paper. In Section 6, we will continue by comparing the above results to the new ones.

3 A Tricky Language

During this section, we show a method to derive two recognizable trace languages from a given instance of POSTs Correspondence Problem (PCP). We examine how properties of the iteration of one of the defined languages are influenced by the existence or non-existence of a solution of the underlying PCP instance.

I briefly introduce the PCP. An instance of the PCP consists of two non-empty finite alphabets Υ and Σ, and two homomorphisms $\alpha, \beta : \Upsilon^* \to \Sigma^*$. We call the letters of Υ indices, and the words of Υ^* index sequences. A solution of such an instance is a non-empty index sequence w of Υ^+, such that $\alpha(w) = \beta(w)$. The PCP means to decide, whether a given instance of a PCP has a solution or not. It is well known that the PCP is undecidable unless we restrict Σ to singletons. The proof of the following theorem originates from POST in 1946 [29], you find it in teaching books concerning theoretical computer science, e.g., [17].

Theorem 3.1 There is no algorithm which decides whether an instance of the PCP has a solution or not. □

If the word w is a solution, for every natural number $n \geq 1$, the word w^n is also a solution. Hence, an instance of the PCP has either not any solution or infinitely many solutions. We can freely assume, that α and β are non-erasing homomorphisms. This restriction of the PCP is also undecidable. For instance, the proof of Theorem 3.1 above in [17] implies that the PCP is undecidable if both morphisms are non-erasing.

3.1 Definition of $\mathbb{R}$ and $\mathbb{P}$

Now, we formally define the languages $\mathbb{R}$ and $\mathbb{P}$. During this subsection, we assume an instance of the PCP, i.e., we assume two non-empty finite alphabets Υ and Σ and two homomorphisms $\alpha, \beta : \Upsilon^* \to \Sigma^*$. As mentioned, we assume that for every index i in Υ, both $\alpha(i)$ and $\beta(i)$ yield non-empty words. We denote the number of letters of Υ by k, such that we can treat Υ as $\{\mathrm{i}_1, \ldots, \mathrm{i}_k\}$. We call this instance the *underlying PCP instance*.

We enrich the alphabet Υ by nine new letters, we set $\Gamma := \{\mathrm{i}_1, \ldots, \mathrm{i}_k, \mathrm{a}_1, \ldots, \mathrm{a}_9\}$, while we assume that the letters $\mathrm{a}_1, \ldots, \mathrm{a}_9$ do not belong to Σ. For every natural numbers m and n with $1 \leq m < n \leq 9$, we abbreviate the word $\mathrm{a}_n \mathrm{a}_{n+1} \ldots \mathrm{a}_m$ by $\mathrm{a}_{n\text{-}m}$, e.g., we write $\mathrm{a}_{3\text{-}5}$ instead of $\mathrm{a}_3\mathrm{a}_4\mathrm{a}_5$.

Later, we will need a function $\gamma : \Upsilon^* \to \Gamma^*$ to "code" index sequences. We set $\gamma(\lambda) := \mathrm{a}_{1\text{-}9}$ and for every $w \in \Upsilon^*$ and every $i \in \Upsilon$, we set $\gamma(wi) := \gamma(w)\, i\, \mathrm{a}_{1\text{-}9}$. For instance, we have $\gamma(\mathrm{i}_6\mathrm{i}_2) = \mathrm{a}_{1\text{-}9}\, \mathrm{i}_6\, \mathrm{a}_{1\text{-}9}\, \mathrm{i}_2\, \mathrm{a}_{1\text{-}9}$. Obviously, γ is not a homomorphism.

We define a language $\mathbb{R}$ in the trace monoid $\binom{\Gamma^*}{\Sigma^*}$. The homomorphisms α and β do not influence this language, but the alphabets Γ and Σ are important.

Definition 3.2
The language $\mathbb{R} \subseteq \binom{\Gamma^*}{\Sigma^*}$ is defined by $\mathbb{R} := \begin{pmatrix} \gamma(\Upsilon^+) \\ \Sigma^* \end{pmatrix}$. □

Soon, we will prove that $\mathbb{R}$ is a recognizable language. We denote the complement of $\mathbb{R}$ by $\overline{\mathbb{R}}$, i.e., $\overline{\mathbb{R}} := \binom{\Gamma^*}{\Sigma^*} \setminus \mathbb{R}$. Consequently, $\overline{\mathbb{R}}$ yields the language $\binom{\Gamma^* \setminus \gamma(\Upsilon^+)}{\Sigma^*}$. We define a language $\mathbb{P}$. The definition is more complex, the morphisms α and β play a crucial role. I recommend to read the definition just briefly, now, and to study the details when we apply the definition.

Definition 3.3 The language $\mathbb{P} \subseteq \binom{\Gamma^*}{\Sigma^*}$ is defined as the union of the following sets:

$$\mathbb{P}_{1,1} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} \mathrm{a}_{1\text{-}9}\, i\, \mathrm{a}_1 \\ \Sigma^{<|\alpha(i)|} \end{pmatrix} \right\} \qquad \mathbb{P}_{1,2} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} \mathrm{a}_{2\text{-}9}\, i\, \mathrm{a}_1 \\ \Sigma^{\leq|\alpha(i)|} \end{pmatrix} \right\} \qquad \mathbb{P}_{1,3} := \left\{ \begin{pmatrix} \mathrm{a}_{2\text{-}9} \\ \lambda \end{pmatrix} \right\}$$

$$\mathbb{P}_{2,1} := \left\{ \begin{pmatrix} \mathrm{a}_{1\text{-}2} \\ \lambda \end{pmatrix} \right\} \qquad \mathbb{P}_{2,2} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} \mathrm{a}_{3\text{-}9}\, i\, \mathrm{a}_{1\text{-}2} \\ \Sigma^{|\alpha(i)|} \end{pmatrix} \right\} \qquad \mathbb{P}_{2,5} := \left\{ \begin{pmatrix} \mathrm{a}_{4\text{-}9} \\ \lambda \end{pmatrix} \right\}$$

$$\mathbb{P}_{2,3} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} \mathrm{a}_{3\text{-}9}\, i\, \mathrm{a}_{1\text{-}3} \\ \Sigma^{|\alpha(i)|} \setminus \{\alpha(i)\} \end{pmatrix} \right\} \qquad \mathbb{P}_{2,4} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} \mathrm{a}_{4\text{-}9}\, i\, \mathrm{a}_{1\text{-}3} \\ \Sigma^{|\alpha(i)|} \end{pmatrix} \right\}$$

$$\mathbb{P}_{3,1} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} \mathrm{a}_{1\text{-}9}\, i\, \mathrm{a}_{1\text{-}4} \\ \Sigma^{>|\alpha(i)|} \end{pmatrix} \right\} \qquad \mathbb{P}_{3,2} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} \mathrm{a}_{5\text{-}9}\, i\, \mathrm{a}_{1\text{-}4} \\ \Sigma^{|\alpha(i)|} \end{pmatrix} \right\} \qquad \mathbb{P}_{3,3} := \left\{ \begin{pmatrix} \mathrm{a}_{5\text{-}9} \\ \lambda \end{pmatrix} \right\}$$

$$\mathbb{P}_{4,1} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} \mathrm{a}_{1\text{-}9}\, i\, \mathrm{a}_{1\text{-}5} \\ \Sigma^{<|\beta(i)|} \end{pmatrix} \right\} \qquad \mathbb{P}_{4,2} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} \mathrm{a}_{6\text{-}9}\, i\, \mathrm{a}_{1\text{-}5} \\ \Sigma^{\leq|\beta(i)|} \end{pmatrix} \right\} \qquad \mathbb{P}_{4,3} := \left\{ \begin{pmatrix} \mathrm{a}_{6\text{-}9} \\ \lambda \end{pmatrix} \right\}$$

$$\mathbb{P}_{5,1} := \left\{ \begin{pmatrix} a_{1..6} \\ \lambda \end{pmatrix} \right\} \qquad \mathbb{P}_{5,2} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} a_{7..9}\, i\, a_{1..6} \\ \Sigma^{|\beta(i)|} \end{pmatrix} \right\} \qquad \mathbb{P}_{5,5} := \left\{ \begin{pmatrix} a_{8..9} \\ \lambda \end{pmatrix} \right\}$$

$$\mathbb{P}_{5,3} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} a_{7..9}\, i\, a_{1..7} \\ \Sigma^{|\beta(i)|} \setminus \{\beta(i)\} \end{pmatrix} \right\} \qquad \mathbb{P}_{5,4} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} a_{8..9}\, i\, a_{1..7} \\ \Sigma^{|\beta(i)|} \end{pmatrix} \right\}$$

$$\mathbb{P}_{6,1} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} a_{1..9}\, i\, a_{1..8} \\ \Sigma^{>|\beta(i)|} \end{pmatrix} \right\} \qquad \mathbb{P}_{6,2} := \bigcup_{i \in \Upsilon} \left\{ \begin{pmatrix} a_9\, i\, a_{1..8} \\ \Sigma^{|\beta(i)|} \end{pmatrix} \right\} \qquad \mathbb{P}_{6,3} := \left\{ \begin{pmatrix} a_9 \\ \lambda \end{pmatrix} \right\}$$

□

This is a neat little rip. We call the sets $\mathbb{P}_{1,1}, \ldots, \mathbb{P}_{6,3}$ the *parts* of $\mathbb{P}$. As the very first observation, we remark that the parts of $\mathbb{P}$ are mutually disjoint by examining the first components of the traces.

3.2 Properties of $\mathbb{R}$, $\mathbb{P}$ and $\mathbb{P}^*$

The first important property of $\mathbb{R}$, $\overline{\mathbb{R}}$ and $\mathbb{P}$ is recognizability. We further need effective constructions of automata for $\mathbb{R}$, $\overline{\mathbb{R}}$ and $\mathbb{P}$ based on the underlying PCP instance.

Lemma 3.4 The languages $\mathbb{R}$, $\overline{\mathbb{R}}$ and $\mathbb{P}$ are recognizable. Moreover, there are algorithms which construct for every instance of the PCP three $\binom{\Gamma^*}{\Sigma^*}$-automata for $\mathbb{R}$, $\overline{\mathbb{R}}$ and $\mathbb{P}$, respectively. □

Proof: We start with the automaton for $\mathbb{R}$. The language $\gamma(\Upsilon^+)$ in the free monoid Γ^* is defined by the rational expression $(a_{1..9}\Upsilon)^+ a_{1..9}$. By Theorem 2.11, we can construct a Γ^*-automaton for $\gamma(\Upsilon^+)$. We can also construct a Σ^*-automaton for Σ^*. By Theorem 2.10, we can use these automata to construct a $\binom{\Gamma^*}{\Sigma^*}$-automaton for $\mathbb{R}$. Based on this automaton for $\mathbb{R}$, we obtain a $\binom{\Gamma^*}{\Sigma^*}$-automaton for $\overline{\mathbb{R}}$ by Theorem 2.7.

Now, we show the construction of an $\binom{\Gamma^*}{\Sigma^*}$-automaton for $\mathbb{P}$. Due to the closure under union (Theorem 2.7), we only need to show constructions of $\binom{\Gamma^*}{\Sigma^*}$-automata for each part of $\mathbb{P}$. We proceed it for $\mathbb{P}_{3,1}$. This set is the union of k Cartesian Products. We use Theorem 2.10, we only need to construct for each index i in Υ a Γ^*-automaton for the singleton language $\{a_{1..9} i a_{1..4}\}$ and a Σ^*-automaton for $\Sigma^{>|\alpha(i)|}$. We regard $a_{1..9} i a_{1..4}$ and $\Sigma^{>|\alpha(i)|}$ as two rational expressions over the free monoids Γ^* resp. Σ^*, and by Theorem 2.11, we can construct automata for $\{a_{1..9} i a_{1..4}\}$ and $\Sigma^{>|\alpha(i)|}$. After we constructed these automata for every index i in Υ, we use Theorem 2.10 to construct a $\binom{\Gamma^*}{\Sigma^*}$-automaton for $\mathbb{P}_{3,1}$. The construction of automata for the other parts of $\mathbb{P}$ is similar but simpler. Based on automata for the parts of $\mathbb{P}$, we use Theorem 2.7 to construct an automaton for $\mathbb{P}$. □

Now, we can go over to examine the iteration of $\mathbb{P}$. We are mainly interested in traces in $\mathbb{P}^*$ whose first compound is a word from $\gamma(\Upsilon^+)$.

Lemma 3.5 For every w in Υ^+, we have the following assertions (1) and (2). If w is not a solution of the underlying PCP instance, we further have assertion (3).

$$(1)\ \begin{pmatrix} \gamma(w) \\ \Sigma^* \setminus \{\alpha(w)\} \end{pmatrix} \subseteq \mathbb{P}^* \qquad (2)\ \begin{pmatrix} \gamma(w) \\ \Sigma^* \setminus \{\beta(w)\} \end{pmatrix} \subseteq \mathbb{P}^* \qquad (3)\ \begin{pmatrix} \gamma(w) \\ \Sigma^* \end{pmatrix} \subseteq \mathbb{P}^* \qquad \square$$

At this point, we somehow firmly feel that something very unpleasant will happen in the case that w is a solution of the underlying PCP instance.

Proof: At first, we prove assertion (1). We assume a non-empty word w of indices from Υ, and we assume further a word u from Σ^*, such that $u \neq \alpha(w)$. We have to show that $\binom{\gamma(w)}{u}$ belongs to $\mathbb{P}^*$. We will branch into three cases, depending on whether $|u| < |\alpha(w)|$, $|u| = |\alpha(w)|$ or $|u| > |\alpha(w)|$. We denote by n the length of $|w|$, such that we can treat w as $j_1 \ldots j_n$ for some indices $j_1, \ldots, j_n \in \Upsilon$. Consequently, $\alpha(w)$ equals the composite $\alpha(j_1) \ldots \alpha(j_n)$.

- Case 1: $|u| < |\alpha(w)|$
  We defactorize u into n words $u_1, \ldots, u_n$. Because u is shorter than $\alpha(w)$, we can choose the words $u_1, \ldots, u_n$ in a way that $|u_1| < |\alpha(j_1)|$, and for every l with $2 \leq l \leq n$, we have $|u_l| \leq |\alpha(j_l)|$. At this point, we need the assumption that $\alpha(j_1)$ does not yield the empty word to ensure the existence of a properly shorter word u_1. We show traces $t_1, \ldots, t_{n+1} \in \mathbb{P}$ such that $t_1 \ldots t_{n+1} = \binom{\gamma(w)}{u}$. We define t_1, t_{n+1} and for every l with $2 \leq l \leq n$ the trace t_l in the following way:

$$t_1 := \begin{pmatrix} \mathrm{a}_{1..9}\, j_1\, \mathrm{a}_1 \\ u_1 \end{pmatrix} \qquad t_l := \begin{pmatrix} \mathrm{a}_{2..9}\, j_l\, \mathrm{a}_1 \\ u_l \end{pmatrix} \qquad t_{n+1} := \begin{pmatrix} \mathrm{a}_{2..9} \\ \lambda \end{pmatrix}$$

  We see that $t_1, \ldots, t_{n+1}$ belong to $\mathbb{P}$, namely to $\mathbb{P}_{1,1}$, $\mathbb{P}_{1,2}$ and $\mathbb{P}_{1,3}$. It is a straightforward verification that $t_1 \ldots t_{n+1}$ yields the required $\binom{\gamma(w)}{u}$.

- Case 2: $|u| = |\alpha(w)|$
  We still remember the assumption $u \neq \alpha(w)$ from the beginning of the proof. Because u has the same length as $\alpha(w)$, we can defactorize u into words $u_1, \ldots, u_n$ such that for every l with $1 \leq l \leq n$, we have $|u_l| = |\alpha(j_l)|$. Because $u \neq \alpha(w)$, we know there is some z with $1 \leq z \leq n$, such that $u_z \neq \alpha(j_z)$. Until this case is finished, we assume l and m as all-quantified numbers with $0 < l < z < m < (n+1)$. We show traces $t_0, \ldots, t_{n+1} \in \mathbb{P}$, such that $t_0 \ldots t_{n+1} = \binom{\gamma(w)}{u}$:

$$t_0 := \begin{pmatrix} \mathrm{a}_{1..2} \\ \lambda \end{pmatrix} \quad t_l := \begin{pmatrix} \mathrm{a}_{3..9}\, j_l\, \mathrm{a}_{1..2} \\ u_l \end{pmatrix} \quad t_z := \begin{pmatrix} \mathrm{a}_{3..9}\, j_z\, \mathrm{a}_{1..3} \\ u_z \end{pmatrix} \quad t_m := \begin{pmatrix} \mathrm{a}_{4..9}\, j_m\, \mathrm{a}_{1..3} \\ u_m \end{pmatrix} \quad t_{n+1} := \begin{pmatrix} \mathrm{a}_{4..9} \\ \lambda \end{pmatrix}$$

  We see that $t_0, \ldots, t_{n+1}$ belong to $\mathbb{P}$, namely to $\mathbb{P}_{2,1}, \ldots, \mathbb{P}_{2,5}$. As in the previous case, we easily it verify that $t_0 \ldots t_{n+1}$ really yields $\binom{\gamma(w)}{u}$.

- Case 3: $|u| > |\alpha(w)|$
  We perform this as we performed the previous two cases. We defactorize u into n words $u_1, \ldots, u_n$, such that u_1 is properly longer than $\alpha(j_1)$, and we further have for every l with $2 \leq l \leq n$, a word u_l with the same length as $\alpha(j_l)$. As we did in the previous two cases, we choose suitable traces $t_1, \ldots, t_{n+1}$, but this time, we choose them from $\mathbb{P}_{3,1}$, $\mathbb{P}_{3,2}$ and $\mathbb{P}_{3,3}$.

Now, we have completed the proof of assertion (1). We can prove assertion (2) in the same way using the parts $\mathbb{P}_{4,1}, \ldots, \mathbb{P}_{6,3}$ of $\mathbb{P}$. However, I think it is not necessary to proceed it. If w is not a solution of the underlying PCP instance, we know that $\alpha(w) \neq \beta(w)$, and consequently, assertion (1) and (2) together imply assertion (3). □

We state the following corollary as an obvious conclusion from Lemma 3.5.

Corollary 3.6 If the underlying PCP instance has no solution, we have $\mathbb{R} \subseteq \mathbb{P}^*$. □

Well, we made half of the way. We need some kind of opposite to Lemma 3.5 and Corollary 3.6. We show that some traces in $\mathbb{R}$ do not belong to $\mathbb{P}^*$ if the underlying PCP has a solution. Together with Corollary 3.6, we obtain a strong tool, which will allow us to proceed straightforward proofs of the main goals of this paper.

Lemma 3.7 Assume any word w in Υ^+.
If w is a solution of the underlying PCP instance, we have $\begin{pmatrix} \gamma(w) \\ \alpha(w) \end{pmatrix} \notin \mathbb{P}^*$. □

You should recognize that we can replace $\alpha(w)$ by $\beta(w)$.

Proof: We perform an indirect proof. We assume a word w from Υ^+ such that $\alpha(w) = \beta(w)$. We assume that the trace $\binom{\gamma(w)}{\alpha(w)}$ belongs to $\mathbb{P}^*$, and show a contradiction. If $\binom{\gamma(w)}{\alpha(w)}$ belongs to $\mathbb{P}^*$, then there are a natural number n and traces $t_1, \ldots, t_n$ in $\mathbb{P}$ such that $t_1 \ldots t_n$ yields $\binom{\gamma(w)}{\alpha(w)}$. Because w is non-empty, we know that $n \geq 1$. Because there is not any trace in $\mathbb{P}$ whose first compound is $\gamma(w)$, we see that $n \geq 2$. For every natural number l with $1 \leq l \leq n$, we denote the components of t_l by v_l and u_l, i.e., $t_l = \binom{v_l}{u_l}$.

For every l with $1 \leq l \leq n$, the word v_l contains at most one index from Υ (cf. Def. 3.3). By j_l, we denote the restriction of v_l to indices from Υ, i.e., we get j_l by removing all letters $a_1, \ldots, a_9$ from v_l. Obviously, for every l with $1 \leq l \leq n$, the word j_l is the empty word or a single letter. We further see that the composite $j_1 \ldots j_l$ yields w.

For every l with $1 \leq l \leq n$, we have $v_l \neq \lambda$ (cf. Def. 3.3). We further have $v_1 \ldots v_n = \gamma(w)$. Hence, the first letter of the word v_1 is the letter a_1. Thus, the trace t_1 belongs to exactly one of the sets $\mathbb{P}_{1,1}$, $\mathbb{P}_{2,1}$, $\mathbb{P}_{3,1}$, $\mathbb{P}_{4,1}$, $\mathbb{P}_{5,1}$ or $\mathbb{P}_{6,1}$. We branch into these six cases.

- Case 1: $t_1 \in \mathbb{P}_{1,1}$
  We examine the traces $t_2, \ldots, t_n$. The trace t_n cannot belong to $\mathbb{P}_{1,2}$, because v_n has to end with the letter a_9. Hence, there is at least one trace among $t_2, \ldots, t_n$, which does not belong to $\mathbb{P}_{1,2}$. We see that there is a natural number z with $1 < z \leq n$ such that firstly $t_z \notin \mathbb{P}_{1,2}$, and further, for every l with $1 < l < z$ we have $t_l \in \mathbb{P}_{1,2}$.

  We examine the trace t_z. Its predecessor, the trace t_{z-1} is the trace t_1 or a trace from $\mathbb{P}_{1,2}$. Hence, the last letter of v_{z-1} is the letter a_1, and consequently, v_z has to start with the letter a_2. This implies that t_z has to belong to $\mathbb{P}_{1,2}$ or $\mathbb{P}_{1,3}$, but we have chosen z in a way that t_z does not belong to $\mathbb{P}_{1,2}$. Therefore, t_z belongs to $\mathbb{P}_{1,3}$, i.e., t_z is the trace $\binom{a_{2 \cdot 9}}{\lambda}$.

  The trace t_z must be the last trace in the factorization, because v_z ends with the letter a_9, such that a subsequent trace t_{z+1} had to have a first compound v_{z+1} starting with an index from Υ, and such traces do not belong to any part of $\mathbb{P}$. Consequently, we have $z = n$.

  To sum up, the factorization $t_1, \ldots, t_n$ starts with a trace t_1 from $\mathbb{P}_{1,1}$, it ends with the trace $t_n = \binom{a_{2 \cdot 9}}{\lambda}$ and the traces $t_2, \ldots, t_{n-1}$ belong to $\mathbb{P}_{1,2}$. We examine the words $j_1, \ldots, j_n$: Except j_n which is empty, these words are single indices, such that w is the composite $j_1 \ldots j_{n-1}$. Now, we compare $\alpha(w)$ to the composite of the second components of $t_1, \ldots, t_n$, i.e., we compare $\alpha(j_1) \ldots \alpha(j_{n-1})$ to $u_1 \ldots u_n$. We defined $\mathbb{P}_{1,1}$ and $\mathbb{P}_{1,2}$ in a such a way that u_1 is properly shorter than $\alpha(j_1)$, and for every l with $1 < l < n$, we have $|u_l| \leq |\alpha(j_l)|$. Further, u_n is the empty word λ. Consequently, the composite $u_1 \ldots u_n$ is properly shorter than $\alpha(j_1) \ldots \alpha(j_{n-1})$, i.e., $u_1 \ldots u_n$ is properly shorter than $\alpha(w)$.

  To sum up this case, we have shown that if $t_1, \ldots, t_n$ is a sequence of traces from $\mathbb{P}$, the trace t_1 belongs to $\mathbb{P}_{1,1}$, and the composite of their first components $v_1 \ldots v_n$ yields $\gamma(w)$, then the composite $t_1 \ldots t_n$ cannot yield the trace $\binom{\gamma(w)}{\alpha(w)}$, because the composite of the second components $u_1 \ldots u_n$ is properly shorter than $\alpha(w)$.

- Case 2: $t_1 \in \mathbb{P}_{2,1}$
  Our beginning is quite similar to Case 1. $\mathbb{P}_{2,1}$ is a singleton such that t_1 is the trace $\binom{a_{1..2}}{\lambda}$. We examine the traces $t_2, \ldots, t_n$. The trace t_n cannot belong to $\mathbb{P}_{2,2}$, because v_n has to end with the letter a_9. Hence, there is at least one trace among $t_2, \ldots, t_n$, which does not belong to $\mathbb{P}_{2,2}$. We see, there is a natural number z with $1 < z \leq n$, such that firstly $t_z \notin \mathbb{P}_{2,2}$, and secondly, for every l with $1 < l < z$, we have $t_l \in \mathbb{P}_{2,2}$.

  We examine the trace t_z. The last letter of v_{z-1} is the letter a_2. Then, the first compound of t_z has to start with a_3, but, t_z does not belong to $\mathbb{P}_{2,2}$. Consequently, t_z belongs to $\mathbb{P}_{2,3}$. We also see that t_z is not the last trace in the factorization, i.e., z is properly smaller than n.

  We examine the traces $t_{z+1}, \ldots, t_n$. The trace t_n cannot belong to $\mathbb{P}_{2,4}$. Hence, there is a number x, such that t_x is the leftmost trace in $t_{z+1}, \ldots, t_n$, which does not belong to $\mathbb{P}_{2,4}$.

  We see that t_x has to belong to $\mathbb{P}_{2,5}$, which implies that t_x is the trace $\binom{a_{4..9}}{\lambda}$. Then, t_x has to be the last trace in the factorization $t_1, \ldots, t_n$, i.e., $x = n$.

  To sum up, there is a z with $1 < z < n$, such that the factorization $t_1, \ldots, t_n$ consists of the trace $\binom{a_{1..2}}{\lambda}$, some traces $t_2, \ldots, t_{z-1}$ from $\mathbb{P}_{2,2}$, a trace t_z from $\mathbb{P}_{2,3}$, some traces $t_{z+1}, \ldots, t_{n-1}$ from $\mathbb{P}_{2,4}$ and the trace $\binom{a_{4..9}}{\lambda}$ at the end. We examine the words $j_1, \ldots, j_n$: Except j_1 and j_n which are empty, these words are single indices, such that w is the composite $j_2 \ldots j_{n-1}$. Now, we compare $\alpha(w)$ to the composite of the second components of $t_1, \ldots, t_n$, i.e., we compare $\alpha(j_2 \ldots j_{n-1})$ to $u_2 \ldots u_{n-1}$. We defined $\mathbb{P}_{2,2}$, $\mathbb{P}_{2,3}$ and $\mathbb{P}_{2,4}$ in a such a way that for each l with $1 < l < n$, the words $\alpha(j_l)$ and u_l have the same length. Further, we defined $\mathbb{P}_{2,3}$ such that we have $\alpha(j_z) \neq u_z$. Consequently, the composite $u_1 \ldots u_n$ is not the word $\alpha(w)$.

  To sum up this case, we have shown that if $t_1, \ldots, t_n$ is a sequence of traces from $\mathbb{P}$, the trace t_1 belongs to $\mathbb{P}_{2,1}$, and the composite of their first components $v_1 \ldots v_n$ yields $\gamma(w)$, then the composite $t_1 \ldots t_n$ cannot yield the trace $\binom{\gamma(w)}{\alpha(w)}$, because the composite of the second components $u_1 \ldots u_n$ is different from $\alpha(w)$.

- Case 3: $t_1 \in \mathbb{P}_{3,1}$
  We can show the contradiction exactly as we dealt Case 1. We obtain that the composite $u_1 \ldots u_n$ is properly longer than $\alpha(w)$.

- Case 4, 5 and 6:
  The contradictions are straightforward adaptations of the methods we used to show contradictions in the first three cases. In every case we find out that the composite $u_1 \ldots u_n$ is different from $\beta(w)$. Because we assumed w as a solution of the underlying PCP, we have $\beta(w) = \alpha(w)$, such that $u_1 \ldots u_n$ is different from $\alpha(w)$.

Finally, each case yields a contradiction such that traces $t_1, \ldots, t_n$ from $\mathbb{P}$ whose composition yields $\binom{\gamma(w)}{\alpha(w)}$ do not exist. Consequently, this trace does not belong to $\mathbb{P}^*$. □

Now, we have a method to define languages $\mathbb{R}$ and $\mathbb{P}$ from a PCP instance. Further, we know by Corollary 3.6 and Lemma 3.7 two strong assertions about the connections of properties of $\mathbb{P}^*$ and the existence of solutions of the underlying PCP instance. Unfortunately, we do not have connections between recognizability of $\mathbb{P}^*$ and the existence of a solution of the underlying instance of the PCP.

4 Main Results

Now, we are able to prove the following theorem.

Theorem 4.1 The following five assertions are equivalent

(1) The underlying PCP instance has no solution.

(2) $\mathbb{R} \subseteq \mathbb{P}^*$

(3) $\overline{\mathbb{R}} \cup \mathbb{P}^* = \binom{\Gamma^*}{\Sigma^*}$

(4) $\overline{\mathbb{R}} \cup \mathbb{P}^*$ is recognizable.

(5) $\mathbb{R} \cap \mathbb{P}^*$ is recognizable. □

Proof: This is mainly a summary of results from the previous section.

- (1)→(2) This is just Corollary 3.6.
- (2)→(1) If the underlying PCP instance has a solution w, then $\binom{\gamma(w)}{\alpha(w)}$ does not belong to $\mathbb{P}^*$ by Lemma 3.7, but it belongs to $\mathbb{R}$ by Definition 3.2.
- (2)↔(3) holds obviously.
- (3)→(4) holds obviously, because $\binom{\Gamma^*}{\Sigma^*}$ is recognizable.
- (2)→(5) $\mathbb{R} \cap \mathbb{P}^*$ yields $\mathbb{R}$, which is recognizable due to Lemma 3.4.
- (4)→(1) Assume the underlying PCP instance has a solution w, but the language $\overline{\mathbb{R}} \cup \mathbb{P}^*$ is recognizable, i.e., there is a $\binom{\Gamma^*}{\Sigma^*}$-automaton $\mathcal{A} = [Q, h, F]$ for $\overline{\mathbb{R}} \cup \mathbb{P}^*$. For every $n \geq 1$, the words w^n are mutually different solutions of the underlying PCP instance. We examine the values of the homomorphism h on the traces $\binom{\gamma(w^n)}{\lambda}$. Because Q is finite, there are two different natural numbers $m \geq 1$ and $n \geq 1$, such that $h\binom{\gamma(w^n)}{\lambda} = h\binom{\gamma(w^m)}{\lambda}$. We get the values of h on $\binom{\gamma(w^n)}{\alpha(w^n)}$ and $\binom{\gamma(w^m)}{\alpha(w^n)}$ by defactorizations: We see that $h\binom{\gamma(w^n)}{\alpha(w^n)} = h\binom{\gamma(w^n)}{\lambda} h\binom{\lambda}{\alpha(w^n)}$ and $h\binom{\gamma(w^m)}{\alpha(w^n)} = h\binom{\gamma(w^m)}{\lambda} h\binom{\lambda}{\alpha(w^n)}$, which yields the same. Hence, either both or none of the traces $\binom{\gamma(w^n)}{\alpha(w^n)}$ and $\binom{\gamma(w^m)}{\alpha(w^n)}$ belong to $\overline{\mathbb{R}} \cup \mathbb{P}^*$. But, on one hand, $\binom{\gamma(w^n)}{\alpha(w^n)}$ does not belong to $\overline{\mathbb{R}}$ and it does not belong to $\mathbb{P}^*$ by Lemma 3.7. On the other hand, we have $\alpha(w^n) \neq \alpha(w^m)$, such that assertion (1) of Lemma 3.5 implies that $\binom{\gamma(w^m)}{\alpha(w^n)}$ belongs to $\mathbb{P}^*$.
- (5)→(1) Now, assume again the underlying PCP instance has a solution w, but, $\mathbb{R} \cap \mathbb{P}^*$ is recognizable. As in the previous point, we show that there are two different natural numbers $m \geq 1$ and $n \geq 1$, such that either both or none of the traces $\binom{\gamma(w^n)}{\alpha(w^n)}$ and $\binom{\gamma(w^m)}{\alpha(w^n)}$ belong to $\mathbb{R} \cap \mathbb{P}^*$. But, $\binom{\gamma(w^n)}{\alpha(w^n)}$ does not belong $\mathbb{P}^*$ (Lemma 3.7), while $\binom{\gamma(w^m)}{\alpha(w^n)}$ belongs to both $\mathbb{R}$ and $\mathbb{P}^*$ because of Definition 3.2 and assertion (1) of Lemma 3.5, respectively. □

We generalize these results to trace monoids which contain a C4-submonoid. We perform this in two steps.

Corollary 4.2 There is no algorithm, whose input are two alphabets Σ_1 and Σ_2, and further two $\binom{\Sigma_1^*}{\Sigma_2^*}$-automata for languages K and L of $\binom{\Sigma_1^*}{\Sigma_2^*}$ which decides one of the following properties:

(1) $K \subseteq L^*$

(2) $K \cup L^* = \binom{\Sigma_1^*}{\Sigma_2^*}$

(3) $K \cup L^* \in \mathrm{REC}\binom{\Sigma_1^*}{\Sigma_2^*}$

(4) $K \cap L^* \in \mathrm{REC}\binom{\Sigma_1^*}{\Sigma_2^*}$ □

Proof: This is a straightforward conclusion from Theorem 4.1 and the undecidability of the PCP. Provided an algorithm which decides one of the properties, we could construct an algorithm deciding the PCP. For instance, assume an algorithm to decide universality (2). Then, we have the following algorithm to decide the PCP: It has two alphabets and two morphisms as input, it constructs automata for $\overline{\mathbb{R}}$ and $\mathbb{P}$ as described in the proof of Lemma 3.4. After that, it uses the assumed algorithm to decide universality. By Theorem 4.1, it deduces whether the PCP instance has a solution or not. □

It is not really satisfying, because the alphabets Σ_1 and Σ_2 are not restricted to doubletons. We should boil down Corollary 4.2 to C4 and generalize it to all trace monoids with a C4-submonoid.

Theorem 4.3 Assume an independence alphabet (Σ, I) which contains a C4-subalphabet. There is no algorithm, whose input are two $\mathbb{M}(\Sigma, I)$-automata for languages K and L of $\mathbb{M}(\Sigma, I)$ which decides one of the following properties:

(1) $K \subseteq L^*$

(2) $K \cup L^* = \mathbb{M}(\Sigma, I)$

(3) $K \cup L^* \in \mathrm{REC}\,\mathbb{M}(\Sigma, I)$

(4) $K \cap L^* \in \mathrm{REC}\,\mathbb{M}(\Sigma, I)$ □

Proof: We assume a monoid $\mathbb{M}(\Sigma, I)$ with a C4-submonoid. We show that an algorithm which decides one of the properties in $\mathbb{M}(\Sigma, I)$ can be used to decide the same property in Cartesian Products of free monoids over arbitrary alphabets, which contradicts Corollary 4.2. We deal with some preliminaries before we prove the assertions. Assume two alphabets Σ_1 and Σ_2. We can fix a canonical code $h : \binom{\Sigma_1^*}{\Sigma_2^*} \to \mathrm{C4}$ as in Definition 2.15. Let g be the identity from C4 to $\mathbb{M}(\Sigma, I)$. Consequently, the composition $g(h(\_))$ is an injective homomorphism from $\binom{\Sigma_1^*}{\Sigma_2^*}$ to $\mathbb{M}(\Sigma, I)$. Due to Theorem 2.9 and 2.16, some language T in $\binom{\Sigma_1^*}{\Sigma_2^*}$ is recognizable iff $h(T)$ is recognizable in $\mathbb{M}(\Sigma, I)$. Moreover, given a $\binom{\Sigma_1^*}{\Sigma_2^*}$-automaton for T, we can construct an $\mathbb{M}(\Sigma, I)$-automaton for $h(T)$.

- (1) Assume there is an algorithm deciding this property in $\mathbb{M}(\Sigma, I)$. Then, we can decide property (1) in $\binom{\Sigma_1^*}{\Sigma_2^*}$ as follows: We have two alphabets Σ_1 and Σ_2 and two $\binom{\Sigma_1^*}{\Sigma_2^*}$-automata for languages K and L. We fix a canonical morphism h and construct $\mathbb{M}(\Sigma, I)$-automata for $h(K)$ and $h(L)$. We have $K \subseteq L^*$ iff $h(K) \subseteq h(L)^*$. We can decide the latter condition by the assumed algorithm.

- (3) The language $K \cup L^*$ is a recognizable language in $\binom{\Sigma_1^*}{\Sigma_2^*}$ iff $h(K \cup L^*)$ it is a recognizable language in $\mathbb{M}(\Sigma, I)$. We have $h(K \cup L^*) = h(K) \cup h(L)^*$.

  Therefore, we can use an algorithm to decide (3) in $\mathbb{M}(\Sigma, I)$ to decide (3) in $\binom{\Sigma_1^*}{\Sigma_2^*}$ as follows. Outgoing from $\binom{\Sigma_1^*}{\Sigma_2^*}$-automata for K and L, we construct $\mathbb{M}(\Sigma, I)$-automata for $h(K)$ and $h(L)$. Then, we use the assumed algorithm to decide (3) to decide whether $h(K) \cup h(L)^*$ is recognizable in $\mathbb{M}(\Sigma, I)$. The set $h(K) \cup h(L)^*$ is recognizable iff $K \cup L^*$ is recognizable.

- (4) We proceed this exactly as (3).

- (2) We have $K \subseteq L^*$ iff $(\mathbb{M}(\Sigma, I) \backslash K) \cup L^*$ yields the complete monoid $\mathbb{M}(\Sigma, I)$. Consequently, decidability of (2) would imply decidability of (1). □

5 Additional Results

For the reason of lucidity, I have split the results into two sections. For now, the following results seem to be less important. But, I surely need them in future papers. Based on results from Section 3, we can proceed short and easy proofs. However, proving the results in this section in a seperate paper would cause much more expenditure.

Assume two alphabets Σ_1 and Σ_2. A language T in $\binom{\Sigma_1^*}{\Sigma_2^*}$ is a Cartesian Product iff there are languages L and R in Σ_1^* and Σ_2^*, resp., such that $T = \binom{L}{R}$. Consequently, T is a Cartesian Product iff for every two traces $\binom{u_1}{v_1}$ and $\binom{u_2}{v_2}$ in T, the trace $\binom{u_1}{v_2}$ belongs to T.

In the rest of this section we use the notions from Section 3.

Theorem 5.1 The following three assertions are equivalent

(1) The underlying PCP instance has no solution.

(2) $\overline{\mathbb{R}} \cup \mathbb{P}^*$ is a Cartesian Product.

(3) $\mathbb{R} \cap \mathbb{P}^*$ is a Cartesian Product. □

Proof: This is very similar to the proof of Theorem 4.1.

- (1)→(2) Due to Theorem 4.1, $\overline{\mathbb{R}} \cup \mathbb{P}^*$ yields $\binom{\Gamma^*}{\Sigma^*}$ which is a Cartesian Product.

- (1)→(3) Due to Theorem 4.1 again, $\mathbb{R}$ is a subset of $\mathbb{P}^*$ such that $\mathbb{R} \cap \mathbb{P}^*$ yields $\mathbb{R}$ which is a Cartesian Product because of Definition 3.2.

- (2)→(1) Assume that the underlying PCP instance has a solution w, and assume that $\overline{\mathbb{R}} \cup \mathbb{P}^*$ is a Cartesian Product. The word w^2 is also a solution of the underlying PCP instance. We see that $\alpha(w) \neq \alpha(w^2)$. Hence, assertion (1) of Lemma 3.5 shows that $\binom{\gamma(w)}{\alpha(w^2)}$ and $\binom{\gamma(w^2)}{\alpha(w)}$ belong to $\mathbb{P}^*$, and thus to $\overline{\mathbb{R}} \cup \mathbb{P}^*$. Because $\overline{\mathbb{R}} \cup \mathbb{P}^*$ is a Cartesian Product, $\binom{\gamma(w)}{\alpha(w)}$ belongs also to $\overline{\mathbb{R}} \cup \mathbb{P}^*$. However, due to Definition 3.2 and Lemma 3.7, this trace does not belong to $\overline{\mathbb{R}} \cup \mathbb{P}^*$.

- (3)→(1) This is similar to the previous point. Again, assume a solution w of the underlying PCP instance, and assume that $\mathbb{R} \cap \mathbb{P}^*$ is a Cartesian Product. The traces $\binom{\gamma(w)}{\alpha(w^2)}$ and $\binom{\gamma(w^2)}{\alpha(w)}$ belong to $\mathbb{P}^*$ by Lemma 3.5 and to $\mathbb{R}$ by Definition 3.2, i.e., they belong to $\mathbb{R} \cap \mathbb{P}^*$. Because $\mathbb{R} \cap \mathbb{P}^*$ is a Cartesian Product, $\binom{\gamma(w)}{\alpha(w)}$ belongs also to $\mathbb{R} \cap \mathbb{P}^*$, and thus to $\mathbb{P}^*$. However, due to Lemma 3.7, this trace does not belong to $\mathbb{R} \cap \mathbb{P}^*$. □

We proceed similar to the previous section.

Corollary 5.2 There is no algorithm, whose input are two alphabets Σ_1 and Σ_2, and further two $\binom{\Sigma_1^*}{\Sigma_2^*}$-automata for languages K and L of $\binom{\Sigma_1^*}{\Sigma_2^*}$ which decides one of the following properties:

(1) $K \cup L^*$ is a Cartesian Product.

(2) $K \cap L^*$ is a Cartesian Product. □

Proof: As in the proof of Corollary 4.2, an algorithm which decides one of the properties can be used to decide the PCP. □

We generalize this Corollary to fixed Cartesian Products over free monoids over alphabets with at least two letters.

Theorem 5.3 Assume two alphabets Σ_1 and Σ_2, such that each of the alphabets contains at least two letters. There is no algorithm, whose input are two $\binom{\Sigma_1^*}{\Sigma_2^*}$-automata for languages K and L of $\binom{\Sigma_1^*}{\Sigma_2^*}$ which decides one of the following properties:

(1) $K \cup L^*$ is a Cartesian Product.

(2) $K \cap L^*$ is a Cartesian Product. □

Proof: At first, we show assertion (1). We assume there are two alphabets Σ_1 and Σ_2 which are at least doubletons and an algorithm which decides (1) in $\binom{\Sigma_1^*}{\Sigma_2^*}$. Then, we can construct an algorithm which decides (1) in Corollary 5.2, which is a contradiction.

The algorithm to decide (1) in Corollary 5.2 has two alphabets Γ_1 and Γ_2, and further two $\binom{\Gamma_1^*}{\Gamma_2^*}$-automata for languages K and L as input. We "code" the problem to $\binom{\Sigma_1^*}{\Sigma_2^*}$. This monoid has a C4-submonoid. Hence, we fix a canonical code h from $\binom{\Gamma_1^*}{\Gamma_2^*}$ to C4 as in Definition 2.15. We can regard h as a morphism from $\binom{\Gamma_1^*}{\Gamma_2^*}$ to $\binom{\Sigma_1^*}{\Sigma_2^*}$. The morphism h is injective. Hence, a language T in $\binom{\Gamma_1^*}{\Gamma_2^*}$ is a Cartesian Product iff $h(T)$ is a Cartesian Product. Consequently, $K \cup L^*$ is a Cartesian Product iff $h(K \cup L^*)$ is a Cartesian Product. Because h is injective, we have $h(K \cup L^*) = h(K) \cup h(L)^*$.

Then, we can decide whether $K \cup L^*$ is a Cartesian Product by constructing $\binom{\Sigma_1^*}{\Sigma_2^*}$-automata for $h(K)$ and $h(L)$ (Theorem 2.16 and 2.9) and using the assumed algorithm to decide whether $h(K) \cup h(L)^*$ is a Cartesian Product. The language $h(K) \cup h(L)^*$ is a Cartesian Product iff $K \cup L^*$ is a Cartesian Product.

We can show undecidability of (2) in exactly the same way. □

6 Conclusions and Future Goals

Now, we continue the discussion we have cut at the end of subsection 2.5. Let us discuss about Theorem 4.3. Opposed to the undecidability of property (1), we can decide whether it holds $K^* \subseteq L$. Given automata for K and L, we can construct a rational expression k for K by Theorem 2.12, and then, we check whether we have $L(k^*) \subseteq L$ by Theorem 2.14.

I consider the undecidability of universality of $K \cup L^*$ as the most strange assertion of Theorem 4.3. Given an automaton for some language L in some trace monoid $\mathbb{M}(\Sigma, I)$, we can trivially decide whether L^* yields the complete trace monoid: we have simply to check whether every letter of Σ occurs as a one letter trace in L. But, given two automata for languages K and L in a trace monoid $\mathbb{M}(\Sigma, I)$ which contains a C4, we cannot decide whether $K \cup L^*$ yields the complete monoid $\mathbb{M}(\Sigma, I)$, i.e., we cannot decide whether the traces missing in L^* are covered by K.

Let us review assertion (3) of Theorem 4.3. The problem to decide whether $K \cup L^*$ is recognizable is somehow between the recognizability problem and the star problem. Firstly, we can regard assertion (3) as a special case of the recognizability problem. Given two rational expressions k and l, can we decide whether the language of the rational expression $k \cup l^*$ is recognizable if we can presume recognizability of $L(k)$ and $L(l)$? Secondly, we can regard the star problem as a special case of the problem to decide whether $K \cup L^*$ is recognizable, namely if K is the empty set.

However, we should not regard the problem to decide recognizability of $K \cup L^*$ just as a slightly modified star problem. From assertion (2) of Theorem 4.3 we conclude the following statement: If we have three automata for languages K, L and M, we cannot decide whether $K \cup L^*$ equals M if the underlying trace monoid contains a C4-submonoid. Opposed to this statement, we have the following assertion for every trace monoid [8]: If we have two automata for languages L and M, we can decide whether L^* equals M. Consequently, we should regard the problem to decide recognizability of $K \cup L^*$ as a more difficult problem than the star problem.

Are the properties (1) to (4) in Theorem 4.3 decidable in trace monoids which do not contain a C4-submonoid? In free monoids, (3) and (4) are always true, while (1) and (2) are decidable due to classical results in automata theory. The properties (1) to (4) remain decidable in trace monoids without a P3-submonoid. Remember, we have Theorem 2.17 for these monoids. Assume a monoid $\mathbb{M}(\Sigma, I)$ without a P3-submonoid. Given two automata for languages K and L, we can construct rational expressions k and l for K and L by Theorem 2.12. We can decide (1) in the following way: we construct an automaton for $\mathbb{M}(\Sigma, I) \setminus K$ by Theorem 2.7, and we construct from l a rational expression for $\mathbb{M}(\Sigma, I) \setminus L^*$ by Theorem 2.17. Then, we can decide $K \subseteq L^*$ by deciding whether the complement of L^* is a subset of the complement of K by Theorem 2.14.

We can decide whether $K \cup L^*$ yields the complete monoid $\mathbb{M}(\Sigma, I)$ by constructing a rational expression r for the complement of $K \cup L^*$ and checking whether $L(r)$ yields the empty set. We check this by checking $L(r) \subseteq \emptyset$ using Theorem 2.14.

We can decide properties (3) and (4) by constructing two rational expressions for $K \cup L^*$ and $K \cap L^*$, resp., and using Theorem 2.17 to decide whether these rational expressions define recognizable languages.

I do not see straightforward proofs to determine decidability or undecidability of the properties (1) to (4) in P3. However, it should be possible to determine it by using techniques from [13].

We sum up the results in the following table:

	$L(r) \subseteq L$	$L^* \in \mathrm{REC}(\mathbb{M})$	$K \subseteq L^*$ $(K \cup L^*) = \mathbb{M}$ $(K \cup L^*) \in \mathrm{REC}(\mathbb{M})$ $(K \cap L^*) \in \mathrm{REC}(\mathbb{M})$	$L(r) = \mathbb{M}$ $L(r) \in \mathrm{REC}(\mathbb{M})$
Σ^*	D Kleene [19]	D Kleene [19]	D Kleene [19]	D Kleene [19]
no P3-sub.	D Mezei Wright [26]	D Sakarovitch [33]	D Sakarovitch [32, 33]	D Sakarovitch [32, 33]
P3	D Mezei Wright [26]	D Gastin and al [13]	?	U Ibarra [18, 33]
no C4-sub.	D Mezei Wright [26]	D Richomme [30, 31]	?	U Ibarra [18, 33]
C4	D Mezei Wright [26]	**?**	U present paper	U Fischer Rosenberg [11, 2]

The first line contains some decisions problems, while we substitute $\mathbb{M}$ by the monoids in the left column. The letters D and U abbreviate decidable and undecidable, respectively. K and L are recognizable languages for which we assume automata, while r is a rational expression. The decidability of $L(r) \subseteq L$ is due to Theorem 2.14. I tend to credit this to Mezei and Wright. We see the already mentioned two dimensional hierarchy of difficulties. The decision problems become harder from the left to the right. The monoids become more difficult from the first line down to the last line.

We should concern the following open questions: The most interesting questions are still the star problem and the FPPP for these trace monoids which contain a C4-submonoid. If one of the properties (3) or (4) would be decidable in C4, the star problem would obviously be decidable in C4. However, we cannot trivially generalize undecidability of (3) and (4) to the star problem. I think, it is less hopeful to search for an extension or adaptation of the ideas in this paper to obtain a proof for the undecidability of the star problem in C4.

There are further some questions of minor interest. Which properties in Theorem 4.3 are undecidable in P3, and further, in trace monoids containing a P3, but no C4?

We developed the ideas in this paper using infinite languages $\mathbb{R}$ and $\mathbb{P}$. Which properties in Theorem 4.3 become decidable if we additionally assume finiteness of K or L? For instance, if we demand that K is finite, properties (1) and (4) are decidable. Further, I think (2) is also decidable as long as K is finite. Property (3) with the restriction $K = \emptyset$ is exactly the star problem. RICHOMME remarked that property (3) restricted to finite sets K is decidable iff the star problem is decidable, because for finite sets K, we have $K \cup L^*$ is recognizable iff L^* is recognizable. The examination of property (3) in the case that both K and L are finite languages leads to the star problem for finite languages, which is open for languages containing more than four traces.

7 Acknowledgments

I acknowledge the discussions with my supervisor HEIKO VOGLER as well as with MANFRED DROSTE and DIETRICH KUSKE from the Institute of Algebra. I thank to GWÉNAËL RICHOMME for reading a preliminary version of this paper and making helpful remarks.

References

[1] I. J. Aalbersberg and E. Welzl. Trace languages defined by recognizable string languages. *R.A.I.R.O. - Informatique Théorique et Applications*, 20:103–119, 1986.

[2] J. Berstel. *Transductions and Contex-Free Languages*. B. G. Teubner, Stuttgart, 1979.

[3] A. Bertoni, G. Mauri, and N. Sabadini. Unambiguous regular trace languages. In G. Demetrovics et al., editors, *Proceedings of the Coll. on Algebra, Combinatorics and Logic in Computer Science*, volume 42 of *Colloquia Mathematica Soc. J. Bolyai*, pages 113–123. North Holland, Amsterdam, 1985.

[4] P. Cartier and D. Foata. *Problèmes combinatoires de commutation et réarrangements*, volume 85 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1969.

[5] M. Clerbout and M. Latteux. Semi-commutations. *Information and Computation*, 73:59–74, 1987.

[6] B. Courcelle. Basic notions of universal algebra for language theory and graph grammars. *Theoretical Computer Science*, 163:1–54, 1996.

[7] V. Diekert. *Combinatorics on Traces*, volume 454 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin Heidelberg New York, 1990.

[8] V. Diekert and Y. Métivier. Partial commutation and traces. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Vol. 3, Beyond Words*, pages 457–534. Springer-Verlag, Berlin Heidelberg, 1997.

[9] V. Diekert and G. Rozenberg, editors. *The Book of Traces*. World Scientific, Singapore, 1995.

[10] S. Eilenberg. *Automata, Languages, and Machines*, Volume A. Academic Press, New York, 1974.

[11] P. C. Fischer and A. L. Rosenberg. Multitape one-way nonwriting automata. *Journal of Computer and System Sciences*, 2:88–101, 1968.

[12] M. Fliess. Matrices de hankel. *J. Math. Pures et Appl.*, 53:197–224, 1974.

[13] P. Gastin, E. Ochmański, A. Petit, and B. Rozoy. Decidability of the star problem in $A^* \times \{b\}^*$. *Information Processing Letters*, 44:65–71, 1992.

[14] S. Ginsburg and E. Spanier. Bounded regular sets. In *Proceedings of the AMS*, volume 17:5, pages 1043–1049, 1966.

[15] S. Ginsburg and E. Spanier. Semigroups, presburger formulas and languages. *Pacific Journal of Mathematics*, 16:285–296, 1966.

[16] K. Hashiguchi. A decision procedure for the order of regular events. *Theoretical Computer Science*, 8:69–72, 1979.

[17] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Language, and Computation.* Addison-Wesley, Reading, 1979.

[18] O. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25:1:116–133, 1978.

[19] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. Shannon and J. McCarthy, editors, *Automata Studies, Annals of Math. Studies 34*, pages 3–40. Princeton, New Jersey, 1956.

[20] A. Mazurkiewicz. Introduction to trace theory. Chapter 1 in [9], pages 3-41.

[21] A. Mazurkiewicz. Concurrent program schemes and their interpretations. DAIMI Rep. PB 78, Aarhus University, 1977.

[22] J. D. McKnight. Kleene quotient theorem. *Pacific Journal of Mathematics*, 14:1343–1352, 1964.

[23] Y. Métivier. Une condition suffisante de reconnaissabilité dans un monoïde partiellement commutatif. *R.A.I.R.O. - Informatique Théorique et Applications*, 20:121–127, 1986.

[24] Y. Métivier and G. Richomme. On the star operation and the finite power property in free partially commutative monoids. In Patrice Enjalbert et al., editors, *STACS'94 Proceedings*, volume 775 of *Lecture Notes in Computer Science*, pages 341–352. Springer-Verlag, Berlin, 1994.

[25] Y. Métivier and G. Richomme. New results on the star problem in trace monoids. *Information and Computation*, 119(2):240–251, 1995.

[26] J. Mezei and J. B. Wright. Algebraic automata and context-free sets. *Information and Control*, 11:3–29, 1967.

[27] E. Ochmański. Recognizable trace languages. Chapter 6 in [9], pages 167-204.

[28] E. Ochmański. *Regular Trace Languages (in Polish).* PhD thesis, Warszawa, 1984.

[29] E. Post. A variant of a recursively unsolvable problem. *Bulletin of Amer. Math. Soc.*, 52:264–268, 1946.

[30] G. Richomme. Some trace monoids where both the star problem and the finite power property problem are decidable. Internal report 755-93, LaBRI - Université Bordeaux I, 1993.

[31] G. Richomme. Some trace monoids where both the star problem and the finite power property problem are decidable. In I. Privara et al., editors, *MFCS'94 Proceedings*, volume 841 of *Lecture Notes in Computer Science*, pages 577–586. Springer-Verlag, Berlin, 1994.

[32] J. Sakarovitch. On regular trace languages. *Theoretical Computer Science*, 52:59–75, 1987.

[33] J. Sakarovitch. The "last" decision problem for rational trace languages. In I. Simon, editor, *Proceedings of the 1st Latin American Symposium on Theoretical Computer Science*, volume 583 of *Lecture Notes in Computer Science*, pages 460–473. Springer-Verlag, Berlin Heidelberg New York, 1992.

[34] I. Simon. Limited subsets of a free monoid. In *Proceedings of the 19th IEEE Annual Symposium on Foundations of Computer Science*, pages 143–150. North Carolina Press, 1978.

[35] S. Yu. Regular languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages, Vol. 1, Word, Language, Grammar*, pages 41–110. Springer-Verlag, Berlin Heidelberg, 1997.