Master Operations Research and Combinatorial Optimization
Master Informatique / Master Mathématiques & Applications

Square-free arithmetic progressions in infinite square-free words

Thomas Delépine

June 16, 2025

Research project performed at LIRMM

Under the supervision of:
Pascal Ochem and Matthieu Rosenfeld

Defended before a jury composed of:

Nadia	BRAUNER	Professor	UGA, G-SCOP
Van-Dat	CUNG	Professor	UGA, G-SCOP
Louis	ESPERET	Senior CNRS Researcher	UGA, G-SCOP

June 2025

Abstract

A word is a sequence $w = w_0w_1w_2\dots$ over some finite alphabet Σ. A word w contains a square uu if there exists words p, s such that $w = puus$ and if a word does not contain any square then it is square-free. In 1906, Thue proved that there exist infinite square-free words over alphabets of size 3 (ternary words), thus initiating the domain called combinatorics on words.

Since then, many variants of this problem have been studied. One such variant consists in considering the subsequences of a word. The subsequence modulo p of w is $w[p] = w_0w_pw_{2p}\dots$. In 2019, Harju proved in [18] that for every integer $p \geq 3$, there exists an infinite ternary square-free word whose subsequence modulo p is square-free. The same year, Currie, Rampersad, Harju, and Ochem proved in [8] that there exists an infinite ternary square-free word such that both its subsequences modulo 3 and 11 were square-free. They also gave a positive answer for the pair $(5,6)$. In this work, we prove that for a pair of relatively prime integers (p,q), as soon as either p or q is large, there exists an infinite ternary square-free word whose subsequences modulo p and q are both square-free, thus reducing the number of unknown pairs from almost all pairs to a finite number where both p and q are small. Through our work, we provide parameterized constructions proving that the languages of words that are square-free, square-free modulo p, and square-free modulo q are exponential. Finally, we explore the algorithmic problem of deciding whether a transducer is square-free. We give some related undecidability and hardness results.

Résumé

un mot est une suite $m = m_0m_1m_2\dots$ sur un alphabet fini Σ. Un mot m contient un carré uu s'il existe des mots p et s tels que $m = puus$ et si un mot ne contient pas de carré alors il est sans-carré. En 1906, Thue prouva qu'il existe des mots infinis sans-carré sur un alphabet de taille 3 (des mots ternaires), lançant alors le domaine de la combinatoire des mots.

Depuis, beaucoup de variantes de ce problème ont été étudiées. Un exemple de tel variante consiste à considérer les sous-séquences des mots. La sous-séquence modulo p de m est $m[p] = m_0m_pm_{2p}\dots$. En 2019, Harju prouva que pour tout entier $p \geq 3$, il existe des mots infinis ternaires sans-carré dont la sous-séquence modulo p est elle aussi sans-carré. La même année, Currie, Rampersad, Harju and Ochem prouvèrent qu'il existe des mots infinis ternaires sans-carré et dont les sous-séquence modulo 3 et 11 sont sans-carrés. Ils montrèrent également la propriété pour la paire d'entiers $(5,6)$. Dans ce travail nous prouvons que pour une paire d'entiers premiers en eux (p,q), dès que p ou q est large, il existe un mot infini ternaire et sans-carré dont les sous-séquences modulo p et q sont sans-carré, réduisant alors le nombre de telles paires inconnues de presque toutes, à un nombre fini où p et q sont tous les deux petits. Nous donnons aussi des constructions paramétrées prouvant au passage que les langages des mots sans-carré, sans-carré modulo p, et sans-carré modulo q sont exponentiels. Enfin, nous explorons le problème algorithmique consistant à décider si un transducteur est sans-carré ou non. Nous donnons des résultats d'indécidabilité et de difficulté sur des problèmes similaires.

Contents

Abstract **i**

Résumé **i**

1 Introduction **1**
1.1 About words and languages 1
1.2 Pattern avoidance 2
1.3 Subsequences 4

2 Large p and large q **6**
2.1 Necessary and sufficient condition for large p and q 7
2.2 Pairs that respect the conditions 12

3 Large q and small p **14**
3.1 Large q and small $p \neq 6$ 14
3.2 Large q and $p = 6$ 15
3.3 Some of the remaining cases 19
3.4 Growth-rate of the languages 21

4 Square-free transducers **23**
4.1 A certificate for square-freeness 24
4.2 Square-free Turing Machines 26
4.3 PCP and SQUARE-FREE PCP 30
4.4 Undecidable transducers problems 32
4.5 Bounded SQUARE-FREE 33

5 Conclusion **35**

Appendix **38**

Bibliography **39**

— 1 —
Introduction

A *word* w is a (possibly infinite) sequence over a finite set, usually denoted Σ. The set Σ is called an *alphabet* and its elements are *letters*. We denote by w_i the $(i+1)$-th element of the sequence w. A word can be visually represented by the *concatenation* of its elements, in which case we write $w = w_0 w_1 w_2 \dots w_n$. Sometimes, we also write $w = \prod_{i\geq 0}^{n} w_i$ where $\prod$ is considered to be a product respecting the natural ordering given by the indices. For instance, the word $w = \prod_{i=0}^{4} \lceil i/3 \rceil$ is the word 01112 over the alphabet $\Sigma = \{0,1,2\}$. We also consider a special word ε, called the *empty word*, which represents words containing no letters. These definitions can be naturally extended to infinite words with infinite products. The field of theoretical computer science and discrete mathematics studying words is called *combinatorics on words*. Words can be studied from many points of view (algebra, algorithmic, combinatorics). In what follows, we will mostly be interested in the combinatorial properties of words and many classical definitions and properties about words can be found in Lothaire's book *Combinatorics on Words* [21].

1.1 About words and languages

For any alphabet Σ, we denote by Σ^+ the set of finite non-empty words over Σ, by Σ^* the set $\Sigma^+ \cup \{\varepsilon\}$, and by Σ^ω the set of infinite words over Σ. Alphabets can be any finite sets of elements so we denote by Σ_k the canonical alphabet $\{0,\dots,k-1\}$ containing k elements. If w is a finite word, then the *length* of w, denoted $|w|$ is its number of elements. For instance, $|four| = 4$ and $|septini| = 7$. The *concatenation* of the words u and v, denoted uv is the word $u_0 \dots u_{|u|-1} v_0 \dots v_{|v|-1}$. We then define the power of words as $w^0 = \varepsilon$ and $w^k = w w^{k-1}$ for $k \geq 1$, so in our context, $1^3 = 111$. A non-empty word u is a *factor* of a word w if there exist (possibly empty) words p, and s such that $w = pus$. If p is the empty word, then u is a *prefix* of w, and if s is the empty word, then u is a *suffix* of w. For instance, *bamboo* is a prefix of *bamboozle* and *gag* is a suffix of *lollygag*.

A *morphism* is a particular kind of function from the set of words over an alphabet Σ to the set of words over an alphabet Σ'. More precisely, a morphism is uniquely determined by the images of the letters in Σ and the following formula

$$h(aw) = h(a)h(w) \text{ for every } a \in \Sigma, w \in \Sigma^+.$$

For instance, for $h(0) = a$, $h(1) = b$, $h(2) = ab$, we can compute $h(120) = baba$, $h(1^k) = a^k$ and $h(222\dots) = ababab\dots$. We say that a morphism h is *uniform* if there exists an integer l

such that for every letter $a \in \Sigma$, $|h(a)| = l$. For every integer $k \geq 1$, we denote $\pi_k : \Sigma_k \to \Sigma_k$ the morphism such that $\pi_k(0) = 1$, $\pi_k(1) = 2$, ..., $\pi_k(k-1) = 0$. Moreover, a uniform morphism $h : \Sigma_k \to \Sigma_k$ is *circular* whenever $h(1) = \pi_k(h(0))$, $h(2) = \pi_k(h(1))$, ..., $h(k-1) = \pi_k(h(k-2))$, $h(0) = \pi_k(h(k-1))$. For instance, the morphism given by $h(0) = 010$ and $h(1) = 101$ is circular. A morphism h over an alphabet Σ is called *right prolongable* if there exists $a \in \Sigma$, and $w \in \Sigma^+$ such that $h(a) = aw$. In this case, $h^{n+1}(a) = \prod_{i=1}^{n} h^i(a)$ so $\lim_{n\to\infty} h^n(a)$ exists, is denoted $h^\omega(a)$, and is called a *fixed-point of h*.

A *language* over an alphabet Σ is a set of words over Σ. Languages are one of the most central concepts in theoretical computer science. Indeed, one of the formalisms of computation is based on the notion of Turing Machines, an abstract machine defining a language, and introduced by Alan Turing in [30] in 1936 . Algorithmic problems are then reduced to questions of the type "Does a word w belong to a language $\mathscr{L}$ described by a Turing Machine ?". For more information about the theory of computations and its relations with languages, we refer the reader to Sipser's book *Introduction to the Theory of Computation* [27] and to Hopcroft, Motwani, and Ullman's book *Introduction to Automata Theory, Languages, and Computation* [19].

The central position of languages motivates the study of languages and families of languages in a more general context. One such family are the regular languages, the languages defined inductively as follows:

- The empty set $\{\}$ is regular, $\{\varepsilon\}$ is regular, and for every letter a, $\{a\}$ is regular.
- For every languages $\mathscr{L}$, $\mathscr{L}'$, the set $\mathscr{L} \cup \mathscr{L}' = \{w, w \in \mathscr{L} \text{ or } w \in \mathscr{L}'\}$ is regular.
- For every languages $\mathscr{L}$, $\mathscr{L}'$, the set $\mathscr{L} \cdot \mathscr{L}' = \{ww', w \in \mathscr{L} \text{ and } w' \in \mathscr{L}'\}$ is regular.
- For every languages $\mathscr{L}$, the set $\mathscr{L}^* = \bigcup_{w\in\mathscr{L}} \{w^k, k \geq 0\}$ is regular.

Languages are connected to some other central areas of theoretical computer sciences such as automata theory and logic with the following two astonishing theorems.

Theorem 1 (Kleene, 1951 [20])**.** *Regular languages are exactly the languages recognized by finite automata.*

Theorem 2 (Büchi, 1960 [5], Elgot, 1961 [13], Trakhtenbrot, 1962 [29])**.** *Regular languages are exactly the languages definable in monadic second order logic.*

1.2 Pattern avoidance

We say that a word w is a *square* if there exists a non-empty word u such that $w = uu$ in which case the *period* of w is $|u|$, and that it is *square-free* if none of its factors is a square. For instance, the word *mama* is a square, the word *mommy* contains a square, and the word *mother* is square-free. The longest square-free words of Σ_2 are of length 3, namely 010 and 101. In 1906, Thue proved a theorem that is often considered as the starting point of combinatorics on words.

Theorem 3 (Thue, 1906 [4, 28])**.** *There exists an infinite ternary square-free word.*

For instance, consider the morphism $h(0) = 012$, $h(1) = 02$, $h(2) = 1$. Its unique fixed-point $h^\omega(0) = 012021012102\ldots$ is called the variant of Thue-Morse, denoted *vtm* and it is well known that *vtm* is an infinite ternary square-free word. This particular word has been widely studied and used in the scope of combinatorics on words (see [1, 7, 8, 28]). We say that a morphism $h : \Sigma \to \Sigma'$ is *square-free* whenever for every square-free word w over Σ, $h(w)$ is square-free. Notice that if h is the morphism constructing *vtm* as defined above, then $h(010) = 01202012$ containing the square 2020, so surprisingly, even if the fixed point of h is square-free, h is not. Square-free morphisms are a central tool to construct square-free words with additional properties. Although they are powerful tools for the study of square-free words, square-free morphisms might be hard to recognize. In fact, Crochemore provided a remarkably simple and useful characterization of square-free morphisms.

Theorem 4 (Crochemore, 1982 [6]). *Let $h : \Sigma^* \to \Sigma^*$ be a morphism.*

1. *If $|\Sigma| = 3$ then h is square-free if and only if for every square-free word w over Σ of length 5, $h(w)$ is square-free.*

2. *If h is uniform then h is square-free if and only if for every square-free word w over Σ of length 3, $h(w)$ is square-free.*

Since then, there has been a lot of interest in pattern avoidance questions related to words. For an alphabet Σ and a set $\mathscr{F} \subseteq \Sigma^+$ of non-empty finite words over Σ, we define the language $\mathscr{L} = Forb(\mathscr{F}, \Sigma) = \{w \in \Sigma^*, \forall f \in \mathscr{F}, f \text{ is not a factor of } w\}$. Here, $\mathscr{F}$ is called the *set of forbidden factors*. For instance, with $\mathscr{F} = \{01, 10\}$, $Forb(\mathscr{F}, \Sigma_2) = \{0^k, k \geq 0\} \cup \{1^k, k \geq 0\}$. Let $\mathscr{S}$ be the set of squares over Σ_3. Theorem 3 states that $Forb(\mathscr{S}, \Sigma_3)$ contains arbitrarily large words. Given a set of forbidden factors $\mathscr{F}$, and an alphabet Σ, two natural questions arise for the language $\mathscr{L} = Forb(\mathscr{F}, \Sigma)$:

- Does $\mathscr{L}$ contain arbitrarily large words ? and if so,
- What is the asymptotic behaviour of the function $C_{\mathscr{L}} : n \mapsto |\mathscr{L} \cap \Sigma^n|$?

Here, $C_{\mathscr{L}}$ is called the *complexity* of the language $\mathscr{L}$. For ternary square-free words, it is known that $1.3017597 < \lim_{n\to\infty} C_{\mathscr{L}}(n)^{1/n} < 1.3017619$ (see [26]). We call a set of forbidden factors $\mathscr{F}$ *unavoidable* if $Forb(\mathscr{L}, \Sigma)$ contains no arbitrarily large words, that is, every large enough word over Σ must contain factors from $\mathscr{F}$.

A language $\mathscr{L}$ is *factorial* whenever for every $w \in \mathscr{L}$, every factor of w is in $\mathscr{L}$. If a word avoids $\mathscr{F}$, then in particular, every factor of this word avoids $\mathscr{F}$ as well. Therefore, for every set of forbidden factors $\mathscr{F}$, $Forb(\mathscr{F}, \Sigma)$ is factorial. For a factorial language $\mathscr{L}$, one can verify that the following holds:

$$\text{for every integer } m, n \geq 0, C_{\mathscr{L}}(m+n) \leq C_{\mathscr{L}}(m) C_{\mathscr{L}}(n).$$

A sequence with this property is said to be *sub-multiplicative*. Sub-multiplicative sequences are usually interesting to study thanks to the following powerful tool called Fekete's lemma.

Theorem 5 (Fekete's lemma). *Let $(u_n)_{n\geq 0}$ be a sub-multiplicative sequence. Then the following limit exists and*

$$\lim_{n\to\infty} u_n^{1/n} = \inf_{n\to\infty} u_n^{1/n}.$$

So, in particular, if this limit is $\gamma > 1$, then $C_{\mathscr{L}}(n) \geq \gamma^n$ and $C_{\mathscr{L}}(n) = f(n)\gamma^n$ for some sub-exponential function f. In this case, we say that the language $\mathscr{L}$ is *exponential* and γ is called the *growth-rate* of $\mathscr{L}$. Square-free words over an alphabet of size at least 3 are an example of exponential factorial languages since the growth-rate of the language of square-free words over Σ_3 is greater than 1.3.

In symbolic dynamics, pattern avoidance is studied in a similar setting. The central object of symbolic dynamics is called *sub-shift*. It is defined as a closed (in the topological sense) set over $\Sigma^{\mathbb{N}}$ (or more generally $\Sigma^{\mathbb{Z}}$) that is invariant under a *shift* operation. In fact, this object can also be defined as the set of infinite words over Σ avoiding some finite factors. The similarity of the notion of sub-shift and our notion of languages avoiding some factors allow many bridges between the two fields. For instance, the notion of *entropy* in symbolic dynamics is exactly the log of the growth-rate of languages avoiding some factors. We refer the reader to [3] and [15] for some introductory books on symbolic dynamics.

many variations on avoidability of squares have been studied such as abelian squares or additive squares. For a word w over an alphabet Σ, and for every $a \in \Sigma$, we denote by $|w|_a$ the number of a in w. We say that a word uv over Σ is an *abelian square* whenever for every $a \in \Sigma$, $|u|_a = |v|_a$. Abelian squares are not avoidable on alphabets of size 3 or less, but are avoidable on alphabets of size at least 4 (see [14] for a recent survey on the topic). Abelian squares generalize squares since if w is a square, then it is an abelian square. A classical method to generalize squares is the following. For an equivalence relation $\equiv$ (a binary relation that is associative, symmetric, and reflexive) over finite words, we define a $\equiv$-square as every words uv such that $u \equiv v$. For instance, consider Σ as a subset of $\mathbb{N}$ (and we consider addition of elements in Σ as the usual addition). We can then define $\equiv$ as $u \equiv v$ if and only if $\sum_{i=0}^{|u|-1} u_i = \sum_{i=0}^{|v|-1} v_i$. Such squares are called additive squares and an open problem of the domain is whether additive squares are avoidable over finite alphabet or not.

Square avoidance has also been studied in graph theory as graph colorings. We say that a coloring of a graph G is *non-repetitive* whenever for every path in G, the colors on the path construct a square-free word. The *non-repetitive chromatic number of a graph* is the smallest integer $\pi(G)$ such that there exists a non-repetitive coloring using $\pi(G)$ colors. Theorem 3 states that $\pi(P_n) = 3$ for every path graph on at least 4 vertices. This notion has been introduced in [2] and we refer the reader to [16] and [31] for surveys about non-repetitive colorings of graphs. The study of the non-repetitive colorings of graphs is an active area of research, and it has recently been proven that planar graphs have bounded non-repetitive chromatic number [12].

1.3 Subsequences

One variant of pattern avoidance languages considers ternary square-free words with additional constraints on their subsequences. For a word $w = w_0 w_1 w_2 \ldots$, we define the *subsequence modulo p of w*, denoted $w[p]$ as $w_0 w_p w_{2p} \ldots$. If a word w has the property that $w[p]$ is square-free, then we say that it is *square-free modulo p*. The largest possible ternary word that is square-free and square-free modulo 2 is 01201021012010 of length 14. We can then wonder what is the smallest possible integer p such that there exists an infinite ternary square-free word that is square-free modulo p.

Theorem 6 (Harju, 2019 [18]). *For every integer $p \geq 3$, there exists an infinite ternary word w that is square-free and square-free modulo p.*

At the end of his paper, Harju proposed the following problem:

Problem 7 (Harju, 2019 [18]). *Does there exist pairs of relatively prime integers p, q such that there exists an infinite ternary word that is square-free, square-free modulo p, and square-free modulo q ?*

This problem has been solved positively in 2019 by Currie, Rampersad, Harju and Ochem in [8] where they gave the pairs $(3, 11)$ and $(5, 6)$, so the problem that we will try to solve is

What are the good pairs ?

In order to answer this question, we first provide a construction of infinite ternary square-free words, square-free modulo q and square-free modulo p when p and q are large in Chapter 2 using some ideas from [8], thus providing the first infinite family of positive pairs. In Chapter 3, we complete our work by providing constructions when q is large and p is small, thus reducing the problem to a finite number of unknown pairs and proving that almost all pairs are good pairs. Through our constructions, we introduced the notion of square-free transducers, a very natural generalization of square-free morphisms. In Chapter 4, we study some complexity and decidability results related to the algorithmic question "is a transducer square-free ?".

— 2 —

Large p and large q

Intuitively, when p and q are large, enforcing that the subwords modulo p and q are square-free results in sparse local constraints, therefore infinite ternary square-free words, square-free modulo p and square-free modulo q should exist. In this chapter, we confirm this intuition.

We will use the following particularly useful morphism in our first constructions. We define h, a *multi-valued* circular morphism over $\Sigma_3 = \{0,1,2\}$ as follows:

$$\begin{aligned}
h_{23}(0) &= 012102120210\mathbf{12}021201210\\
h_{24}(0) &= 012102120210\mathbf{201}021201210\\
h_{25}(0) &= 012102120210\mathbf{2012}021201210\\
h_{26}(0) &= 012102120210\mathbf{20121}021201210
\end{aligned}$$

and for all $i \in \{23,24,25,26\}$, $h_i(1) = \pi(h_i(0))$ and $h_i(2) = \pi_3(h_i(1))$ where $\pi_3 : \Sigma_3 \to \Sigma_3$ is the morphism given by $\pi_3(0) = 1$, $\pi_3(1) = 2$, $\pi_3(2) = 0$. Given a word $w \in \Sigma_3^\omega$ and an infinite sequence $\Gamma = (\gamma_i)_{i\geq 0} \in \{23,24,25,26\}^\omega$, the *image of w by h of guiding sequence* Γ is given by

$$h_\Gamma(w) = h_{\gamma_0}(w_0)h_{\gamma_1}(w_1)h_{\gamma_2}(w_2)\dots$$

This morphism, introduced in a similar setting in [8], will be quite handy thanks to the following property.

Lemma 8 (Currie, Harju, Ochem, Rampersad, 2019 [8])**.** *For any infinite square-free word $w \in \Sigma_3^\omega$, and any guiding sequence Γ, the word $h_\Gamma(w)$ is square-free.*

This property is extremely useful because it gives us two ways to control the square-free word $h_\Gamma(w)$ by changing either w or Γ.

A *pattern* is a set of words over some alphabet Σ and we denote by

$$\mathscr{P} = \Sigma^{(0)} \diamond^{\delta_1 - 1} \Sigma^{(1)} \dots \Sigma^{(n-1)} \diamond^{\delta_n - 1} \Sigma^{(n)}$$

the pattern containing words of length $1 + \sum_{i=1}^{n} \delta_i$ over Σ such that $w \in \mathscr{P}$ if and only if for every integer $m \leq n$, $w_{\sum_{i=0}^{m} \delta_i} \in \Sigma^{(m)}$. For the sake of conciseness, in pattern notation, we denote $\Sigma^{(i)} = \{a\}$ by a, and $\Sigma^{(i)} = \Sigma - a$ by $\overline{a}$. We say that a word w *contains* the pattern $\mathscr{P}$ at position i, and that $\mathscr{P}$ *occurs* at position i in w whenever there exists $v \in \mathscr{P}$ such that $w_i \dots w_{i+|v|-1} = v$. For instance, $\mathscr{P} = 0 \diamond^2 0$ over $\Sigma^{(2)}$ is exactly $\{0000, 0010, 0100, 0110\}$, and the word 010010 contains the pattern $\mathscr{P}$ at position 0 and at position 2. A pattern $\mathscr{P}$ is (Δ, h_{26})-*recurrent* if

for every infinite ternary square-free word t, and for every integer $i \geq 0$, there exists j such that $h_{26}(t)$ contains $\mathscr{P}$ at position j with $i \leq j \leq i+\Delta$. For instance, if $\mathscr{P} = \{0\}$, then it is $(3, h_{26})$-recurrent since every factor of length 4 of a square-free word over Σ_3 must contain 0, 1, and 2.

2.1 Necessary and sufficient condition for large p and q

For an infinite ternary word w, we denote by $(*)$ the following condition :

$$\forall i \geq 0, \begin{cases} i \equiv 0 \pmod{p} \text{ and } i+1 \equiv 0 \pmod{q}, \\ \text{or} \\ i \equiv 0 \pmod{q} \text{ and } i+1 \equiv 0 \pmod{p} \end{cases} \Rightarrow w_i \neq w_{i+1}. \qquad (*)$$

Let $p, q \in \mathbb{N}$. If there exists an infinite ternary square-free word w that is square-free modulo p and square-free modulo q, then in particular it satisfies the condition $(*)$. In this section, we shall prove the following theorem stating that when p and q are large, a somewhat converse holds.

Theorem 9. *Let $p, q \in \mathbb{N}_{\geq 267}$ and let w be an infinite ternary (non-necessarily square-free) word satisfying condition $(*)$. Then there exists an infinite ternary square-free word w' such that, $w'[p] = w[p]$ and $w'[q] = w[q]$.*

So, in particular, whenever $w[p]$ and $w[q]$ are square-free, w' is square-free, square-free modulo p and square-free modulo q. The rest of this section is dedicated to the proof of Theorem 9. In Section 2.2, we will study for which pairs (p,q) of relatively prime numbers, there exists a word w that is square-free modulo p, square-free modulo q and that respects $(*)$. To prove Theorem 9, we start from $h_\Gamma(t)$ where t is any infinite ternary square-free word and Γ is the guiding sequence of constant value 26, and we will modify Γ so that $h_\Gamma(t)[p] = w[p]$ and $h_\Gamma(t)[q] = w[q]$. With this technique, the subwords modulo p and q are fixed and can be seen as additional constraints asking for some letters at some positions. Since p and q are large, these constraints are sparse, but sometimes, when the subsequences modulo p and q almost coincide, these constraints are actually patterns of shape $a \diamond^{\delta-1} b$ with δ small. In what follows, we will prove some lemmas that will help us deal with such pairs of constraints.

Lemma 10. *Let t be an infinite ternary square-free word, let $\Gamma = (\gamma_i)_{i\geq 0}$ be a guiding sequence, let $\mathscr{P}$ be a (Δ, h_{26})-recurrent pattern, and let $N \geq 0$. Then for every integer $N' \geq N + 4 + 26\lfloor \Delta/3 \rfloor$, there exists a guiding sequence Γ' such that for every integer $i \leq N$, $h_{\Gamma'}(t)_i = h_\Gamma(t)_i$ and P occurs at position N' in $h_{\Gamma'}(t)$.*

In the proof of this lemma, we use the fact that the multi-valued morphism h has images of length 23, 24, 25, 26 for each letter in Σ_3. The idea is that by taking $N' - N$ sufficiently large, we will be able to contract enough images of h_{26} into images of h_{23}, between position N and position N', to shift P towards the left at position N', as illustrated in Figure 2.1.

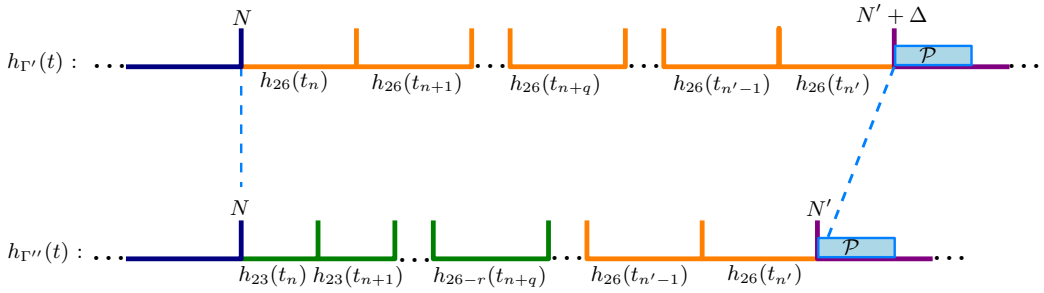


Figure 2.1: In order to have the pattern $\mathscr{P}$ at position N', we first find it at position $N'+\Delta$ and then we transform some images of h_{26} into images of h_{23}, h_{24} or h_{25} to shift $\mathscr{P}$ towards the left up to position N'.

Proof. Let n be the smallest integer such that $N \leq 11+\sum_{i=0}^{n-1}\gamma_i$. Observe that then $N+14 \geq \sum_{i=0}^{n-1}\gamma_i$. Indeed, by minimality of n,

$$\begin{aligned} N &> 11+\sum_{i=0}^{n-2}\gamma_i \\ &> 11-26+\sum_{i=0}^{n-1}\gamma_i \\ &\geq -14+\sum_{i=0}^{n-1}\gamma_i. \end{aligned}$$

We start by constructing a new guiding sequence $\Gamma'=(\gamma'_i)_{i\geq 0}$. For every integer $i \leq n-1$, $\gamma'_i=\gamma_i$ and for every integer $i \geq n$, $\gamma'_i=26$. Let Δ' be the smallest non-negative integer such that $h_{\Gamma'}(t)$ contains the pattern $\mathscr{P}$ at position $N'+\Delta'$, so by hypothesis, $\Delta' \leq \Delta$ since $\mathscr{P}$ is (Δ,h_{26})-recurrent. If $\Delta'=0$, then $h_{\Gamma'}(t)$ already contains $\mathscr{P}$ at position N' and we are done. Therefore, for the rest of the proof, we can assume that $\Delta' \geq 1$. Let q and r be the quotient and the rest in the euclidean division of Δ' by 3, so $\Delta'=3q+r$. We now construct Γ'' from Γ' such that $\gamma''_i=\gamma'_i$ except for $\gamma''_n=\cdots=\gamma''_{n+q-1}=23$ and $\gamma''_{n+q}=26-r$.

For every $i<\sum_{j=0}^{n-1}\gamma_j$, $h_{\Gamma''}(t)_i=h_{\Gamma}(t)_i$ since for every $i\leq n-1$, $\gamma''_i=\gamma_i$. Moreover, for every $i\in\{\sum_{j=0}^{n-1}\gamma_j,\ldots,N\}$, $h_{\Gamma''}(t)_i=h_{\Gamma}(t)_i$ since $N\leq 11+\sum_{j=0}^{n-1}\gamma_j$, and every image of h shares a common prefix of length 12. Observe that when constructing Γ'' from Γ', we have only changed the values of $\gamma'_n,\ldots,\gamma'_{n+\lceil\Delta'/3\rceil-1}$ since when $\Delta'\equiv 0 \pmod 3$, $\gamma''_{n+q}=26=\gamma'_{n+q}$. We thus have,

$$\begin{aligned} 16+\sum_{i=0}^{n+\lceil\Delta'/3\rceil-2}\gamma'_i &= 16+\sum_{i=0}^{n-1}\gamma'_i+\sum_{i=n}^{n+\lceil\Delta'/3\rceil-2}\gamma'_i \\ &\leq 16+N+14+26(\lceil\Delta'/3\rceil-1) \\ &= 4+N+26\lceil\Delta'/3\rceil \\ &\leq 4+N'-(4+26\lceil\Delta/3\rceil)+26\lceil\Delta'/3\rceil \\ &\qquad \text{since by hypothesis, } N\leq N'-(4+26\lceil\Delta/3\rceil) \\ &\leq N' \text{ since } \Delta'\leq\Delta. \end{aligned}$$

So, in particular, $N'+\Delta' \geq 17+\sum_{i=0}^{n+\lceil \Delta'/3 \rceil -2} \gamma'_i$. s Every image of h shares a common suffix of length $9 = 26-17$, so for every integer $i \geq N'+\Delta'$, $i \geq 17+\sum_{j=0}^{n+\lceil \Delta'/3 \rceil -2} \gamma'_j$ therefore $h_{\Gamma''}(t)_{i-\Delta'} = h_{\Gamma'}(t)_i$ so $h_{\Gamma''}(t)$ contains the pattern $\mathscr{P}$ at position N' as desired. □

The following lemma gives some recurrent pattern that we will need to use later together with Lemma 10.

Lemma 11. *Let $\delta \in \{2,3,5,6,7,8,9\}$. For every letter a, $b \in \Sigma_3$, the pattern $a \diamond^{\delta-1} b$ is $(27, h_{26})$-recurrent. Moreover, for every letter a, $b \in \Sigma_3$ such that $a \neq b$, the pattern ab is $(13, h_{26})$-recurrent.*

Let $\mathscr{S}$ be the set of all factors of length 200 that can be obtained in a square-free word generated by h_{26}. By generating $\mathscr{S}$, one can verify using a computer program that for every pattern $a \diamond^{\delta-1} b$, every factor from $\mathscr{S}$ contains at least two occurrences of this pattern and the largest distance between two consecutive occurrences is 28, hence they are $(27, h_{26})$-recurrent. This script performing this verification can be founded in `https://github.com/ThomasDelepine/SquareFreeWords/tree/main`.

Consider the pattern $a \diamond^{25} a$. Since h_{26} is 26-uniform and circular, this pattern never appears in any image of h_{26}. Although, some other patterns are (Δ, h_{26})-recurrent but some Δ very large. To solve the problem a pattern being avoided, and to improve the bounds that would be worse due to sporadic patterns, we propose the following generalization of the notion of (Δ, h_{26})-recurrence and of Lemma 10. A pattern $\mathscr{P}$ is (Δ, h)-*constructible* if there exists $k \in \mathbb{N}$ such that for every square free word w of length $k+1$, there exists a guiding sequence Γ of length $k+1$ such that $h_\Gamma(w)$ contains $\mathscr{P}$ at position at most $\Delta \leq \gamma_0 - 1$. Notice that every (Δ, h_{26})-recurrent pattern is (Δ, h)-constructible. Intuitively, we want $\mathscr{P}$ to start in the first block of $h_\Gamma(w)$ and thus are interested in (Δ, h)-constructible as small as possible. This definition leads to the following lemma:

Lemma 12. *Let t be an infinite ternary square-free word and let $\Gamma = (\gamma_i)_{i\geq 0}$ be a guiding sequence. For every integer $N \geq 0$, $N' \geq N+238$, and for every (Δ, h)-constructible pattern $\mathscr{P}$ with $\Delta \leq 25$, there exists a guiding sequence Γ' such that for every integer $i \leq N$, $h_\Gamma(t)_i = h_{\Gamma'}(t)_i$ and $h_{\Gamma'}(t)$ contains the P at position N'.*

In the proof of Lemma 12, we proceed as for Lemma 10 except that we perform an additional step to construct the pattern.

Proof. Let n be the smallest integer such that $N \leq \sum_{i=0}^{n-1} \gamma_i$, so $N+14 \geq \sum_{i=0}^{n-1} \gamma_i$. We start by constructing a new guiding sequence $\Gamma' = (\gamma'_i)_{i\geq 0}$ such that for every integer $i \leq n-1$, $\gamma'_i = \gamma_i$ and for every integer $i \geq n$, $\gamma'_i = 26$. Let n' be the smallest integer such that $N' \leq \Delta + \sum_{i=0}^{n'} \gamma'_i$, so, since by hypothesis $\Delta \leq 25$, $N'+25-\Delta \geq \sum_{i=0}^{n'} \gamma'_i$. Since $\mathscr{P}$ is (Δ, h)-constructible, there exists a finite guiding sequence $\Phi = (\varphi_i)_{i\geq 0}$ of length $k+1$ such that $h_\Phi(t_{n'} \dots t_{n'+k})$ contains $\mathscr{P}$ at position $\Delta' \leq \Delta$. Therefore, we can define a new guiding sequence $\Gamma'' = (\gamma''_i)_{i\geq 0}$ from Γ' by setting $\gamma''_i = \gamma'_i$ for every $i \geq 0$ except for $i \in \{0, \dots, k\}$ where $\gamma''_{n'+i} = \varphi_i$. Hencec, $h_{\Gamma''}(t)$

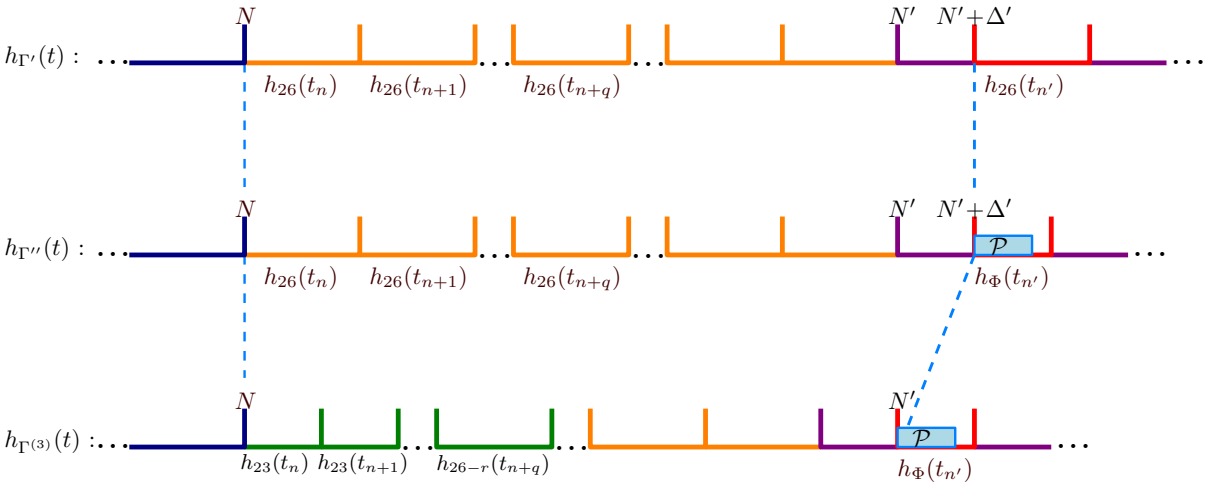


Figure 2.2: In order to have the pattern $\mathscr{P}$ at position N', we first construct it at position $N'+\Delta'$ and then we transform some images of h_{26} into images of h_{23}, h_{24} or h_{25} to shift $\mathscr{P}$ towards the left up to position N'.

contains $\mathscr{P}$ at position

$$\begin{aligned}\Delta' + \sum_{i=0}^{n'-1} \gamma''_i &= \Delta' + \sum_{i=0}^{n'-1} \gamma'_i \text{ since for all } i \leq n'-1, \gamma''_i = \gamma'_i \\ &\leq \Delta' + N' + 25 - \Delta \text{ since } \sum_{i=0}^{n'-1} \gamma'_i \leq N' + 25 - \Delta \\ &\leq N' + 25.\end{aligned}$$

Let Δ'' be the smallest integer such that $h_{\Gamma''}(t)$ contains $\mathscr{P}$ at position $N'+\Delta''$. By the previous observation, $\Delta'' \leq 25$. If $\Delta'' = 0$, then $h''_\Gamma(t)$ already contains $\mathscr{P}$ at position N' as desired, so we can assume that $\Delta'' \geq 1$. Let q and r be the quotient and the rest in the euclidean division of Δ'' by 3, so $\Delta'' = 3q+r$. We now construct $\Gamma^{(3)} = (\gamma_i^{(3)})_{i\geq 0}$ from Γ'' by setting $\gamma_i^{(3)} = \gamma''_i$ except for $\gamma_n^{(3)} \ldots, \gamma_{n+q-1}^{(3)} = 23$ and $\gamma_{n+q}^{(3)} = 26-r$. For every $i \leq \sum_{j=0}^{n-1} \gamma_j$, $h_{\Gamma^{(3)}}(t)_i = h_\Gamma(t)_i$ since $N \leq 11 + \sum_{j=0}^{n-1} \gamma_j$ and every image of h shares a common prefix of length 12. Observe that when constructing $\Gamma^{(3)}$ from Γ'', we have only changed the values of $\gamma''_n, \ldots, \gamma''_{n+\lceil \Delta''/3 \rceil - 1}$ since when $\Delta'' \equiv 0 \pmod 3$, $\gamma_{n+q}^{(3)} = 26 = \gamma''_{n+q}$. We thus have

$$\begin{aligned}16 + \sum_{i=0}^{n+\lceil \Delta''/3 \rceil - 2} \gamma''_i &= 16 + \sum_{i=0}^{n-1} \gamma''_i + \sum_{i=n}^{n+\lceil \Delta''/3 \rceil - 2} \gamma''_i \\ &\leq 16 + N + 26(\lceil \Delta''/26 - 1) \\ &= 4 + N + 26\lceil \Delta''/3 \rceil \\ &\leq 4 + N' - (4 + 26\lceil 25/3 \rceil) + 26\lceil \Delta''/3 \rceil \\ &\qquad \text{since by hypothesis, } N \leq N' - (4 + 26\lceil 25/3 \rceil) \\ &\leq N' \text{ since } \Delta'' \leq 25.\end{aligned}$$

So, in particular, $N'+\Delta'' \geq 17+\sum_{i=0}^{n+\lceil \Delta/3 \rceil -2} \gamma''_i$. Every image of h shares a common suffix of length $9 = 26-7$, so for every integer $i \geq N'+\Delta$, $i \geq 17+\sum_{i=0}^{n+\lceil \Delta/3 \rceil -2} \gamma''_i$ therefore $h_{\Gamma^{(3)}}(t)_{i-\Delta''} = h_{\Gamma''}(t)_i$ so $h_{\Gamma^{(3)}}(t)$ contains the pattern $\mathscr{P}$ at position N' as desired. □

Lemma 13. *For every a, $b \in \Sigma_3$, and for every integer $\delta \geq 8$,*

- *the pattern $a \diamond^3 b$ is $(11,4)$-constructible, and*
- *the pattern $a \diamond^{\delta-1} b$ is $(11,4)$-constructible.*

Proof. For every letter a, $b \in \Sigma_3$, it is simple to see that

$$h_{23}(0) = 01210212021012021201210$$

contains the pattern $a \diamond^3 b$ at position at most 11. Since h_{23} is a circular morphism, then so does $h_{23}(1)$ and $h_{23}(2)$.

Let $\delta \geq 8$. Observe that by definition of h, for every $\gamma \in \{23,24,25,26\}$, $h_\gamma(0)$ contains a at position 9, 10, or 11. Moreover, since h is a circular morphism, then so does $h_\gamma(1)$ and $h_\gamma(2)$. Let $k \geq 0$ be such that $11+\delta+3 \leq 26k+25$, and let $\Gamma = (\gamma_i)_{0 \leq i \leq k-1}$ and for every $i \leq k-1$, $\gamma_i = 26$. Then by the choice of k, and since every letter is $(3,h_{26})$-recurrent, for every square-free word w of length k, $h_\Gamma(w)$ contains the pattern $a \diamond^{\delta+\delta'-1} b$ at position $9+\delta''$ for some integers $\delta' \leq 3$ and $\delta'' \leq 2$. Since $9+\delta \geq 9+8 = 17$, by setting $\Gamma' = (\gamma'_i)_{0 \leq i \leq k-1}$ such that $\gamma'_i = \gamma_i$ except for $\gamma'_0 = 26-\delta'$, then $h_{\Gamma'}(w)$ contains the pattern $a \diamond^{\delta-1} b$ at position $9+\delta'' \leq 11$ as desired. □

We are now ready to prove Theorem 9 :

Proof of Theorem 9. Let w be an infinite ternary word respecting condition $(*)$. We want to construct an infinite ternary square-free word w' such that $w'[p] = w[p]$ and $w'[q] = w[q]$. Let $(u_i)_{i \geq 0}$ be the ordered sequence of the multiples of p or of q. So, for every $n \geq 0$, w'_{u_n} must be equal to w_{u_n}. We start by setting w' to be $h_{26}(t)$ for some infinite ternary square-free word t such that $w'_0 = w_0$. Therefore, there exists $n \geq 0$ such that for every integer $i \leq n$, $w'_{u_i} = w_{u_i}$ and the prefix of length u_n+1 of w' is square-free. We will show how to increase the value of n by only doing modifications on Γ so that w' remains square-free.

- If $u_{n+1}-u_n \geq 30$, then since the pattern $w_{u_{n+1}}$ is $(4,h_{26})$-recurrent, we can apply Lemma 10 with $P = w_{u_{n+1}}$, $\Delta = 3$, $N = u_n$, $N' = u_{n+1}$, and we have indeed that $N' \geq N+30$.
- Otherwise, if $u_{n+1}-u_n \leq 29$, then $u_n - u_{n-1} \geq 238$ since $u_{n+1}-u_{n-1} \geq \min(p,q) \geq 267$. Let $\delta = u_{n+1}-u_n$, we would like to find or construct the pattern $w_{u_n} \diamond^{\delta-1} w_{n+1}$ and shift it towards the left. If $\delta \in \{1,2,3,5,6,7,8\}$, then by Lemma 11, the pattern $w_{u_n} \diamond^{\delta-1} w_{n+1}$ is $(27,h_{26})$-recurrent so we can apply Lemma 10 with $\mathscr{P} = w_{u_n} \diamond^{\delta-1} w_{n+1}$, $\Delta = 27$, $N = u_{n-1}$, $N' = u_n$ and we indeed have that $N' \geq N+238$. Otherwise, $\delta = 4$ or $\delta \geq 9$. In both cases, by Lemma 13, the pattern $w_{u_n} \diamond^{\delta-1} w_{n+1}$ is $(11,h)$-constructible. Therefore, we can apply Lemma 12 with $\mathscr{P} = w_{u_n} \diamond^{\delta-1} w_{n+1}$, $N = u_{n-1}$, $N' = u_n$, and $\Delta = 11$.

In every previous case, by applying either Lemma 10 or Lemma 12, we managed to modify w' so that it remains an infinite ternary square-free word, and for every integer $i \leq n+1$, $w'_{u_i} = w_{u_i}$. □

In Theorem 9, there are no conditions on the subsequences modulo p and q besides condition $(*)$, so this result could be used to prove existence of words with subsequences having other constraints than being square-free.

2.2 Pairs that respect the conditions

In this section, we are interested in finding infinite ternary words that are square-free modulo p, square-free modulo q, and that respect condition $(*)$.

Theorem 14. *Let (p,q) be a pair of relatively prime numbers such that $p, q \geq 3$ and $\max(p,q) \geq 274$. Then there exists an infinite ternary word w that respects condition $(*)$ and such that $w[p]$ and $w[q]$ are square free.*

We first give some lemmas about the structure of the constraints for such words before proving Theorem 14.

Lemma 15. *Let (p,q) be a pair of relatively prime numbers such that $p,q \geq 3$. For every integer $i \geq 0$, there exists a, b*

- *$2a+b=q$, and*
- *the only multiples of p in $]ipq,(i+1)pq[$ that are congruent to ± 1 modulo q are $ipq+ap$, and $ipq+ap+bp$, and*
- *one of a and b is congruent to 1 modulo q and the other one is congruent to -1 modulo q.*

Proof. We first prove existence of a and b for fixed p and q. Since p and q are coprime, there exists an integer p^{-1} such that $0 < p^{-1} < q$ and $pp^{-1} \equiv 1 \pmod{q}$. Since $0 < pp^{-1} < pq$, $ipq < ipq + pp^{-1} < (i+1)pq$ and $ipq + pp^{-1} \equiv 1 \mod \pmod{q}$. Let $a = p^{-1}$. So, let $b = q - 2a$. We then have $ipq + pa + pb \equiv 1 + pb \equiv 1 + pq - 2pa \equiv -1 \pmod{q}$. If $ipq+pa+pb > (i+1)pq$, since $2a+b = pq$, $a+b \leq q$ so $ipq+pa+pb-pq \in]ipq,(i+1)pq[$ and $ipq+pa+pb-pq \equiv -1 \pmod{q}$ as desired.

We then prove uniqueness of a and b for fixed p and q. Let α, $\beta \in \mathbb{N}_{\geq 0}$ be such that $\alpha \leq \beta$, $\alpha p \in]ipq,(i+1)pq[$, $\alpha p \equiv 1 \mod q$, $\beta p \in]ipq,(i+1)pq[$ and $\beta p \equiv 1 \pmod{q}$. Then $(\alpha-\beta)p \equiv 0 \pmod{q}$ so since p and q are coprime, $\alpha-\beta$ is a multiple of q. If $\alpha-\beta > 0$, then $\alpha p + pq \leq \beta p$ but $\alpha p + pq \geq (i+1)pq$ so $\beta p \notin]ipq,(i+1)pq[$ which is absurd. Therefore, $\alpha = \beta$ and uniqueness holds for the case of a multiple of p in $]ipq,(i+1)pq[$ congruent to 1 modulo q. Similarly, we can prove that there is a unique multiple of p in $]ipq,(i+1)pq[$ congruent to -1 modulo q. □

In order to prove Theorem 14, we will need to find the pattern $\overline{a} \diamond^{\delta-1} \overline{b}$ or the pattern $\overline{a} \diamond^{\delta-1} b \diamond^{\delta-1} \overline{c}$ in a square-free word. In what follows, we will give results on their recurrence. We first define $\mathscr{P}_{bad}$, a set of bad patterns containing the following patterns, that will be treated separately.

$$\begin{array}{lll}
\overline{a}\diamond^{17-1} a \diamond^{17-1} \overline{a} & \overline{a}\diamond^{17-1} a \diamond^{17-1} \overline{a-1} & \overline{a}\diamond^{15-1} a \diamond^{15-1} \overline{a+1} \\
\overline{a}\diamond^{13-1} a \diamond^{13-1} \overline{a} & \overline{a}\diamond^{13-1} (a-1) \diamond^{13-1} \overline{a} & \overline{a}\diamond^{13-1} (a+1) \diamond^{13-1} \overline{a-1} \\
\overline{a}\diamond^{11-1} (a+1) \diamond^{11-1} \overline{a} & \overline{a}\diamond^{9-1} (a-1) \diamond^{9-1} \overline{a} & \overline{a}\diamond^{9-1} (a+1) \diamond^{9-1} \overline{a-1} \\
\overline{a}\diamond^{7-1} (a-1) \diamond^{7-1} \overline{a+1} & \overline{a}\diamond^{1-1} (a-1) \diamond^{1-1} \overline{a} & \overline{a}\diamond^{1-1} (a+1) \diamond^{1-1} \overline{a}
\end{array}$$

and that for every $a \in \Sigma_3$.

Lemma 16. *For every $\delta \in \{1,\ldots,18\}$ and for every letter a, b, $c \in \Sigma_3$, either $\mathscr{P} \in \mathscr{P}_{bad}$, or*

- *the pattern $\mathscr{P} = \overline{a} \diamond^{\delta-1} \overline{b}$ is $(6, h_{26})$-recurrent, and*

- *the pattern* $\mathscr{P} = \overline{a} \diamond^{\delta-1} b \diamond^{\delta-1} \overline{c}$ *is* $(27, h_{26})$*-recurrent*

Lemma 17. *Every pattern* $\mathscr{P} \in \mathscr{P}_{bad}$ *is* $(20,h)$*-constructible.*

Both Lemma 16 and Lemma 17 can be checked by a computer program, by performing a finite search on images of h. We are now ready to prove Theorem 14 :

Proof of Theorem 14. Without loss of generality, assume that $p \leq q$. Let us fix s, an infinite ternary square-free word, s will be $w[q]$. Since p and q are coprime, and $w[q]$ is fixed, there are some constraints on $w[p]$, where either a letter is forced (when $w[p]$ and $w[q]$ coincide in w), or a letter is forbidden (when $w[p]$ and $w[q]$ are at distance one in w). Let $(u_i)_{i\geq 0}$ be the ordered sequence of the positions of the constraints on $w[p]$ in $w[q]$ and let $(v_i)_{i\geq 0}$ be the ordered sequence of the positions of the constraints on $w[p]$ in $w[p]$. That is, if the i-th constraints on $w[p]$ is that $w[p]_j$ must be equal or different than $w[q]_{j'}$, then $u_i = j'$ and $v_i = j$.

By Lemma 15, there exist exactly two integers $a, b \geq 0$ such that $2a+b = q$ and for every integer $i \geq 0$, $v_{3i+1} - v_{3i} = a$, $v_{3i+2} - v_{3i+1} = b$, and $v_{3i+3} - v_{3i+2} = a$. In particular, $w[p]$ must be compatible with the partial word

$$w' = \prod_{i\geq 0} s_{u_{3i}} \diamond^{a-1} \overline{s_{u_{3i+1}}} \diamond^{b-1} \overline{s_{u_{3i+2}}} \diamond^{a-1} .$$

If $a \geq 18$ and $b \geq 18$, then by Theorem 19, there exists an infinite ternary square-free word w'' compatible with w'. So in this case we can just set $w[p] = w''$. Otherwise, let $\Gamma = (\gamma_i)_{i\geq 0}$ be the guiding sequence of constant value 26, and let t be an infinite ternary square-free word such that $h_\Gamma(t)_0 = s_0$. Therefore, there exists $n \geq 0$ such that the prefix of length $v_n + 1$ of $h_\Gamma(t)$ is compatible with w'. We will show how to increase the value of n by only doing modifications on Γ so that $h_\Gamma(t)$ remains square-free.

- If $a \leq 18$, then by definition of a and b, $b \geq q - 2a \geq 238$ since by hypothesis $q \geq 274$. Let $m \leq 2$ be such that $v_{n-m} - v_{n-m-1} = b$, and let $\mathscr{P} = s_{u_{n-m}} \diamond^{a-1} s_{u_{n-m+1}} \diamond^{a-1} s_{u_{n-m+2}}$. If $\mathscr{P} \in \mathscr{P}_{bad}$, then we can apply Lemma 12 with $N = v_{n-m-1}$, $N' = v_{n-m}$, $\mathscr{P}$, and $\Delta \leq 11$ by Lemma 17 since $\mathscr{P} \in \mathscr{P}_{bad}$ and we indeed have that $N' - N = b \geq 238$. Otherwise, $\mathscr{P}$ is $(27, h_{26})$-recurrent by Lemma 16 so we can apply Lemma 10 with $N = v_{n-m-1}$, $N' = v_{n-m}$, $\mathscr{P}$, and $\Delta \leq 27$ and we indeed have that $N' - N = b \geq 238$.

- If $b \leq 18$, then by definition of a and b, $a \geq (q-b)/2 \geq 128$ since by hypothesis, $q \geq 274$. If $v_{n+1} - v_n = a$, then we can apply Lemma 10 with $N = v_n$, $N' = v_{n+1}$, $\mathscr{P} = s_{u_{n+1}}$, and $\Delta = 3$ and we indeed have that $N' - N \geq 30$. Otherwise, $v_{n+1} - v_n = b$. We can thus apply Lemma 12 with $N = v_{n-1}$, $N' = v_n$, $\mathscr{P} = \overline{s_{u_n}} \diamond^{b-1} \overline{s_{u_{n+1}}}$, and $\Delta \leq 6$ since by Lemma 16, P is $(6,h)$-constructible, and we indeed have that $N' - N \geq 56$.

In all previous cases, by applying Lemma 10 or Lemma 12, we managed to construct a guiding sequence Γ' such that the prefix of length $v_{n+1} + 1$ of $h_{\Gamma'}(t)$ is compatible with w'. □

We are now ready to apply Theorem 9 together with Theorem 14 to deduce the main result of this Chapter.

Theorem 18. *For every integer* p, $q \geq 267$ *such that* $\max(p,q) \geq 274$*, there exists an infinite ternary square-free word that is square-free modulo* p *and* q*.*

— 3 —
Large q and small p

3.1 Large q and small $p \neq 6$

In Chapter 2, we proved that when p and q are coprime and large enough, there always exists an infinite ternary square-free word that is square-free modulo p and square-free modulo q. in this section, we will present another strategy that will allow us to deal with pairs with p small and q large (and symmetrically, with p large and q small). Our strategy is based on the existence of some morphisms and on the notion of compatible words.

A *partial word* over an alphabet Σ is a (possibly infinite) word over $\Sigma \cup \{\diamond\}$. For any partial word $v \in (\Sigma \cup \{\diamond\})^* \cup (\Sigma \cup \{\diamond\})^\omega$ and word $w \in \Sigma^* \cup \Sigma^\omega$, we say that w is *compatible* with v if $|w| \leq |v|$ and for every $i \leq |w|$, $v_i \neq \diamond \Rightarrow w_i = v_i$. For instance, the word *chocolate* is compatible with the word $c\diamond o\diamond\diamond\diamond at\diamond$ and is not compatible with the word $c\diamond\diamond a\diamond c\diamond ao\diamond$. A partial word can be seen as a set of forced letters at some positions in a square-free word. The following theorem states that if the constraints are at distance at least 19 from each other, then we can construct a square-free word fulfilling the constraints.

Theorem 19 ([24])**.** *Let $(p_i)_{i\geq 0} \in \mathbb{N}_{\geq 18}^{\mathbb{N}}$ and $(c_i)_{i\geq n} \in \Sigma^\omega$ be two sequences. For any partial word $v = \prod_{i\geq 0} c_i \diamond^{p_i}$, there exists an infinite ternary square-free word w compatible with v.*

We then define the shift of a word. The *shift* of a word $w = w_0 w_1 w_2 \ldots$, denoted by $s(w)$, is the word $w_1 w_2 \ldots$. If we consider a word w, then for every integer $\alpha \geq 0$, $s^\alpha(w)[p] = w_\alpha w_{\alpha+p} w_{\alpha+2p} \ldots$ and we call this word the *subsequence congruent to α modulo p of w*. The following proposition provides a sufficient condition on p and q, for an infinite ternary square-free word, square-free modulo p, square-free modulo q to exist.

Proposition 20. *Let $p \geq 0$. If there exists a morphism h such that*

- *h is circular,*
- *h is uniform with images of length kp for some $k \geq 1$,*
- *h is square-free, and*
- *there exists $\alpha \geq 0$ such that for every infinite ternary square-free word t, $s^\alpha(h(t))[p]$ is square-free,*

then for every integer $q \geq 20kp$, there exists an infinite ternary square-free word w such that $s^\alpha(w)[p]$ and $s^\alpha(w)[q]$ are square-free.

Proof. By hypothesis, for every infinite ternary square-free word w, $s^\alpha(h(w))$ and $s^\alpha(h(w))[p]$ are square-free. Let us fix t as any infinite ternary square-free words. We would like to find a word w such that $s^\alpha(h(w))[q] = t$ and doing so, $s^\alpha(h(w))$, $s^\alpha(h(w))[p]$ and $s^\alpha(h(w))[q]$ would be square-free. Since h is circular, for every letter $a \in \Sigma_3$ and for every integer $i < kp$, $a \in \{h(0)_i, h(1)_i, h(2)_i\}$. Let $i \geq 0$ and let d, r be the quotient and the rest in the euclidean division of iq by kp so that $iq = dkp + r$. Since t is fixed, $h(w)_{dkp+r}$ must be equal to t_i. Since h is circular, and by the previous observation, this constraint is actually a constraint on w_d. Let d', r' be the quotient and the rest in the euclidean division of $(i+1)q$ by kp. We have that $(i+1)p - ip = p = (d'-d)kp + r' - r \geq 20kp$ so $d' - d \geq 19$. Let $(u_i)_{i\geq 0}$ be the sequence of positions of w that have forced values. The previous observation implies that for every integer i, $u_{i+1} - u_i \geq 19$ therefore by Theorem 19, w exists. □

In order for us to apply Proposition 20 to prove the existence of the desired words, we need said morphisms. It appears that such morphisms exist for almost all integers for the cases $k = 1$ and $\alpha = 0$.

Proposition 21 (Currie, 2012 [7])**.** *For every integer $p \in \{13, 17, 18, 19\} \cup \mathbb{N}_{\geq 23}$, there exists a square-free circular morphism with images of length p.*

We can now apply Proposition 20 with the morphisms from Proposition 21 to obtain the following family of pairs of integers p, q for which there exists an infinite ternary square-free word that is square-free modulo p and q.

Corollary 22. *For all integers p, q such that $p \in \{13, 17, 18, 19\} \cup \mathbb{N}_{\geq 23}$ and $q \geq 20p$, there exists an infinite ternary square-free word w such that $w[p]$ and $w[q]$ are square-free.*

For the remaining cases except $p = 6$, we found morphisms with the desired properties. Said morphisms are presented in Table 5.1 which provides for every

$$p \in \{3, 4, 5, 7, 8, 9, 10, 11, 12, 14, 15, 16, 20, 21, 22\}$$

the morphism, the parameters k and α used in Proposition 20, and the upper-bound on the smallest value of q from which there exist infinite ternary square-free words that are square-free modulo p and square-free modulo q. This leads to the following Proposition.

Proposition 23. *For every $p \in \{3, 4, 5, 7, 8, 9, 10, 11, 12, 14, 15, 16, 20, 21, 22\}$, there exists l such that for every $q \geq l$, there exists an infinite ternary square-free word that is square-free modulo p and square-free modulo q.*

3.2 Large q and $p = 6$

This section is dedicated to the remaining case $p = 6$. Let R_{01}, R_{02}, R_{10}, R_{12}, R_{20}, R_{21}, be ternary words defined as follows :

- $R_{01} := \mathbf{0}12102$
- $R_{02} := \mathbf{0}12021$
- $R_{10} := \mathbf{1}20102$

- $R_{12} := \mathbf{1}20210$
- $R_{20} := \mathbf{2}01021$
- $R_{21} := \mathbf{2}01210$

In the following, addition is always taken modulo 3. Observe that for every i, j, $R_{ij} = iabcde$ where $a = i+1$, $b = a+1$, $c = d+1$, and $d = e+1$ and this observation, together with the fact that $R_{ij}j$ must be square free uniquely determines each of the R_{ij}. For every ternary square-free word t, we define the *completion* of t by $\mathscr{R}$ as

$$\mathscr{R}(t) = \prod_{i\geq 0}^{|t|-1} R_{w_i w_{i+1}}$$

and this definition naturally extends to infinite words. For instance, $\mathscr{R}(012) = \mathbf{0}12102\mathbf{1}20210$. The rule set $\mathscr{R}$ can be seen as a functional Finite State Transducer as shown in Figure 3.1. In Chapter 4, we will define transducers more formally and explore some of their properties. This notion of rule set is also closely related to the notion of two-blocks substitutions as studied in [10] for instance.

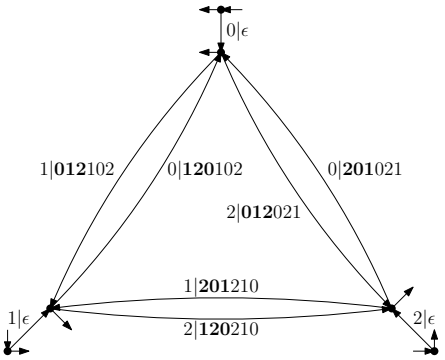


Figure 3.1: $\mathscr{R}$ represented as a transducer.

Notice that $\mathscr{R}(t)[6] = t'$ where t' is the prefix of size $|t|-1$ of t so since t is square-free, $\mathscr{R}(t)$ is square-free modulo 6. Proposition 24 also states that applying $\mathscr{R}$ to square-free words only produce square-free words, thus with $\mathscr{R}$, we can construct words that are square-free and square-free modulo 6.

Proposition 24. *For every (infinite) ternary square-free word t, $\mathscr{R}(t)$ is an (infinite) ternary square-free word.*

Proof. Let $w = \mathscr{R}(t)$ and assume that w contains a square uu with $|u| = \Delta$. Assume that $\Delta \geq 9$, so u must contain the factor abc with $b = a+1$ and $c = b+1$. We can assume without loss of generality that $abc = 012$ so that u contains 012 at position $|u| - \delta$. Let $t = t'a$ for some $a \in \Sigma_3$. As previously mentionned, $\mathscr{R}(t)[6] = t'$, so since t is square-free, then so is $\mathscr{R}(t)[6]$. If $\Delta \equiv 0 \pmod 6$, then there exists $i \in \{0,1,2,3,4,5\}$ such that the subsequence congruent to i modulo 6 of uu is a square and is a factor of $w[6]$ which is absurd since $w[6]$ is square-free. Therefore, we can assume that $\Delta \equiv \alpha \pmod 6$ with $0 < \alpha < 6$. observe that 012 only appears at position 0 of R_{01}, at position 0 of R_{02}, and at position 1 of R_{21} therefore, either $\alpha = 1$ or $\alpha = 5$. If $\alpha = 1$,

then if $\delta \geq 3$, then uu must contain 012102 or 012021 at position d and 012101 at position $d+\Delta$ which is absurd. Otherwise, $\delta \leq 2$ so uu must contain 0102012 at position $d-4$ and 2102012 at position $d-4+\Delta$ which is absurd. For the remaining case $\alpha = 5$, we can proceed similarly. If $\Delta \leq 8$, then if there exists a square in $\mathscr{R}(t)$, then this square must be a factor of $\mathscr{R}(t')$ where t' is a square-free word of length 4, so it is enough to see that the following words are square-free :

$$\mathbf{0}12102\mathbf{1}20102\mathbf{0}12021, \mathbf{0}12102\mathbf{1}20210\mathbf{2}01021, \mathbf{0}12102\mathbf{1}20210\mathbf{2}01210,$$
$$\mathbf{0}12021\mathbf{2}01021\mathbf{0}12102, \mathbf{0}12021\mathbf{2}01210\mathbf{1}20102, \mathbf{0}12021\mathbf{2}01210\mathbf{1}20210$$

and their images by π_3 and π_3^2 (where π_3 is the basic circular permutation of the alphabet as already used earlier). □

In order to construct words that are square-free, square-free modulo 6, and square-free modulo q, we would like to shift some patterns to fulfill constraints as previously. However, if we perform contractions on a word t, then we can only shift patterns in $\mathscr{R}(t)$ by distances that are multiple of 6. Therefore, we need some recurrence result on the subsequences that are congruent to α modulo 6 for $\alpha \in \{0,1,2,3,4,5\}$.

Lemma 25. *For every letter $a \in \Sigma_3$, for every infinite ternary square-free word t, for every integer $\alpha \leq 5$, and for every integer $i \geq 0$, there exists $\delta \leq 6$ such that $s^\alpha(\mathscr{R}(h_{23}(t)))[6]_{i+\delta} = a$.*

Proof. Let q and r be the quotient and the rest in the euclidean division of i by 23, so $i = 23q+r$. Since h is a circular morphism, we can assume without loss of generality that $t_q = 0$. It is enough to prove that for every $\alpha \leq 5$ and for every $i \leq 22$, there exists δ, $\delta' \leq 6$ such that $s^\alpha(\mathscr{R}(h_{23}(01)))[6]_{i+\delta} = a$ and $s^\alpha(\mathscr{R}(h_{23}(02)))[6]_{i+\delta'} = a$. By definition,

$$\begin{aligned}\mathscr{R}(h_{23}(0)) =& 0121021202102012101201020120212012101202102 0-\\ & 1021012021201210120102012102120210201021012 0-\\ & 2120121012021020102101210212021020121012 0102\end{aligned}$$

so in particular,

$$\begin{aligned}
s^0(\mathscr{R}(h_{23}(01)))[6] &= 012102120210120212012101202102010212010201202,\\
s^1(\mathscr{R}(h_{23}(01)))[6] &= 120210201021201020120212010210121020121012010,\\
s^2(\mathscr{R}(h_{23}(01)))[6] &= 201021012102012101201020121021202101202120121,\\
s^3(\mathscr{R}(h_{23}(01)))[6] &= 122102200211200220122112002100110220110012002,\\
s^4(\mathscr{R}(h_{23}(01)))[6] &= 011021122100122112011001221022002112002201221,\\
s^5(\mathscr{R}(h_{23}(01)))[6] &= 200210011022011001200220110211221001221120110,\\
s^0(\mathscr{R}(h_{23}(02)))[6] &= 012102120210120212012102010210121020121012010,\\
s^1(\mathscr{R}(h_{23}(02)))[6] &= 120210201021201020120212010210212021012021201 21,\\
s^2(\mathscr{R}(h_{23}(02)))[6] &= 201021012102012101201021202102010212010201202,\\
s^3(\mathscr{R}(h_{23}(02)))[6] &= 122102200211200220122100110211221001221120110,\\
s^4(\mathscr{R}(h_{23}(02)))[6] &= 011021122100122112011022002100110220110012002,\\
s^5(\mathscr{R}(h_{23}(02)))[6] &= 200210011022011001200211221022002112002201221,
\end{aligned}$$

and it is simple to see that they all have the desired property. □

Lemma 26. *Let t be an infinite ternary square-free word, and let $\Gamma = (\gamma_i)_{i\geq 0}$ be q guiding sequence. For every integer $N \geq 0$, $N' \geq N+63$, $\alpha \leq 5$, and for every letter $a \in \Sigma$, there exists a guiding sequence Γ' such that for every integer $i \leq N$, $h_{\Gamma'}(t)_i = h_\Gamma(t)_i$ and $\mathscr{R}(h_{\Gamma'}(t))_{6N'+\alpha} = a$.*

Proof. Let n be the smallest integer such that $N \leq \sum_{i=0}^{n-1} \gamma_i$, so $N+14 \geq \sum_{i=0}^{n-1} \gamma_i$. We start by constructing a new guiding sequence $\Gamma' = (\gamma'_i)_{i\geq 0}$ such that for every integer $i \leq n-1$, $\gamma'_i = \gamma_i$ and for every integer $i \geq n$, $\gamma'_i = 23$. By Lemma 25, there exists $\Delta \leq 6$ such that $\mathscr{R}(h_{\Gamma'}(t))_{6N'+\alpha-6\Delta} = a$. Let q and r be the quotient and the rest in the euclidean division of Δ by 3, so $\Delta = 3q+r$. We construct a new guiding sequence $\Gamma'' = (\gamma''_i)i \geq 0$ from Γ' by setting $\gamma''_n, \ldots, \gamma''_{n+q-1} = 26$ and $\gamma_{n+q} = 23+r$.

For every $i < \sum_{j=0}^{n-1} \gamma_j$, $h_{\Gamma''}(t)_i = h_\Gamma(t)_i$ since for every $i \leq n-1$, $\gamma''_i = \gamma_i$. Moreover, for every $i \in \{\sum_{j=0}^{n-1} \gamma''_i, \ldots, N\}$, $h_{\Gamma''}(t)_i = h_\Gamma(t)_i$ since $N \leq 11 + \sum_{j=0}^{n-1} \gamma_j$, and every image of h shares a common prefix of length 12. Observe that when constructing Γ'' from Γ', we have only changed the valued of $\gamma'_n, \ldots, \gamma'_{n+\lceil \Delta/3 \rceil - 1}$ since when $\Delta \equiv 0 \pmod 3$, $\gamma''_{n+q} = 23 = \gamma'_{n+q}$. We thus have

$$\begin{aligned}
23 + \sum_{i=0}^{n+\lceil \Delta/3 \rceil - 2} \gamma'_i &= 23 + \sum_{i=0}^{n-1} \gamma'_i + \sum_{i=2}^{n+\lceil \Delta/3 \rceil - 2} \gamma'_i \\
&\leq 23 + N + 14 + 26(\lceil \Delta/3 \rceil - 1) \\
&= 11 + N + 26\lceil \Delta/3 \rceil \\
&\leq 11 + N' - (11 + 26\lceil 6/3 \rceil) + 26\lceil \Delta/3 \rceil \\
&\qquad \text{since by hypothesis, } N \leq N - (11 + 26\lceil 6/3 \rceil) \\
&\leq N' \text{ since } \Delta \leq 6.
\end{aligned}$$

So, in particular, $N' - \Delta \geq 17 + \sum_{i=0}^{n+\lceil \Delta/3 \rceil - 2} \gamma'_i$. Every image of h shares a common suffix of length $9 = 26-17$, so for every integer $i \geq N' - \Delta$, $i \geq 7 + \sum_{i=0}^{n+\lceil \Delta/3 \rceil - 2} \gamma'_i$ therefore $h_{\Gamma''}(t)_{i+\Delta} = h_{\Gamma'}(t)_i$ so in particular, $\mathscr{R}(h_{\Gamma''}(t))_{6N'+\alpha} = a$ as desired. □

Proposition 27. *For every integer $q \geq 384$, there exists an infinite ternary square-free word that is square-free modulo 6 and square-free modulo q.*

Proof. Let s, t be any infinite ternary square-free word such that $s_0 = t_0$ and let $\Gamma = (\gamma_i)_{i\geq 0}$ be the guiding sequence of constant value 26. We will do modifications on Γ so that $\mathscr{R}(h_\Gamma(t))[q] = s$. By definition of s and t, there exists $n \geq 0$ such that n is a multiple of q and the subsequence modulo q of the prefix of length $n+1$ of $\mathscr{R}(h_\Gamma(t))$ is the prefix of length n/q of s. We will show how to increase n. Let d and r be the quotient and the rest in the euclidean division of $n+q$ and 6, so $n+q = 6d+r$. We can apply Lemma 26 with $N = \lceil n/6 \rceil$, $N' = d$, $\alpha = r$ and $a = s_d$ and since by hypothesis $q \geq 384$, we indeed have that $N' \geq N+63$:

$$\begin{aligned}
N' - N &= \lfloor (n+q)/6 \rfloor - \lceil N/6 \rceil \\
&\geq (n+q)/6 - 1 - n/6 \\
&= q/6 - 1 \\
&\geq 63 \text{ since } q \geq 384 = 6 \times 64.
\end{aligned}$$

□

We can then conclude this Section by combining Corollary 22, Proposition 23 and Proposition 27 to get the following Theorem.

Theorem 28. *For every integer $p \geq 3$, there exists $l \geq$ such that for every $q \geq l$, there exists an infinite ternary square-free word that is square-free modulo p and square-free modulo q.*

Notice that for every $p \geq 3$, Corollary 22, Proposition 23 or Proposition 27 provide an explicit value for l.

3.3 Some of the remaining cases

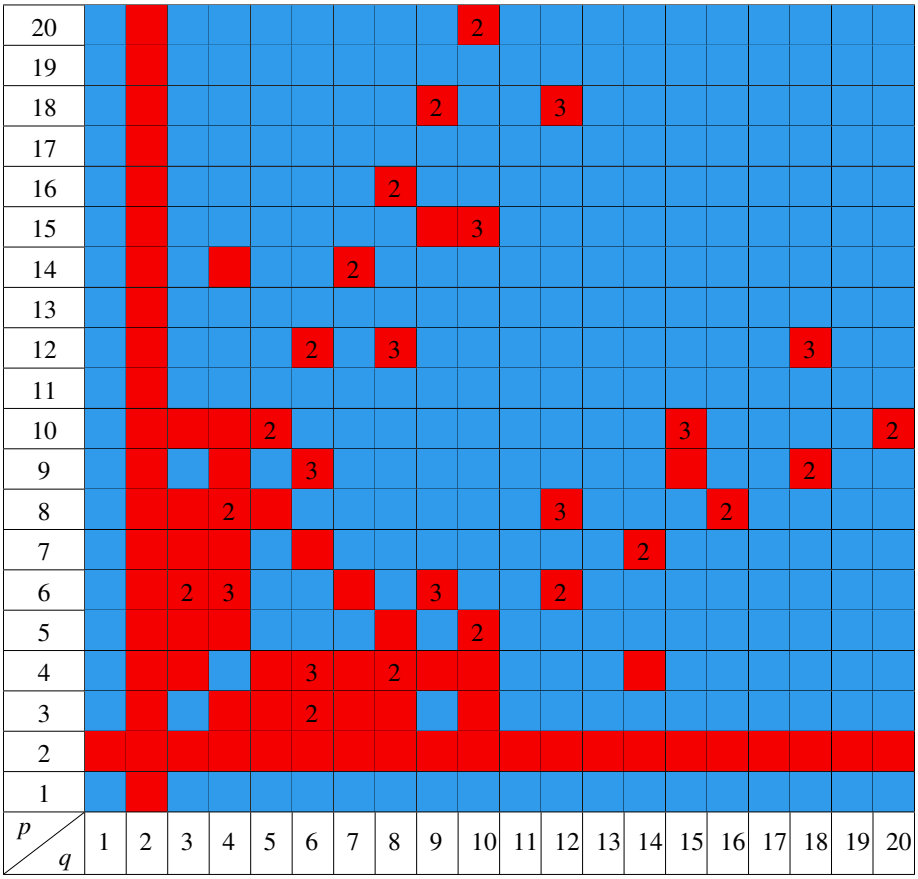


Table 3.1: Every blue cell is a positive solution and if $p \geq 3$ and $q \geq 3$, then we provide an explicit morphism constructing such words. A red cell means that no good word exists. A red cell with a 2 means it is on the line $(t, 2t)$ or $(2t, t)$ and a red cell with a 3 means it is on the line $(2t, 3t)$ or $(3t, 2t)$.

With Theorem 18 and Theorem 28, we only leave a finite uncovered space where both p and q are small. To start filling this finite space, for every positive pair (p,q) with both $p \leq 20$ and $q \leq 20$, we provide an explicit morphism at `https://www.lirmm.fr/~ochem/pairs.txt`. To prove that no such word exists, we can perform a backtracking algorithm that aims at creating the longest possible word fulfilling the constraints. If the search algorithm terminates, then no

such word exists. An implementation of this backtracking algorithm can be found at `https://github.com/ThomasDelepine/SquareFreeWords/tree/main`. We can also eliminate the pairs where $p = 2$ or $q = 2$ since there is no infinite ternary square-free word that is square-free modulo 2, and the pairs $(t, 2t)$, $(2t, t)$, $(2t, 3t)$, $(3t, 2t)$ since there is no infinite ternary (non-necessarily square-free) word that is square-free modulo 2 and square-free modulo 3. Both the negative pairs obtained from the backtracking algorithm and the positive pairs obtained with explicit morphisms are presented in Table 3.1. Moreover, Figure 3.2 is a sketch of the set of relatively prime integers that we managed to cover.

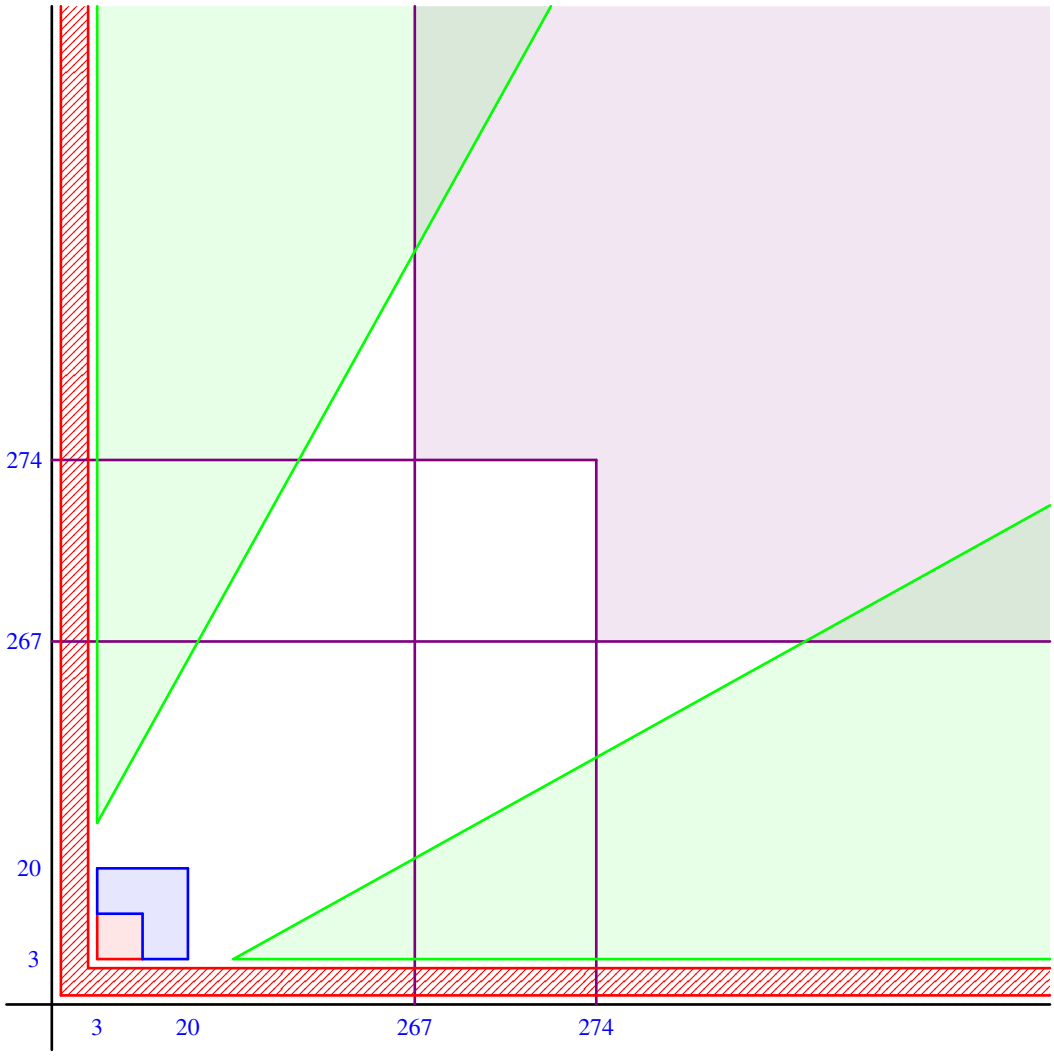


Figure 3.2: Sketch of the pairs of relatively prime integers covered by our constructions. The purple north-east quarter corresponds to the pairs covered by Theorem 18. The two green cones correspond to the pairs covered by Theorem 28. The red space corresponds to the locations of negative solutions and the blue space corresponds to Table 3.1.

3.4 Growth-rate of the languages

In both of our constructions we choose an infinite ternary square-free word to be the subsequence modulo q. It is well known that the language $\mathscr{L}$ of ternary square-free word is exponential in the sense that the complexity $C_{\mathscr{L}}(n) = f(n)\gamma^n$ for some sub-exponential function f and some constant γ. Moreover, it is known that $1.3017597 < \lim_{n\to\infty} C_{\mathscr{L}}(n)^{1/n} < 1.3017619$ as given in [26]. This means that if $\mathscr{L}_{p,q}$ is the language of ternary words that are square-free, square-free modulo p, and square-free modulo q, then, $C_{\mathscr{L}_{p,q}}(n) \geq (\gamma^{1/q})^n$ and so, for every pair (p,q) that we can cover with our constructions, $\mathscr{L}_{p,q}$ is an exponential language. However, this exponential lower bound is far from being tight. For instance, when $q \geq 100000$, $\gamma^{1/100000}$ is really close to 1 even if the constraints given by the subsequence modulo q are really sparse (and so we expect the growth-rate of $\mathscr{L}_{p,q}$ to be much closer to γ). In fact, the same problem is open even for the simplest case of $\mathscr{L}_3$, the language of ternary square-free words that are square-free modulo 3

The only two ternary words compatible with $0\diamond\diamond1$ are 0121 and 0201. Now consider the partial words $0\diamond\diamond1\diamond\diamond0\diamond\diamond2$ and $0\diamond\diamond1\diamond\diamond2$. To complete their prefixes of size 4, we have two possibilities each by the previous observation, namely $0121\diamond\diamond0\diamond\diamond2$, $0201\diamond\diamond0\diamond\diamond2$, $0121\diamond\diamond2$ and $0201\diamond\diamond2$. If we try to continue the completion of these partial words, we have

- $0121\diamond\diamond0\diamond\diamond2$ must be completed 0121020102.
- $0201\diamond\diamond0\diamond\diamond2$ can be either completed 0201210212 or $0201020\diamond\diamond2$ but $0201020\diamond\diamond2$ cannot be further completed.
- $0121\diamond\diamond2$ must be completed 0121012.
- $0201\diamond\diamond2$ must be completed 0201202.

Therefore, once we fixed the prefix of size 4 of a word $w = l_0\prod_{i=1}^{n}\diamond^2 l_i$ where $l = l_0\dots l_n$ is a ternary square-free word, there is exactly one compatible ternary square-free word of length $3n+1$. Since there are two possible prefixes of size 4, there exist exactly two ternary square-free word of length $3n+1$ compatible with w. In other words,

Proposition 29. *Let w be a ternary square-free word. There exist exactly two completions of $w_0\prod_{i=1}^{n}\diamond^2 w_i$ that are square-free.*

Corollary 30. $C_{\mathscr{L}_3}(n) = \Theta(\gamma^{n/3})$.

Also, a word is in $\mathscr{L}_3$ if and only if it is constructed by the square-free transducer presented in Figure 3.3.

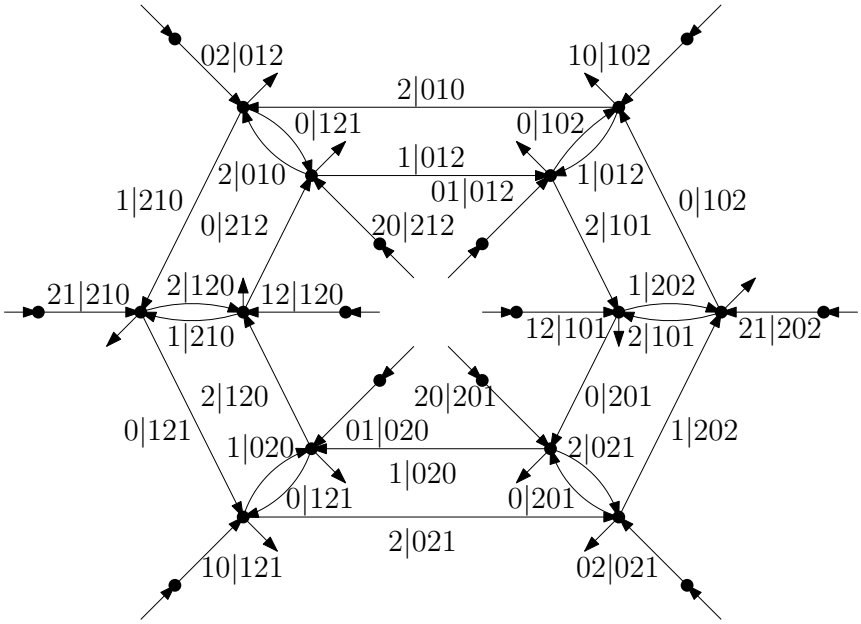


Figure 3.3: Transducer constructing words square-free and square-free modulo 3.

— 4 —

Square-free transducers

Automata are finite memory abstract machines that take a word in input and either accept or reject this word. This model can be generalized by allowing the abstract machines to produce outputs, thus leading to transducers. A *finite state transducer (FST)* is a 6-uplet $\mathscr{T} = (\Sigma, \Gamma, S, T, I, F)$ where

- Σ is the *input alphabet*,
- Γ is the *output alphabet*,
- S is a finite set (of *states*),
- $T \subseteq S \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \times S$ is the set of *transitions*,
- $I \subseteq S$ is the set of *initial states*,
- $F \subseteq S$ is the set of *final states*.

A *run* of a FST $\mathscr{T}$ is a sequence (s_0, i_0, o_0, s_1), (s_1, i_1, o_1, s_2), $\ldots$, $(s_{n-1}, i_{n-1}, o_{n-1}, s_n)$ such that

- for every $k \in \{1, \ldots, n\}$, $(s_{k-1}, i_{k-1}, o_{k-1}, s_k) \in T$, and
- $s_0 \in I$, and $s_n \in F$.

We say that the *input* of the run (s_0, i_0, o_0, s_1), $\ldots$, $(s_{n-1}, i_{n-1}, o_{n-1}, s_n)$ is $w = \prod_{k=0}^{n-1} i_k$ and that the *output* is $w' = \prod_{k=0}^{n-1} o_k$. Notice that with this definition, if we forget about the outputs, then we get the definition of automata. We denote by $\mathscr{T}(w)$ the (possibly empty) set of outputs of runs of input w (where w is a word over Σ). A FST can be visually represented in a graph fashion as follows :

- draw a vertex v_s per state $s \in S$, and
- draw a directed edge from v_s to $v_{s'}$, with label $i|o$ for every transition $(s, i, o, s') \in T$, and
- draw a directed edge entering v_s for every $s \in I$, and a directed edge leaving v_s for every $s \in F$.

For instance, consider the FST $\mathscr{T}$ represented in Figure 4.1 and defined as follows.

$$\mathscr{T} = (\{0,1\},\{0,1\},\{s_1,s_2,s_3\},\{(s_1,11,1,s_1),(s_2,11,00,s_2),(s_2,1,0,s_3)\},\{s_1,s_2\},\{s_1,s_3\})$$

For any integer $k \geq 0$, $\mathscr{T}(1^{2k}) = \{1^k\}$ and $\mathscr{T}(1^{2k+1}) = \{0^{2k+1}\}$ whereas $\mathscr{T}(010) = \{\}$.

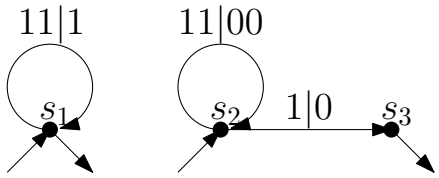


Figure 4.1: Example of the visual representation of $\mathscr{T}$

Here, we assume that every transition of our FSTs takes a single letter word as an input. This can be generalize by having the transition (s,i,o,s') where both i and o are finite words instead of the sequence of transitions $(s,i_0,\varepsilon,s_1)\dots(s_{n-2},i_{n-2},\varepsilon,s_{n-1}),(s_{n-1},i_{n-1},o,s')$ for $|i| = n$. We say that a FST $\mathscr{T}$ is *square-free* if for every square-free word w, $\mathscr{T}(w)$ only contains square-free words. In Section 3.2, we used a square-free transducer as a tool in a construction. This motivates the more general study of square-free transducers, and in particular of the algorithmic problem of deciding whether a given transducer is square-free or not.

Problem 31. *(SQUARE-FREE)*
Input : *A transducer $\mathscr{T}$*
Output : *Either YES if $\mathscr{T}$ is square-free, or NO otherwise.*

This problem is co-semi-decidable in the sense that a brute force algorithm looking for a word w such that $\mathscr{T}$ contains a non-square-free word will stop if $\mathscr{T}$ is a NO-instance of SQUARE-FREE. The difficulty in this problem comes from the fact that the period of the smallest square produced by a transducer might be unbounded. Based on this observations, we propose the following conjecture.

Conjecture 32. *SQUARE-FREE is undecidable.*

In what follows, we give some ideas to tackle this problem. We first provide a sufficient condition on transducers for them to be square-free, and then we study some variants of SQUARE-FREE for which we have undecidability results. As this section is somewhat off-topic compared to the previous sections and is still an ongoing work, we opted for a less formal tone.

4.1 A certificate for square-freeness

There exists a simple certificate to assert that a transducer is not square-free, namely the word creating squares. In this section, we provide a finite certificate that asserts that a transducer can only create small squares. If, for a given transducer, there is such certificate, then the verification procedure to assert that the transducer is square-free or not only consists in checking that the transducer cannot create small squares.

Consider the transducer $\mathscr{T}_6$ given in Figure 3.1. We prove in Proposition 24 that this transducer is square-free, and another proof of this result would work as follows. Observe that for

every square-free word $wa \in \Sigma_3^+$ in input, where $w \in \Sigma^*$ and $a \in \Sigma_3$, there is exactly one word w' in $\mathscr{T}_6(wa)$. the subsequence congruent to 0 modulo 6 of w' is exactly w, the subsequence congruent to 1 modulo 6 of w' is exactly $\pi_3(w)$, the subsequence congruent to 2 modulo 6 of w' is exactly $\pi_3^2(w)$ and every factor of length 5 of the subsequences congruent to 3, 4 or 5 modulo 6 contain a square. Now assume that w' is a square of period at least $30 = 6 \times 5$. If the period is a multiple of 6, then a subsequence of w' that should be square-free is a square so this cannot happen. Otherwise, a factor of length at least 5 of a subsequence congruent to 0, 1 or 2 modulo 6 is equal to a factor of length at least 5 of a subsequence congruent to 3, 4 or 5 modulo 6. So a square-free subsequence contains a square which is absurd. Therefore, if w' contains a square, then it is of period at most 29 and a brute force algorithm can assert that it cannot be the case. This idea can be generalized (and applied for instance to the square-free transducer in Figure 3.3) with the notion of representative vectors.

Let $\mathscr{T}$ be a transducer and let $p \geq 3$. A vector $V \in (\mathbb{N} \cup \{\infty\})^p$ is a *representative vector* of $\mathscr{T}$ whenever

- $V_i = \infty$ if and only if for every square-free word w, and for every square-free word $w' \in \mathscr{T}(w)$, the subsequence congruent to i modulo p of w' is square-free.

- $V_i \in \mathbb{N}$ if and only if for every square-free word w, and for every word $w' \in \mathscr{T}(w)$, every factor of length V_i in the subsequence congruent to i modulo p of w' contains a square.

We say that a representative vector V of $\mathscr{T}$ is *arithmetic* whenever there exists $\alpha \in \{1, \ldots, p-1\}$ such that $\{i, V_i = \infty\} = \{j, i + \alpha \equiv j \pmod{p}\}$ and $V_i = \infty\}$. For instance $(\infty, \infty, \infty, 5, 5, 5)$ is a non-arithmetic representative vector of the transducer represented in Figure 3.1. Notice that if a representative vector has all its entries in $\mathbb{N}$, then it is arithmetic, so every non-arithmetic representative vector must have at least one of its entries equal to ∞.

Proposition 33. *Let $\mathscr{T}$ be a transducer and let $V \in (\mathbb{N} \cup \{\infty\})^p$ be a representative vector of $\mathscr{T}$ that is not arithmetic. Then for every square-free word w, and for every square-free word $w' \in \mathscr{T}(w)$, w' cannot contain squares of period at least $p \times \max\{V_i, V_i \neq \infty\}$.*

Proof. Let V be a representative vector of dimension p of $\mathscr{T}$ that is not arithmetic. Assume for the sake of contradiction that there exists w, w' such that w is square-free, $w' \in \mathscr{T}(w)$, and w' contains a square of period at least $p \times \max\{V_i, V_i \neq \infty\}$. Let u be the largest factor of w' that is a square. The period of u cannot be congruent to 0 modulo p because since V is not arithmetic, there exists i such that the subsequence of u congruent to i modulo p must be square-free and so u would not be a square. So let α be such that the period of u is congruent to α modulo p. Since V is not arithmetic, there must be a subsequence avoiding squares in the first period of u that is equal to a subsequence not avoiding squares in the second period (or the converse). Let j be such that this second subsequence corresponds the subsequence congruent to j modulo p. Since u is of period at least $p \times \max\{V_i, V_i \neq \infty\}$, then this second subsequence is of length at least V_j and therefore must contain a square. So a subsequence that is assumed to be square-free must contain a square which is absurd. Therefore, w' cannot contain a square of period at least $p \times \max\{V_i, V_i \neq \infty\}$. □

4.2 Square-free Turing Machines

In the remainder of this chapter, we prove that some variants of SQUARE-FREE are undecidable. To do so, we can harden the constraints on the outputs or give more computational expression to the transducers. Let $\mathscr{U}$ be an undecidable problem. To prove that a problem $\mathscr{P}$ is undecidable, we proceed by contradiction. We assume that we have an algorithm $\mathscr{A}$ that solves $\mathscr{P}$. If we can exhibit an algorithm $\mathscr{R}$ such that for every instance U of $\mathscr{U}$, $\mathscr{R}(U)$ is an equivalent instance of $\mathscr{P}$ (U is a YES-instance of $\mathscr{U}$ if and only if $\mathscr{R}(U)$ is a YES-instance of $\mathscr{P}$), then $\mathscr{A} \circ \mathscr{R}$ is an algorithm solving $\mathscr{U}$ thus leading to a state of absurdity. We refer the reader to [9, 27] for additional informations on this topic, and to [22] for a survey listing many undecidable problems in various fields of mathematics and computer sciences.

A Turing Machine is an abstract model of computing (i.e. a mathematical object capable of performing basic operations) introduced by Turing in [30]. Turing Machines have been used as a base to formally define the concept of computing, of complexity, and of decidability. There are many undecidable problems related to Turing Machines, and we strongly advise the reader to consult [27] for a comprehensive introduction to the topics of decidability and Turing Machines. Nevertheless, for the sake of completeness, we will provide a brief overview of Turing Machines.

A Turing Machine is very similar to FSTs, except that Turing Machines can choose to move their lecture heads towards the right or towards the left and can modify their inputs (thus giving memory to the model). A Turing Machine has two components. A *Controller* that will read some input and perform a small computation, and a *Tape* that will act as a memory, and is a "line of cells" of indices $0, 1, \ldots$. The Controller will then read some (finite) data on the tape at the position of the *lecture head*, possibly rewrite some data on the tape, move the head towards the right or towards the left, and change its own state. More formally, a Turing Machine is a 6-uplet $\mathscr{M} = (\Sigma, \Gamma, S, T, s_0, A)$ where

- Σ is the input alphabet, not containing $\sqcup$ (the "space" symbol),
- Γ is the tape alphabet such that $\Sigma \subseteq \Gamma$ and $\sqcup \in \Gamma$,
- S is a finite set (of states),
- $T \subseteq S \times \Gamma \times \{L, R\}, \times \Gamma \times S$ is the set of transitions,
- $s_0 \in S$ is the starting state,
- A is a set of accepting states.

At the beginning of the process, the Turing Machine reads a word $w \in \Sigma^*$ as its *input*. The tape of the machine has w written on it in the sense that the cell at position 0 contains w_0, the cell at position 1 contains $w_1, \ldots$, the cell at position $|w| - 1$ contains $w_{|w|-1}$, and all other cells contain $\sqcup$. The *tape word* of a tape is the largest word written on the tape that does not finish with $\sqcup$. For instance, the input *hello*, *world* will be represented as the tape

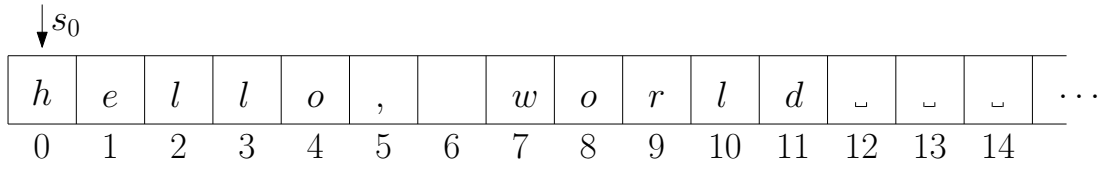


Figure 4.2: Example of a tape with an input word

with the lecture head pointing at position 0. A *configuration* (or *snapshot*) of $\mathscr{M}$ is a representation of the current state of the computation that contains the position of the head, the current state in the Turing Machine, and the current tape word. More concretely, a configuration is a word usv meaning that the tape word is uv, that the current state is s, and that the lecture head is at position $|u|$ (it reads v_0), so the configuration of the state of the Turing Machine from Figure 4.2 is $s_0 hello$, *world* (if $v = \varepsilon$, then the lecture head reads ␣). A *step of computation* then works as follows. Assume the current configuration is $usbv$. If $s \in A$, then the initial word is *accepted*. Otherwise, if $(s,b,R,c,s') \in T$, then the next configuration is $ucs'v$. If $(s,b,L,c,s') \in T$, then we assume that u is not empty, equals $u'a$ for some word u and some letter a, and the next configuration is $u's'acv$. In this model, we only consider *deterministic* Turing Machines, so we cannot have both (s,b,D,c,s') and (s,b,D',c',s'') in T at the same time for some $D \neq D'$ or $c \neq c'$ or $s' \neq s''$. If none of (s,b,D,c,s') is in T for any $D \in \{L,R\}$, $c \in \Gamma$, and $s' \in S$, then the computation stops, and the initial word is *rejected*.

For instance, let's define a Turing Machines that computes the following function $f(a^m) = ba^m$ as in Figure 4.3.

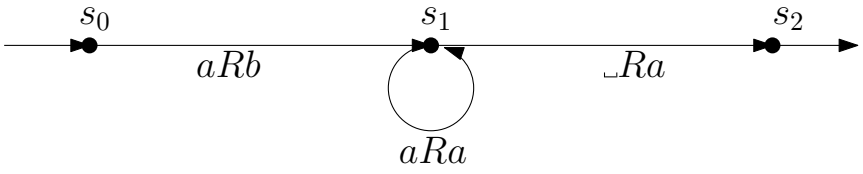


Figure 4.3: A Turing Machine computing $f(a^m) = ba^m$

For an initial input *aaaa*, the sequence of configurations is

$$s_0aaaa \to bs_1aaa \to bas_1aa \to baas_1a \to baaas_1 \to baaaas_2$$

and s_2 is an accepting state. Notice that this function can be expressed by a FST (in fact, it is even simpler). However, Turing Machines are strictly more powerfull than transducers in the sense that every function expressible by a transducer is expressible by a Turing Machine, and there exists functions expressible by a Turing Machine not expressible by a transducer. The Turing Machines are very expressive, and therefore are subject to many undecidable problems, one of them being to know if an input will, or will not, be accepted by a given Turing Machine.

Problem 34. *(A_{TM})*
Input : *A Turing Machine $\mathscr{M}$, a word w*
Output : *Either YES if w is accepted by $\mathscr{M}$ or NO otherwise.*

This problem is one of the most basic undecidable problems about Turing Machines and used as a base for the proof of many undecidable problems.

Theorem 35 (Turing, 1936 [30]). *A_{TM} is undecidable.*

In the scope of this work, we are interested in square-free words, so in the remainder of this section, we will study square-free Turing Machines. A Turing Machine $\mathscr{M}$ is *square-free with input w* whenever w is square-free and at every step of the computation, the tape word is square-free.

Problem 36. *(SF-A_{TM})*
Input : *A Turing Machine $\mathscr{M}$, a word w such that $\mathscr{M}$ is square-free with input w*
Output : *Either YES if w is accepted by $\mathscr{M}$ or NO otherwise.*

Rice theorem (see [19]) states that every non-trivial (not always true or always false) property about the language of a Turing Machine is undecidable. Here, SF-A_{TM} is not whether a Turing Machine $\mathscr{M}$ is square-free or not (it is undecidable as a consequence of Theorem 37) but about whether a Turing Machine $\mathscr{M}$ accepts a word w, with the additional information (or hint) that $\mathscr{M}$ is square-free on input w so we cannot infer anything about the decidability of SF-A_{TM} with Rice theorem. SF-A_{TM} is by definition at most as hard as A-$_{TM}$, and in fact it is undecidable as well.

Theorem 37. *SF-A_{TM} is undecidable.*

Proof. Let $(\mathscr{M},w)$ be an instance of A_{TM} with $\mathscr{M} = (\Sigma,\Gamma,S,T,s_0,A)$. We will construct $(\mathscr{M}',w')$, an equivalent instance of SF-A_{TM}, where $\mathscr{M}'$ is square-free with input w'.

Before that, let us go back to *vtm*. As explained in Chapter 1, we can construct an infinite ternary square-free word with the morphism $h(0) = 012$, $h(1) = 02$, and $h(2) = 1$. Indeed, even if h is not a square-free morphism, the fixed point of h called *vtm* is square-free. We can then construct an increasing sequence of prefixes of *vtm* that are all square-free. We start with the word $vtm_0 = 0$. At the $(i+1)$-th step of the procedure, let a be the letter at position i of vtm_i. We construct $vtm_{i+1} = vtm_i h(a)$, so the first few steps of the procedure are $vtm_0 = 0$, $vtm_1 = 012$, $vtm_2 = 01202$, $vtm_3 = 012021$ and it is well known that this procedure constructs *vtm* as desired.

The idea to construct $(\mathscr{M}',w')$ is to simulate the computation of $\mathscr{M}$ with input w, and the construction of *vtm* at the same time. To do so, we define $\mathscr{M}' = (\Sigma',\Gamma',S',T',s_0,A)$ with

- $\Sigma' = \{\vdash\} \cup [(\Sigma \cup \{\sim\}) \times \{0,1,2\} \times \{\sim,\#,\triangle\} \times \{\sim,\bullet\}]$,
- $\Gamma' = \Sigma' \cup \{\sqcup\}$,
- $S \subseteq S'$,

and we replace all transitions $(s,a,D,b,s') \in T$ by the gadget in Figure 4.4. For a word $u = (\alpha_i,\beta_i,\gamma_i,\delta_i)_{i\in\{0,\dots,N\}}$, we denote $\alpha(u) = \prod_{i=0}^{N} \alpha_i$, and similarly for $\beta(u), \gamma(u)$, and $\delta(u)$. Let i be the smallest integer such that $|vtm_i| \geq |w|$. We construct $w' = \vdash t$ where t is the word over $(\Sigma \cup \{\sim\}) \times \{0,1,2\} \times \{\sim,\#,\triangle\} \times \{\sim,\bullet\}$ such that $\alpha(t) = w \sim^{|vtm_i|-|w|}$, $\beta(t) = vtm_i$, $\gamma(t) = \sim^i \# \sim^{|vtm_i|-i-2} \triangle$, and $\delta(t) = \sim^{|vtm_i|}$. So, for instance, if $w = 01234567$, then $i = 4$, $vtm_4 = 012021012$, $w' = \vdash t$, and t is exactly

$$(0,0,\sim,\sim)(1,1,\sim,\sim)(2,2,\sim,\sim)(3,0,\sim,\sim)(4,2,\#,\sim)(5,1,\sim,\sim)(6,0,\sim,\sim)(7,1,\sim,\sim)(\sim,2,\triangle,\sim)$$

Assume that at some point in the computation of $\mathscr{M}'$ on input w', the lecture head is at position p, the state of the machine is s with $s \in S$ (so a state originally of $\mathscr{M}$), and the tape word is $\vdash t$, with $\beta(t) = vtm_i$, $\gamma(t) = \sim^i \# \sim^{|vtm_i|-i-2} \triangle$, and $\delta(t) = \sim^{|vtm_i|}$.

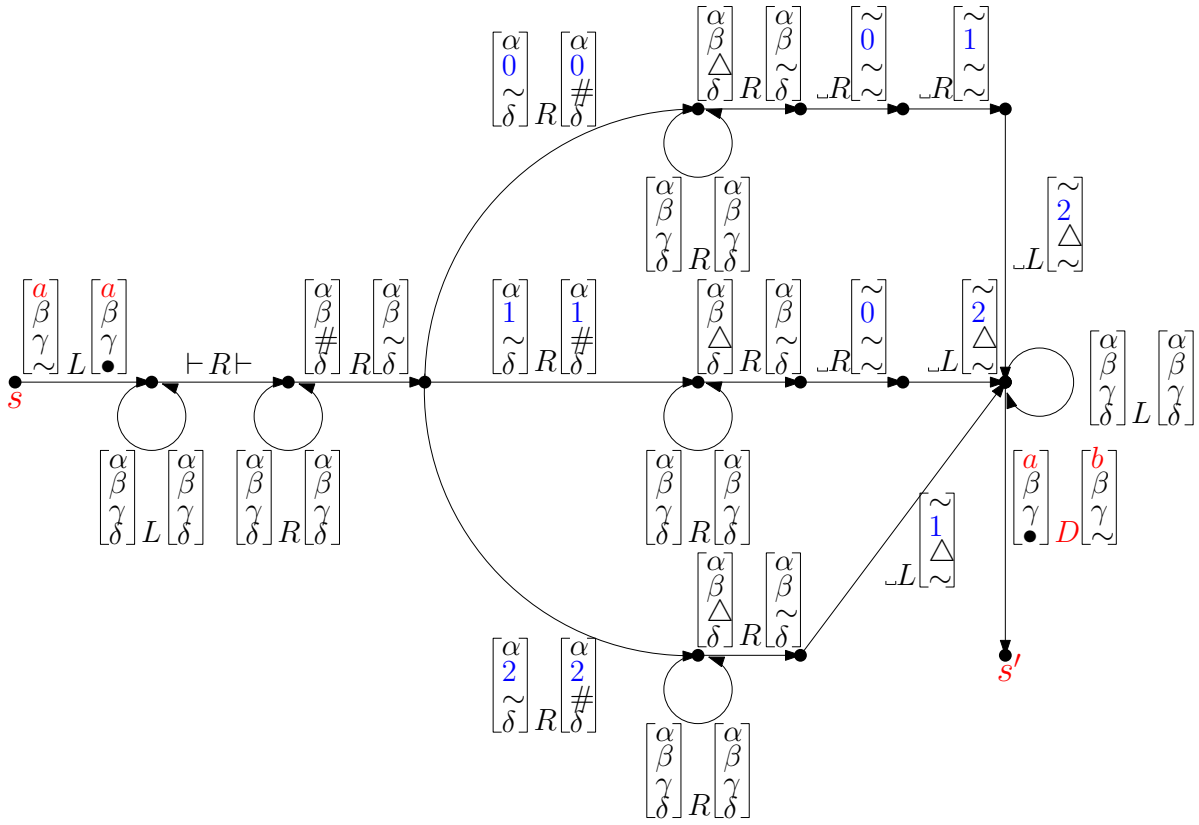


Figure 4.4: The gadget in the reduction to replace a transition (s, a, D, b, s')

The computation of $\mathscr{M}'$ will consist of the following steps until some state s' of $\mathscr{M}$ is reached :

1. Mark the current position by modifying its symbol x with $\delta(x) = \bullet$.
2. Move to the leftmost position of the tape without doing any modification.
3. Move towards the right until you find a position with a symbol x such that $\gamma(x) = \#$.
4. Modify x with $\gamma(x) = \sim$ and move one step towards the right.
5. Read the new symbol x and remember $\beta(x)$, say it is 1 for instance.
6. Move towards the right until you reach a position with a symbol x such that $\gamma(x) = \triangle$.
7. Replace the $\triangle$ with a $\sim$, then move one one step towards the right.
8. Write $(\sim, 0, \sim, \sim)$ then move one step towards the right and write $(\sim, 2, \triangle, \sim)$.
9. Move towards the left until you find a position with a symbol x such that $\delta(x) = \bullet$.
10. Replace the $\bullet$ by a $\sim$. Assuming $\alpha(x) = a$, set $\alpha(x) = b$.
11. Move one step towards the right, the current state in the automaton is s'.

The symbol • is used to remember the position of the lecture head at the beginning of the process previously described. The symbol # is always at position i if $\beta(t)$ is vtm_i, and the symbol $\triangle$ is always at position $|vtm_i|$. At each step of the computation, if $\vdash t$ is the tape word, then $\beta(t)$ is a prefix of vtm so the tape word is always square-free. If we look at the transformation of $\alpha(t)$, then we exactly followed the transition (s,a,D,b,s') of $\mathscr{M}$, so $\alpha(t)$ simulates the computation of $\mathscr{M}$. Therefore, $(\mathscr{M}',w')$ is an instance of SF-A_{TM}, and since $\alpha(t)$ simulate the computation of $\mathscr{M}$ on input w, $(\mathscr{M},w)$ is a YES-instance of A_{TM} if and only if $(\mathscr{M}',w')$ is a YES-instance of SF-A_{TM}. So SF-A_{TM} is undecidable as desired. □

4.3 PCP and SQUARE-FREE PCP

The Post correspondence problem (PCP) is a well-known undecidable problem introduced by Post in [23]. PCP is widely used for undecidability proofs due to its simplicity and expressiveness. A lot of attention has been given on variants of PCP to understand what the limits of its undecidability are (see [17,25] for instance). In what follows, we will use the undecidability of PCP as a base to prove the undecidability of some other problems.

Problem 38. *(PCP)*
Input : *Two morphisms g, h*
Output : *Either YES if there exists a finite word w such that $g(w) = h(w)$ or NO otherwise.*

Theorem 39 (Post, 1946 [23])**.** *PCP is undecidable.*

The aim of this section is to prove that a square-free variant of PCP asking for a square-free input and a square-free output, is undecidable.

Problem 40. *(SF-PCP)*
Input : *Two morphisms g, h*
Output : *Either YES if there exist square-free words w, w' such that $g(w) = h(w) = w'$ or NO otherwise.*

We will later use the instances of SF-PCP to prove that a related problem is undecidable, but first, we shall prove that SF-PCP itself is undecidable.

Lemma 41. *SF-PCP is undecidable.*

Proof. For the proof of this undecidability result, we will do small modifications on the proof of undecidability of PCP from [27] and use Theorem 37 to conclude.

The proof of the undecidability of PCP as given in [27] goes as follows. From a Turing Machine $\mathscr{M} = (\Gamma,\Gamma',S,T,s_0,A)$, and an input word w, we define the following two morphisms $g : \Sigma \to \Sigma'$ and $h : \Sigma \to \Sigma'$ for some alphabets Σ and Σ':

- $g(\vdash) = \#$, $h(\vdash) = \#s_0w\#$.
- $g(R_{s,a}) = sa$, $h(R_{s,a}) = bs'$ for every $(s,a,R,b,s') \in T$.
- $g(L_{c,s,a}) = csa$, $h(L_{c,s,a}) = s'cb$ for every $c \in \Gamma$ and $(s,a,L,b,s') \in T$.
- $g(a) = l_a$, $h(a) = l_a$ for every $a \in \Gamma$.
- $g(\#) = \#$ and $h(\#) = \#$.

- $g(A_{a,s}) = as$, $g(A_{s,a}) = sa$, $h(A_{a,s}) = s$, $h(A_{s,a}) = s$ for every $a \in \Gamma$, $s \in A$.
- $g(F_s) = s\#\sim$, $h(F_s) = \sim$ for every $s \in A$.

The largest (and most important) part of our proof comes from the proof of undecidability of PCP. The following claim is a summary of the proof from Sipser's book Introduction to the Theory of Computation [27].

Claim 42 ([27])**.** *$(\mathscr{M}, w)$ is a YES-instance of A_{TM} if and only if (g,h) is a YES-instance of PCP where we enforce that the solution must start with s (and this enforcement can be bypassed by doing a small modification to our morphisms), so PCP is undecidable.*

Moreover, assume $(\mathscr{M}, w)$ is a YES-instance of A_{TM}, then let $(c_i)_{i\in\{0,\dots,n\}}$ be the finite sequence of all configurations in the accepting run of $\mathscr{M}$ on input w. We define a function $inv : \Sigma^* \to \Sigma'^*$ such that

- *$inv(usav) = uR_{s,a}v$ for every u, $v \in \Sigma^*$ whenever $(s,a,R,b,s') \in T$*
- *$inv(ucsav) = uL_{c,s,a}v$ for every u, $v \in \Sigma^*$ whenever $(s,a,L,b,s') \in T$*

The function inv can be seen as a somewhat inverse function of g and h over configurations of $\mathscr{M}$. Then the smallest solution for the instance (g,h) of PCP is a word S such that

$$S =\vdash (\prod_{i=1}^{n-1} inv(c_i)\#)f$$

and

$$g(S) = h(S) = \#(\prod_{i=0}^{n} c_i\#)f'$$

where f and f' are defined from the configuration $c_n = u_0 \dots u_m s u'_0 \dots u'_{m'}$ as

$$f = u_0 \dots u_{m-1} A_{u_m,s} u'_0 \dots u'_{m'} \# u_0 \dots u_{m-2} A_{u_{m-1},s} u'_0 \dots u'_{m'} \# \dots \# A_{s,u'_0} u'_1 \dots u'_{m'} \# A_{s,u'_1} u'_2 \dots u'_{m'} \# \dots \# A_{s,u'_{m'}} \# F_s$$

and

$$f' = u_0 \dots u_{m-1} s u'_0 \dots u'_{m'} \# u_0 \dots u_{m-2} s u'_0 \dots u'_{m'} \# \dots \# s u'_0 \dots u'_{m'} \# s u'_1 \dots u'_{m'} \# \dots \# s \# \sim .$$

For instance, assume that the sequence of configurations of an accepting Turing Machine $\mathscr{M}$ on input 01 is $s_0 01 \to 2 s_0 1 \to s_0 20 \to 1 s_a 0$. Then the smallest solution to the equivalent PCP instance provided by the reduction is

$$g(\vdash R_{s_0,2} l_1 \# L_{2,s_0,1} \# R_{s_0,2} l_0 \# A_{1,s_a} l_0 \# A_{s_a,0} \# F_{s_a}) = \#s_0 01 \# 2 s_0 1 \# s_0 20 \# 1 s_a 0 \# s_a 0 \# s_a \# \sim$$

If $\mathscr{M}$ accepts w and is square-free on input w, then for every i, c_i is square-free. Moreover, for every i, j such that $i \neq j$, $c_i \neq c_j$ otherwise, since we consider deterministic Turing Machines, the computation entered an infinite loop. Since all the c_i are square-free and distinct, $g(S)$ and $h(S)$ are square-free and since all the c_i are square-free and distinct, then so are all the $inv(c_i)$ (notice that the converse is not necessarily true, for instance $R_{s,0}0$ is square-free but $s00$ is not), so S is also square-free. Therefore, $(\mathscr{M}, w)$ is a YES-instance of SF-A_{TM} if and only if (g,h) is a YES-instance of SF-PCP. Since SF-A_{TM} is undecidable by Theorem 37, then SF-PCP is undecidable as well. □

4.4 Undecidable transducers problems

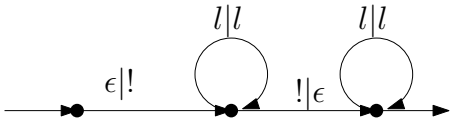
In this section, we apply results from Section 4.3 to prove that some problems related to square-free transducers are undecidable.

Problem 43. *(2-SQ)*
Input : *A transducer $\mathscr{T}$*
Output : *Either YES if there exists a square-free word w such that there exists w', $w'' \in \mathscr{T}(w)$, $w' \neq w''$ and w' and w'' contain a square, or NO otherwise.*

Proposition 44. *2-SQ is undecidable.*

Proof. To prove that 2-SQ is undecidable, we will provide a transducer that is a YES-instance of 2-SQ if and only if the instance of SF-PCP given in the proof of Lemma 41 is a YES-instance of SF-PCP, so the reduction uses the undecidability of SF-A_{TM} given in Theorem 37 to conclude. Let $(\mathscr{M}, w)$ be an instance of SF-A_{TM} and let (g,h) be the equivalent SF-PCP instance constructed in the proof of Lemma 41. Consider the following transducer :

Top Component :



Bottom Component :

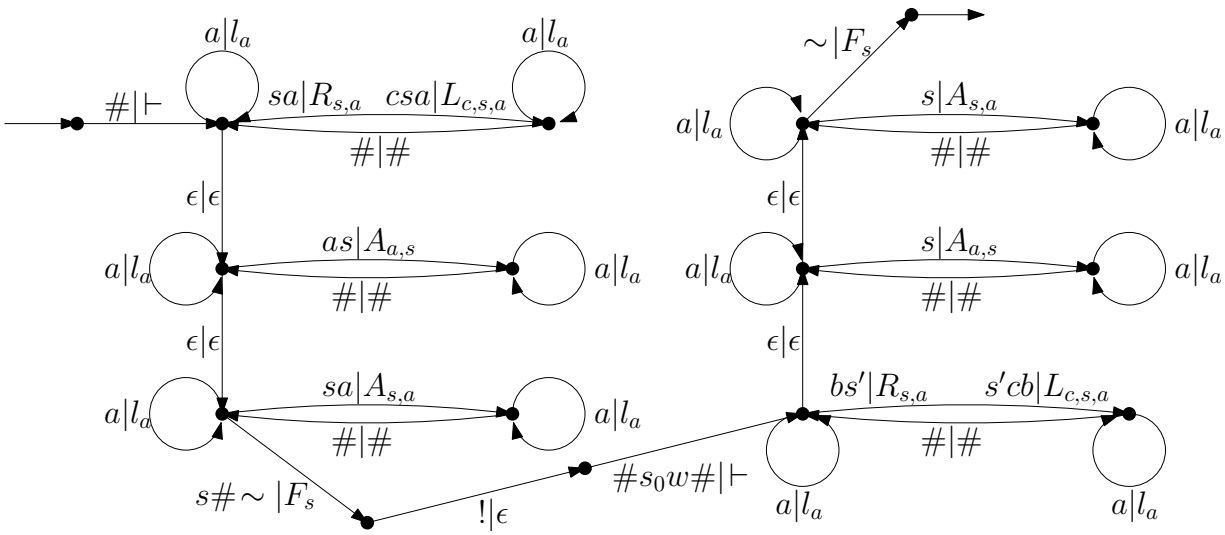


Figure 4.5: A transducer $\mathscr{T}$ with two disjoint components. The label $l|l$ hides an implicit "for every $l \in \Gamma'$". The label $a|l_a$ hides an implicit "for every $a \in \Gamma'$". The labels $sa|R_{s,a}$, $csa|L_{c,s,a}$, $bs'|R_{s,a}$, and $s'cb|L_{c,s,a}$ hide an implicit "for every $(s,a,R,b,s') \in R$". The labels $sa|A_{s,a}$, $as|A_{a,s}$, $s|A_{s,a}$, and $s|A_{a,s}$ hide an implicit "for every $a \in \Gamma', s \in A$". The labels $s\#\sim|F_s$ and $\sim|s$ hide an implicit "for every $s \in A$".

If (g,h) is a YES-instance of SF-PCP, then let u be a square-free word such that $g(u) = h(u)$ and $g(u)$ and $h(u)$ are square-free. Then $\mathscr{T}(g(u)!h(u))$ contains two words that are squares,

namely $!u!u$ and $!g(u)!h(u)$. On the other hand, assume that $\mathscr{T}$ is a YES-instance of 2-SQ. The two components produce at most one word each, so one word with a square must have been produced in the top component, and one word with a square must have been produced in the bottom component (as represented in Figure 4.5). The top component reads a word $u!v$ and outputs $!u!v$. Since $! \notin \Gamma'$, u and v do not contain $!$ as a letter. Moreover, since $u!v$ must be square-free, then u and v are square-free. Therefore, if the top component produces a word containing a square, then the square is exactly $!u!v$ and $u = v$. Similarly, the bottom component reads a word $w = g(u)!h(v)$ and outputs the word $!u!v$. Since $g(u)$ and $h(u)$ are square-free, u and v are also square-free for the reasons given in the proof of Lemma 41. So, if $!u!v$ contains a square, then $!u!v$ is a square and $u = v$. But then $g(u) = h(u)$ so u is a solution to the instance (g,h) of square-free PCP. Therefore, (g,h) is a YES-instance of SF-PCP if and only if $\mathscr{T}$ is a YES-instance of 2-SQ and so $\mathscr{T}$ is a YES-instance of 2-SQ if and only if $\mathscr{M}$ accepts the input w. Since SF-A_{TM} is undecidable by Theorem 37, then so is 2-SQ.

□

Here, we needed to produce two squares to ensure that the input was a word $u!u$. This can be avoided by either only considering inputs that can be written $u!u$, or by giving more power to our transducers (such as two-pass transducers, two-ways transducers or stack transducers).

4.5 Bounded SQUARE-FREE

The difficulty in SQUARE-FREE comes from the fact that w could have arbitrarily large length. In the bounded version of SQUARE-FREE, we ask for w to have bounded length, thus a brute force algorithm solves this problem. In this section, we prove that a somewhat complement of BOUNDED-SQUARE-FREE is NP-Hard.

Problem 45. *(BOUNDED-SQUARE-FREE^c)*
Input : *A transducer $\mathscr{T}$, an integer n*
Output : *Either YES if there exists a word w of length n such that for every $w' \in \mathscr{T}(w)$, w' is square-free or NO otherwise.*

A graph G is a pair (V,E) where V is a finite set of elements called *vertices*, and E is a set of binary relations over V called *edges*, and if the pair $(u,v) \in E$, then we simply denote it uv. We refer the reader to [11] for a complete introduction on graph theory.

Figure 4.6: A graph $G = (V,E)$ with $V = \{a,b,c,d\}$ and $E = \{ab,ac,bc,bd,cd\}$

A *path* in a graph $G = (V,E)$ is a sequence $v_0,\ldots,v_n$ such that for every $i \geq 0$, $v_i \in V$, for every i, j, if $i \neq j$ then $v_i \neq v_j$, and for every $i \in \{0,\ldots,n-1\}$, $v_iv_{i+1} \in E$. A path is of length n if it contains n elements, and we say that a path is a *Hamiltonian path of G* whenever it is of size $|V|$. For instance, in Figure 4.6, a,b,d is a path, a,b,c,d is a hamiltonian path, and a,d is not a path.

Problem 46. *(HAMILTONIAN PATH)*
Input : *A graph G*
Output : *Either YES if G has a Hamiltonian path, or NO otherwise.*

Theorem 47 (Folklore)**.** *HAMILTONIAN PATH is NP-Complete.*

Proposition 48. *BOUNDED-SQUARE-FREE^c is NP-Hard.*

Proof. Let G be a graph on n vertices. For every vertex $v \in V(G)$, we define the Finite State Transducer $\mathscr{T}_v = (V(G), V(G), S \cup \{s\}, T, S, S)$ such that $S = \{s_u, u \in V(G)\}$ and

$$T = \{(s_u, u, \varepsilon, s_{u'}), uu' \in E(G) \text{ and } u \neq v\} \cup \{(s_v, v, v, s_{u'}), vu' \in E(G)\}$$

We then construct $\mathscr{T} = \biguplus_{v \in V(G)} \mathscr{T}_v$. For instance, assume that G is as in Figure 4.6. Then $\mathscr{T}_a$ is exactly

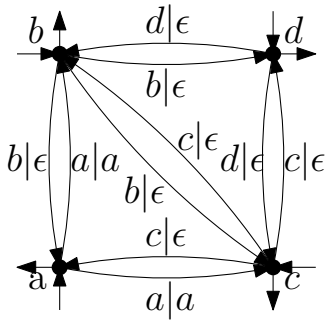


Figure 4.7: example for $\mathscr{T}_a$

And then we take $\mathscr{T}$ to be the disjoint union of $\mathscr{T}_a$, $\mathscr{T}_b$, $\mathscr{T}_c$ and $\mathscr{T}_d$. To construct $\mathscr{T}$, we construct a transducer that is the disjoint union of $|V(G)|$ transducers of size $|V(G)|$, so constructing $\mathscr{T}$ is doable in time polynomial in the size of G. Assume that G has a hamiltonian path $P = v_0, \ldots, v_{n-1}$, then $\mathscr{T}(P) = \{\varepsilon\}$ since there is no repetition of vertex in P. If G has no hamiltonian path, then in every path P of length n in G, there exists a vertex v that is repeated, in which case, $\mathscr{T}(P)$ must contain a non-square-free word produced by $\mathscr{T}_v$. Therefore, G is a YES-instance of HAMILTONIAN if and only if $(\mathscr{T}, n)$ is a YES-instance of BOUNDED-SQUARE-FREE^c. Since HAMILTONIAN is NP-Hard, then so is BOUNDED-SQUARE-FREE^c. □

Corollary 49. *If BOUNDED-SQUARE-FREE^c has $n = O(|\mathscr{T}|)$ as an additional constraint, then it is NP-Complete.*

Proof. If $n = O(|\mathscr{T}|)$, then the path itself constitutes a certificate of polynomial size. □

— 5 —
Conclusion

In this report, we were interested in pairs of relatively prime integers (p,q) for which there exist infinite ternary words that are square-free, square-free modulo p, and square-free modulo q. This question was asked by Harju in 2019 in [18] and the same year, two positive solutions, $(3,11)$ and $(5,6)$, were given by Currie, Rampersad, Harju and Ochem in [8]. Since then, no new pairs were given thus leaving an infinite number of unknown solutions.

We first presented a construction for pairs of relatively prime integers (p,q) where both p and q are large, thus providing the first infinite family of positive solutions covering the north-east quarter of the plan. This construction uses a multi-valued morphism that allows us to perform corrections on a square-free word to make its subsequences square-free while maintaining the word itself square-free. The idea is that thanks to this powerful morphism, we can reduce our problem to the problem of finding a (non-necessarily square-free) word whose subsequences modulo p and q are square-free.

Then, for the remaining cases where either p or q is small, we provided another construction using a family of circular morphisms. With these morphisms, we reduced the problem to constructing a square-free word with forced letters at some positions, and it was already known that such word exists when the constraints are at distance at least 19 from each other (see Theorem 19). Our technique uses a family of morphisms that were known for almost all values of p except a few. For the last few cases, we generalized our method and found morphisms and transducers that worked perfectly. These two constructions are sufficient to cover almost all the set of pairs of relatively prime numbers, leaving a finite uncovered subset thus proving that all but a finite number of pairs of relatively prime numbers are positive solutions to the problem proposed by Harju. Finally, we started to fill this finite hole by providing some explicit morphisms to construct the desired words when both p and q are small.

We focused on pairs of relatively prime numbers since the initial question of Harju only considered such pairs. However, there exist some very natural generalizations of this problem that are yet to be studied, for instance, one can consider all pairs of integers.

Question 50. *For what values of integers p, q, $s_p < p$, $s_q, < q$ does there exist an infinite ternary square-free word w such that the subsequences congruent to s_p modulo p and congruent to s_q modulo q are square-free ?*

For the simpler case $s_p = s_q = 0$, we already know that if $q = 2p$ or if $q = \frac{3}{2}p$ then there is no such words, thus there exists an infinite number of negative solutions. Moreover, our constructions given in Chapter 3 never use the hypothesis that p and q are relatively prime integers, so there also exists an infinite number of positive solutions. Notice however that all

of this holds for the case $s_p = s_q = 0$ and the more general problem considering other values for s_p and s_q might have a much different behaviour. For instance, one could consider the case where $p = q = 2k$, $s_p = 0$ and $s_q = k$. Since the two subsequences never meet, we already know that there exist positive answers when $k \geq 19$ by Theorem 19 and we expect that this bound is improvable.

Another natural generalization could be to consider more than two subsequences, that is k-uplets instead of pairs for some $k \geq 3$.

Question 51. *Let $k \geq 3$ be an integer. For what values of relatively prime integers $p_1, \ldots, p_k$ does there exist an infinite ternary square-free word whose subsequence modulo p_i is square-free for every $i \in \{1, \ldots, k\}$?*

A more in-depth use of the technique presented in Chapter 2 could work although it is very much not clear. Intuitively, as in Chapter 2, if all the p_i are large integers we expect that such words exist. Similarly, if the sequence $(p_i)_{1 \leq i \leq k}$ grows sufficiently fast (for instance $p_i \geq p_1 \times 20^{i-1}$ as in Chapter 3) then such word might exist. Although they might seem somewhat precise, these two suggestions will need to be rephrased and further clarified before they can be considered as conjectures. Similarly to Question 50, we can give a generalization of Question 51 that considers all k-uplets of integers.

Question 52. *Let $k \geq 2$ be an integer. For what values of integers $p_1, \ldots, p_k$ and $s_1, \ldots, s_k$ with $s_i < p_i$ for every $i \in \{1, \ldots, k\}$ does there exist an infinite ternary square-free word whose subsequence congruent to s_i modulo p_i is square-free for every $i \in \{1, \ldots, k\}$?*

The comments on Question 50 suggest that considering pairs of non-relatively prime numbers makes the problem harder. It seems that this might not be the case for Question 52, indeed the transducer provided in Chapter 3 (see Figure 3.3) allows us to construct words that are square-free, square-free modulo 3, square-free modulo 3^2, ..., and square-free modulo 3^k for k arbitrarily large, and similarly for 6 instead of 3 with the transducer presented in Figure 3.1. More generally, let $m_0, \ldots, m_k$ be a sequence of integers greater than 3. By composing methods that construct infinite ternary words that are square-free and square-free modulo p when the subsequence modulo p is provided, we can manage to construct infinite ternary square-free words that are square-free modulo $p_n = \prod_{i=0}^{n} m_i$ for every $i \in \{0, \ldots, k\}$.

The transducers from Figure 3.1 and Figure 3.3 share the property that they are square-free in the sense that if they read a square-free word in input, then they will write a square-free word in output. Transducers are a very natural generalization of morphisms, and even tho there exists simple characterizations for square-free morphisms (see Crochemore's characterizations, Theorem 4), we do not know any algorithm asserting that a transducer is square-free or not and this motivated the study of the decision problem "is a given transducer square-free ?". This lead to Chapter 4 where we gave a sufficient condition on transducers for them to have a certificate of their square-freeness. We also studied variants of this algorithmic problem and gave some results of undecidability and NP-Hardness about them and about related algorithmic problems.

Appendix

p	$h(0)$	k	α	$q \geq$
3	0120212012102010210121020120210121021201020121021202 10	18	1	1080
4	0120212010210120102120121012021020120210121020121012021012102012021 0	17	0	1360
5	0121020102101202120102012101202101210201021202101210201202120121 0	13	0	1300
7	0120210201021012010212012102010212010201202102012101202101210212012 0- 1012010212012102012021 0	13	0	1860
8	0120212010201210201021201210120102012021201020121012021020120212012 0- 1012102010212012102010210121021202 10	13	0	2080
9	0120210201021012102010212021012010210120212012101201020121020120210 1- 2010201202101210201202120102101210212012010012021 0	13	0	2340
10	0120212012102012021020102120121020120212010212021020102120121020120 2- 1020102120210201202120102012021020102120121020120212010212021 0	13	0	2600
11	0120210201021202102012102120102012021201210201021201020121021202 0- 1202120121020120210201210212010201210201021201210212021020120210 2- 0120210	13	0	2860
12	0120212012101202101210212012102012021012102120210120212012102012021 0- 2012101202101210201202120121012021012102120121012021201210201202120 1- 2101202101210212021 0	13	0	3120
14	0120212012102012021012102120210120212012101202101210201202102012102 1- 2021012021201210120210121021202102012021012102012101202120121020120 2- 1201210120210121021201210201202102012102120210	13	0	3640
15	0121021201210201202102012101202101210212012101202101210201210120210 2- 0121021201210120212012102120210121021201210120210120201202102012101 - 2021012102012021201210212021012102012101202102012102012021012102120210	13	0	3900
16	0121021201210120102012102010210120102012101201021012102012101201020121 0- 2102010212012101201021012102120102012101201021012102010210120102012 1- 0201021012102120102012102010212012102012010210120102012101201020 - 1210	13	0	4160
20	0121021201210120212012102012021012102120210120212012101202101210201 2- 0210201210120210121020121012021020120210121021201210120210120201202 - 1201210120210121021201210120210201202101210212012101202101210201202 1- 2012101202102012102120121012021201210212021012102120210	13	0	5200
21	0121021201210120210121020120210201210120212012102012021020120121012 0- 1012021012102012021020121012021020120210121020120210201210120210121 0- 2012101202120121020120210121021201210120210121020120210201210120210121 0- 2102012021201210120210121021201210120212021012102120210	13	0	5460
22	0121021201210120212012102012021012102120121020120210201210120210121 0- 2012021020121021202101202120121012021012102012021020121012021012021 0- 1202120121012021012102012101202102012102120121012021012021012021012 0- 2012021020121021201210120210121020121012021020121021202101210212 1- 02012021201210	13	0	5720

Figure 5.1: Morphisms for Proposition 23. A ligne provides p, the morphism, the parameters k and α of Proposition 23, and the value of q from which good words exist.

Bibliography

[1] Jean-Paul Allouche and Jeffrey Shallit. The ubiquitous prouhet-thue-morse sequence. In *Sequences and their Applications: Proceedings of SETA'98*, pages 1–16. Springer, 1999.

[2] Noga Alon, Jarosław Grytczuk, Mariusz Hałuszczak, and Oliver Riordan. Nonrepetitive colorings of graphs. *Random Structures & Algorithms*, 21(3-4):336–346, 2002.

[3] Marie-Pierre Béal and Dominique Perrin. Symbolic dynamics and finite automata. In *Handbook of Formal Languages: Volume 2. Linear Modeling: Background and Application*, pages 463–506. Springer, 2013.

[4] Jean Berstel. *Axel Thue's papers on repetitions in words: a translation*, volume 20. Départements de mathématiques et d'informatique, Université du Québec à Montréal, 1995.

[5] Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

[6] Maxime Crochemore. Sharp characterizations of squarefree morphisms. *Theoretical Computer Science*, 18(2):221–226, 1982.

[7] James Currie. Infinite ternary square-free words concatenated from permutations of a single word. *arXiv preprint arXiv:1207.3445*, 2012.

[8] James Currie, Tero Harju, Pascal Ochem, and Narad Rampersad. Some further results on squarefree arithmetic progressions in infinite words. *Theoret. Comput. Sci.*, 799:140–148, December 2019.

[9] Martin Davis. *Computability and unsolvability*. Courier Corporation, 2013.

[10] Michel Dekking and Michael Keane. Two-block substitutions and morphic words. *Advances in Applied Mathematics*, 148:102536, 2023.

[11] Reinhard Diestel. *Graph theory*. Springer (print edition); Reinhard Diestel (eBooks), 2024.

[12] Vida Dujmović, Louis Esperet, Gwenaël Joret, Bartosz Walczak, and David R Wood. Planar graphs have bounded nonrepetitive chromatic number. *arXiv preprint arXiv:1904.05269*, 2019.

[13] Calvin Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51, 1961.

[14] Gabriele Fici and Svetlana Puzynina. Abelian combinatorics on words: A survey. *Computer Science Review*, 47:100532, 2023.

[15] Pytheas Fogg, Valéry Berthé, Sébastien Ferenczi, Christian Mauduit, and Anne Siegel. *Substitutions in dynamics, arithmetics and combinatorics*. Springer, 2002.

[16] Jarosław Grytczuk. Nonrepetitive colorings of graphs—a survey. *International journal of mathematics and mathematical sciences*, 2007(1):074639, 2007.

[17] Vesa Halava, Mika Hirvensalo, and Ronald de Wolf. Marked pcp is decidable. *Theoretical Computer Science*, 255(1):193–204, 2001.

[18] Tero Harju. On square-free arithmetic progressions in infinite words. *Theoretical Computer Science*, 770:95–100, 2019.

[19] John Hopcroft, Rajeev Motwani, and Jeffrey Ullman. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1):60–65, 2001.

[20] Stephen Cole Kleene. Representation of events in nerve nets and finite automata. *CE Shannon and J. McCarthy*, 1951.

[21] Lothaire. *Combinatorics on words*, volume 17. Cambridge university press, 1997.

[22] Bjorn Poonen. Undecidable problems: a sampler. *Interpreting Gödel: Critical Essays*, page 211, 2014.

[23] Emil Post. A variant of a recursively unsolvable problem. 1946.

[24] Matthieu Rosenfeld. How far away must forced letters be so that squares are still avoidable? *Mathematics of Computation*, 89(326):3057–3071, 2020.

[25] Keijo Ruohonen. On some variants of post's correspondence problem. *Acta Informatica*, 19:357–367, 1983.

[26] Arseny Shur. Growth properties of power-free languages. *Computer Science Review*, 6(5):187–208, 2012.

[27] Michael Sipser. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29, 1996.

[28] Axel Thue. Uber unendliche zeichenreihen. *Norske Vid Selsk. Skr. I Mat-Nat Kl.(Christiana)*, 7:1–22, 1906.

[29] Boris Trakhtenbrot. Finite automata and monadic second order logic. *Siberian Math. J*, 3:101–131, 1962.

[30] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58(345-363):5, 1936.

[31] David Wood. Nonrepetitive graph colouring. *The Electronic Journal of Combinatorics*, 1000, September 2021.