An algorithm to test if a given circular HDOL-language avoids a pattern

Julien Cassaigne

L.I.T.P., Institut Blaise Pascal, Paris

Abstract

To prove that a pattern p is avoidable on a given alphabet, one has to construct an infinite language L that avoids p. Usually, L is a DOL-language (obtained by iterating a morphism h) or a HDOL-language (obtained by coding a DOL-language with another morphism g). Our purpose is to find an algorithm to test, given a HDOL-system G, whether the language $L(G)$ generated by this system avoids p. We first define the notions of *circular morphism*, *circular DOL-system* and *circular HDOL-system*, and we show how to compute the *inverse image* of a pattern by a circular morphism. Then we prove that by computing successive inverse images of p, we can decide whether the language $L(G)$ avoids p for any fixed pattern p (which may even contain constants), provided that the HDOL-system G is circular and expansive.

1 Introduction

The theory of avoidable patterns, introduced by Zimin [13] and Bean, Ehrenfeucht and McNulty [2], generalizes problems studied by Axel Thue [12] and many others, such as the existence of square-free or overlap-free words on a given alphabet. A major question in this theory is the following: given a pattern p and an alphabet Σ, is it possible to decide whether p is avoidable on Σ ?

To prove that a pattern p is avoidable, one has to build an infinite word (or an infinite language) that avoids p. All known examples use infinite words obtained as fixed points of morphisms, or morphic images of these words; in other terms, DOL-languages and HDOL-languages. It seems reasonable to conjecture that this is a general situation, i.e. that every k-avoidable pattern is k-HDOL-avoidable [7]. Note however that there are k-avoidable patterns that are not k-DOL-avoidable, e.g. $p = AABBA$ with $k = 2$ [5].

In this paper, we construct an algorithm which, given a HDOL-system G in a certain class (G must be circular and expansive) and a pattern p with or without constants, decides whether the HDOL-language generated by G avoids p or not.

This algorithm gives the possibility to prove automatically the k-avoidability of many patterns, whereas this kind of proof is always very long to write down. But it also has a theoretical interest, as it links the two problems mentioned above: if the conjecture is true, deciding if a pattern is k-avoidable is just a matter of finding a HDOL-language that avoids it.

2 Preliminaries

2.1 Patterns

First, let us recall usual definitions about patterns:

Let Σ and Δ be two alphabets; elements of Σ are called *letters* or *constants*, and elements of Δ are called *variables*. In examples, we shall use the alphabets $\Sigma_2 = \{a, b\}$, $\Sigma_3 = \{a, b, c\}$, etc., and $\Delta = \{A, B, C, \ldots\}$. A *word* is a finite sequence of letters, i.e. an element of the free monoid Σ^* generated by Σ, and a *pattern* is an element of Δ^*. A *pattern with constants* is an element of $(\Delta \cup \Sigma)^*$, with variables and constants mixed.

Let w be a word, and p a pattern. We say that w *contains* p if there exists a non-erasing morphism $\varphi\colon \Delta^* \to \Sigma^*$ such that $\varphi(p)$ is a factor of w (a morphism φ is called *non-erasing* if $\varphi(X)$ is a non-empty word for every $X \in \Delta$). Otherwise, we say that w *avoids* p. This notion can be extended to patterns with constants; in this case, φ is a non-erasing morphism from $(\Delta \cup \Sigma)^*$ to Σ^* such that $\varphi(x) = x$ for every letter $x \in \Sigma$.

If there are arbitrarily long words in Σ^* which avoid p, we say that p is *avoidable* on Σ, or *k-avoidable* if k is the cardinal of Σ. Otherwise, we say that p is *unavoidable* on Σ. For instance, the pattern AA is unavoidable on Σ_2, whereas the pattern AAA is avoidable on Σ_2 [12].

If $\mathscr{L}$ is a family of languages on Σ, we say that a pattern p is *$\mathscr{L}$-avoidable* on Σ if there is an infinite language $L \in \mathscr{L}$ such that every word in L avoids p (in this case we say that the language L itself avoids p).

2.2 DOL and HDOL languages

Let Σ be an alphabet, $h\colon \Sigma^* \to \Sigma^*$ be a morphism in Σ^* and w_0 a word on Σ: the triplet $G = (\Sigma, h, w_0)$ is called a *DOL-system* and the language $L(G) = \{\ h^n(w_0) \mid n \in \mathbb{N}\ \}$ is called the *DOL-language* generated by this system.

If $g\colon \Sigma^* \to \Sigma'^*$ is a second morphism, where Σ' is another alphabet (usually smaller then Σ), then $G' = (\Sigma, h, w_0, \Sigma', g)$ is called a *HDOL-system* and the language $L(G') = g(L(G))$ is a *HDOL-language*. The morphism g is the *coding morphism* of G' and the DOL-system G is the *underlying DOL-system.*

Using the above definition of $\mathscr{L}$-avoidable patterns, we can define k-DOL-avoidable (resp. k-HDOL-avoidable) patterns as patterns that can be avoided by infinite DOL-languages (resp. HDOL-languages) on the k-letter alphabet Σ.

2.3 Expansive DOL-systems

If a morphism h is such that the image of a letter is always a word of length at least 2, we say that h is *strictly growing.* A DOL-system $G = (\Sigma, h, w)$ such that h^n is a strictly growing morphism for some integer n is called an *expansive* DOL-system. Under this condition, for every letter $a \in \Sigma$, the DOL-language $L(\Sigma, h, a)$ is infinite.

We say that a HDOL-system is *expansive* if the underlying DOL-system is expansive.

3 Circular morphisms

3.1 Circularity on Σ^*

Let $h: \Sigma^* \to \Sigma'^*$ be an injective morphism. This morphism is called *circular* if $C = h(\Sigma)$ is a circular code, i.e.

$$\begin{aligned}&\forall u \in \Sigma^*, \forall v \in \Sigma^*, \forall x \in \Sigma, \forall t \in \Sigma'^*, \forall s \in \Sigma'^+,\\ &\quad [(h(x) = ts \text{ and } h(u) = sh(v)t) \Longrightarrow (t = \varepsilon \text{ and } u = xv)]\,.\end{aligned}$$

Let $w \in \Sigma'^*$. We say that (w_1, w_2) is a *synchronization point* of w (for h) if $w = w_1w_2$ and

$$\forall v_1 \in \Sigma'^*, \forall v_2 \in \Sigma'^*, [v_1wv_2 \in C^* \Longrightarrow (v_1w_1 \in C^* \text{ and } w_2v_2 \in C^*)]$$

and we denote this fact by $w = w_1|w_2$. In this notation, the symbol $|$ is not related to divisibility; instead, it should be understood as a *measure bar* indicating a place in w that always match the boundary between two elements of C whenever w appears as a factor of an element of C^*. It is permitted to put several bars in the same word : $w = w_1|w_2|w_3$ just means that (w_1, w_2w_3) and (w_1w_2, w_3) are two synchronization points of w. Remark that if w is not a factor of any word in C^*, all pairs (w_1, w_2) such that $w = w_1w_2$ are trivially synchronization points of w, which means that bars can be inserted everywhere in w.

We shall use the following characterization of circular morphisms [3, Theorem 7.2.6]:

Proposition 1. *An injective morphism $h: \Sigma^* \to \Sigma'^*$ is circular if and only if every sufficiently large word in Σ'^* has at least one synchronization point.*

We say that a circular morphism h has *synchronization delay* L if every word which length is greater than or equal to L has at least one synchronization point for h.

Let S_h be the set of non-empty words that have at most one synchronization point; if h is circular with synchronization delay L, S_h is clearly finite and the length of its elements is at most $2L$. To every word $w \in \Sigma'^+ \setminus S_h$ which is a factor of at least one word in C^*, we can associate three words $c_w \in \Sigma^+$, $d_w \in \Sigma'^*$, $e_w \in \Sigma'^*$ such that $|d_w| \leqslant L$, $|e_w| \leqslant L$ and $w = d_w|h(c_w)|e_w$, i.e. $(d_w, h(c_w)e_w)$ and $(d_w h(c_w), e_w)$ are synchronization points of w.

Examples:

If h is injective and C is a comma-free code, then h is a circular morphism and its synchronization delay is $L \leqslant 2\max\limits_{u\in C}|u| - 2$.

Thue-Morse's morphism:

$$\begin{array}{rccc} \theta\colon & \Sigma_2^* & \longrightarrow & \Sigma_2^* \\ & a & \longmapsto & ab \\ & b & \longmapsto & ba \end{array}$$

is not circular, as $(ab)^n$ has no synchronization point for any n.

The morphism

$$\begin{array}{rccc} \nu\colon & \Sigma_2^* & \longrightarrow & \Sigma_2^* \\ & a & \longmapsto & aab \\ & b & \longmapsto & ba \end{array}$$

is circular, with synchronization delay 5: the only words without synchronization points are ε, a, b, aa, ab, ba, aba, baa, $abaa$ (because of the synchronization points $a|aa$, $|aab|$, $b|b$, $|ba|b$). S_ν contains these words and bb, aaa and abb.

3.2 Circularity on U

Thue-Morse's morphism is not circular, but this is only due to the fact that some very particular words (the powers of a or b) have images that do not synchronize. We would like to say that θ is circular except for this words. The purpose of the present section is to give a precise signification to this sentence.

Let $h\colon \Sigma^* \to \Sigma'^*$ be a morphism, not necessarily injective, and $U \in \mathscr{P}(\Sigma^*)$ a factorial langage (the following definition is useful only when U is infinite).

Let w be a word in Σ'^*. We say that (w_1, w_2) is a synchronization point of w (for h) *on* U if $w = w_1 w_2$ and

$$\begin{array}{l} \forall v_1 \in \Sigma'^*, \forall v_2 \in \Sigma'^*, \forall u \in U, \exists u_1 \in U, \exists u_2 \in U, \\ \quad [v_1 w v_2 = h(u) \Longrightarrow (u = u_1 u_2 \text{ and } v_1 w_1 = h(u_1) \text{ and } w_2 v_2 = h(u_2))] \end{array}$$

and we denote this by $w = w_1|_U w_2$, or simply $w = w_1|w_2$ if the choice of U is clear. If h is *injective on U*, which means that two distincts elements of U always have distinct images by h, the above definition is equivalent to

$$\forall v_1 \in \Sigma'^*, \forall v_2 \in \Sigma'^*, [v_1 w v_2 \in h(U) \Longrightarrow (v_1 w_1 \in h(U) \text{ and } w_2 v_2 \in h(U))]$$

and in particular, when $U = \Sigma^*$, we recognize the definition of synchronization points givent in the previous section.

We say that the morphism h is *circular on U* if h is injective on U and every long enough word in Σ'^* has at least one synchronization point on U. A morphism is therefore circular if and only if it is circular on Σ^*. It is also possible to define the *synchronization delay on U* of a morphism.

We denote by $S_h(U)$ the set of non-empty words that have at most one synchronization point on U. If h is circular on U with synchronization delay L, then $S_h(U)$ is finite and every word $w \in F(h(U)) \setminus (S_h(U) \cup \{\varepsilon\})$, where $F(h(U))$ denotes the set of factors of words in $h(U)$, can be written as $w = d_w|_U h(c_w)|_U e_w$ with $c_w \in U$, $d_w \in \Sigma'^*$ and $e_w \in \Sigma'^*$ such that $c_w \neq \varepsilon$, $|d_w| \leqslant L$ and $|e_w| \leqslant L$.

With these definition, if U is the set of words in Σ_2^* that contain neither a^3 nor b^3, we can say that Thue-Morse's morphism is circular on U with synchronization delay 4: indeed, there are new synchronization points $ab|_U ab$ and $ba|_U ba$, so that $S_h(U) = \{a, b, aa, ab, ba, bb, aba, bab\}$.

Finally, let us remark that

Proposition 2. *If U and U' are two factorial languages of Σ^* such that $U' \subset U$, and $h: \Sigma^* \to \Sigma'^*$ is a morphism, then every synchronization point for h on U is also a synchronization point on U'. If h is circular on U with synchronization delay L, then h is also circular on U', with synchronization delay less than or equal to L.* ∎

3.3 Circular L-systems

In the case of DOL-systems, there is a natural choice for the language U: the set of factors of the langage generated by the DOL-system itself. Thus, we shall say that a DOL-system $G = (\Sigma, h, w_0)$ is *circular* if the morphism h is circular on $F(L(G))$. This notion coincides with the *circular DOL-langages* of Mignosi and Séébold [9] and the *recognizable substitutions* of Mossé [10], at least when the morphism is injective.

For a HDOL-system, the natural choice is the same as for the underlying DOL-system: we shall say that the HDOL-system $G = (\Sigma, h, w_0, \Sigma', g)$ is *circular* if the morphisms h and g are both circular on $F(L(\Sigma, h, w_0))$.

Examples:

Let $\theta: \Sigma_2^* \to \Sigma_2^*$ be Thue-Morse's morphism. We know that the language generated by the HDOL-system $G = (\Sigma_2, \theta, a)$ is cube-free, and is therefore contained in $U = \Sigma_2^* \setminus \Sigma_2^*\{a^3, b^3\}\Sigma_2^*$. We have seen above that θ is circular on U, hence by Proposition 2 it is also circular on $F(L(G))$, which proves that G is a circular DOL-system.

Let $G = (\Sigma_3, \mu, a, \Sigma_2, \psi)$ be the HDOL-system used by Roth [11] to prove that the pattern $ABAAB$ is 2-avoidable. The first morphism,

$$\begin{array}{rrcl} \mu\colon & \Sigma_3^* & \longrightarrow & \Sigma_3^* \\ & a & \longmapsto & abc \\ & b & \longmapsto & ac \\ & c & \longmapsto & b \end{array}$$

is circular; the second one,

$$\begin{array}{rrcl} \psi\colon & \Sigma_3^* & \longrightarrow & \Sigma_2^* \\ & a & \longmapsto & aaa \\ & b & \longmapsto & bbb \\ & c & \longmapsto & ababab \end{array}$$

is not circular, but as $U = F(L(\Sigma_3, \mu, a))$ is a square-free langage it is possible to prove that ψ is circular on U, with synchronization delay 7.

We can now give our main result, which will be proved in Section 5 :

Theorem 1. *There exists an algorithm which, given a circular and expansive HDOL-system G and a pattern with constants p, decides whether the language generated by G avoids the pattern p.*

In the case of patterns without constants, it is possible to strengthen this result using a theorem of Mignosi and Séébold [9]: every non-repetitive DOL-system is circular.

Theorem 2. *There exists an algorithm which, given a HDOL-system $G = (\Sigma, h, a, \Sigma', g)$ such that a is a letter of Σ and g is circular on $F(L(\Sigma, h, a))$, and a pattern without constants p, decides whether the language $L(G)$ avoids the pattern p.*

Proof: If h is not circular on $U = F(L(\Sigma, h, a))$, the language $L(\Sigma, h, a)$ is repetitive by [9, Theorem 1, Corollary], hence $L(G)$ is also repetitive as g is non-erasing, which means that $L(G)$ contains arbitrary powers and cannot avoid p.

If G is circular and expansive, we can apply Theorem 1. The case where G is not expansive will be studied in Section 6, where we need the fact that the axiom is just a letter. ∎

For DOL-systems, this theorem reduces to:

Corollary 1. *There exists an algorithm which, given a DOL-system G whose axiom is a letter, and a pattern without constants p, decides whether the language $L(G)$ avoids the pattern p.*

The only remaining restriction is that the axiom must be a letter, which is not very annoying.

Remark: Carpi obtained independently a similar result [4, Proposition 7.4] for expansive DOL-systems, using a quite different method, based on *representation systems*, a powerful generalization of numeration systems that has many other applications.

4 The inverse image of a pattern by a circular morphism

Notation: Let $p \in (\Delta \cup \Sigma')^*$ be a pattern with constants in an alphabet Σ'. We denote by $[p]_{\Sigma'}$ the set of the words in ${\Sigma'}^*$ which contain an occurrence of p. If no ambiguity is possible, we simply denote this set by $[p]$.

With this notation, a language $L \subset {\Sigma'}^*$ avoids a pattern p if and only if $F(L) \cap [p] = \emptyset$, and p is avoidable on Σ' if and only if ${\Sigma'}^* \setminus {\Sigma'}^*[p]{\Sigma'}^*$ is an infinite language.

We call *valuation* on an alphabet Σ a monoid morphism from Σ^* to the additive group $(\mathbb{R}, +)$, and the image of a word u by a valuation is usually written $\|u\|$. Usual valuations are the length $|u|$ and the number of occurrences $|u|_a$ of a letter a, and a valuation is always a linear combination of these. For instance, if Σ' is a subset of Σ, the application mapping u to $|u|_{\Sigma'} = \sum_{a \in \Sigma'} |u|_a$ is a valuation. A valuation with values in $\mathbb{R}^+$ is a *positive valuation*.

Proposition 3. *Let Σ and Σ' be two alphabets with positive valuations, $U \in \P(\Sigma^*)$ be a factorial langage, $h: \Sigma^* \to {\Sigma'}^*$ be a morphism circular on U, m and M be two real numbers such that $0 < m \|x\|_\Sigma \leqslant \|h(x)\|_{\Sigma'} \leqslant M$ for all $x \in \Sigma$, L be a real number such that every word in ${\Sigma'}^*$ with valuation at least L has a synchronization point on U, and such that the valuation of the elements of $S_h(U)$ is at most $2L$, and $p \in (\Delta \cup \Sigma')^*$ be a pattern with constants in Σ'. Then there exists a finite set $Q \subset (\Delta \cup \Sigma)^*$ of patterns with constants in Σ such that:*

(i) $h^{-1}([p]_{\Sigma'}) \cap U = \bigcup_{q \in Q} [q]_\Sigma \cap U$

(ii) $\forall q \in Q, |q|_\Delta \leqslant |p|_\Delta$ and $\|q\|_\Sigma \leqslant \frac{2M}{m} + \frac{1}{m}\|p\| + \frac{2L}{m}|p|_\Delta$

where the valuations of Σ and Σ' are extended respectively to $\Delta \cup \Sigma$ and $\Delta \cup \Sigma'$ by $\|X\|_\Sigma = \|X\|_{\Sigma'} = 0$ for $X \in \Delta$.

In most cases we simply take $\|u\| = |u|$, and L is the synchronization delay of h on U. If moreover $U = \Sigma^*$, we get the following simplified version :

Proposition 4. *Let $h: \Sigma^* \to \Sigma'^*$ be a circular morphism od synchronization delay L, m and M be two integers such that $1 \leqslant m \leqslant |h(x)| \leqslant M$ for all $x \in \Sigma$, and $p \in (\Delta \cup \Sigma')^*$ be a pattern with constants in Σ'. Then there exists a finite set $Q \subset (\Delta \cup \Sigma)^*$ of patterns with constants in Σ such that:*

(i) $h^{-1}([p]_{\Sigma'}) = \bigcup_{q \in Q} [q]_\Sigma$

(ii) $\forall q \in Q, |q|_\Delta \leqslant |p|_\Delta$ and $|q|_\Sigma \leqslant \frac{2M}{m} + \frac{1}{m}|p|_{\Sigma'} + \frac{2L}{m}|p|_\Delta$

■

Proof: Let $w \in h^{-1}([p]_{\Sigma'}) \cap U$. We suppose that w is minimal, which means that no proper factor of w is in $h^{-1}([p]_{\Sigma'})$. If $p = u_0 X_1 u_1 X_2 \ldots X_r u_r$ with $X_i \in \Delta$ and $u_i \in \Sigma'^*$, then

$$h(w) = v_1 u_0 \varphi(X_1) u_1 \varphi(X_2) u_2 \ldots \varphi(X_r) u_r v_2$$

with $v_i \in \Sigma'^*$ and $\varphi: \Delta \to \Sigma'^+$. As w is minimal, $\|v_i\|_{\Sigma'} < M$.

Let $\Delta' = \{\, X \in \Delta \mid \varphi(X) \notin S_h(U) \,\}$. Let $i_1,\ldots,i_s$ be the indices i such that $X_i \in \Delta'$, $i_0 = 0$, $i_{s+1} = r+1$, $Y_j = X_{i_j}$ and

$$u'_j = u_{i_j} \varphi(X_{i_j+1}) u_{i_j+1} \varphi(X_{i_j+2}) u_{i_j+2} \ldots \varphi(X_{i_{j+1}-1}) u_{i_{j+1}-1}$$

for $0 \leqslant j \leqslant s$. Then $h(w)$ can also be written as

$$h(w) = v_1 u'_0 \varphi(Y_1) u'_1 \varphi(Y_2) u'_2 \ldots \varphi(Y_s) u'_s v_2 \ .$$

As $\varphi(Y_j) \in F(h(U)) \setminus (S_h(U) \cup \{\varepsilon\})$, we have

$$\varphi(Y_j) = d_{\varphi(Y_j)} h(c_{\varphi(Y_j)}) e_{\varphi(Y_j)} = d_j h(\psi(Y_j)) e_j$$

where $\psi: \Delta' \to \Sigma^+$ is the morphism defined by $\psi(Y) = c_{\varphi(Y)}$.

Because of the synchronization points $d_j|_U h(\psi(Y_j))|_U e_j$, the $s+1$ words $v_1 u'_0 d_1$, $e_j u'_j d_{j+1}$ (for $1 \leqslant j \leqslant s-1$) and $e_s u'_s v_2$ are elements of $h(U)$ hence we can apply h^{-1}, which is well-defined on $h(U)$ as h is injective on U, to get:

$$w = h^{-1}(v_1 u'_0 d_1) \psi(Y_1) h^{-1}(e_1 u'_1 d_2) \ldots \psi(Y_s) h^{-1}(e_s u'_s v_2)$$

therefore $w \in [q]_\Sigma$ with

$$q = h^{-1}(v_1 u'_0 d_1) Y_1 h^{-1}(e_1 u'_1 d_2) \ldots Y_s h^{-1}(e_s u'_s v_2) \ .$$

If $w' \in h^{-1}([p]_{\Sigma'})$ is not minimal and contains w, then we also have $w' \in [q]_\Sigma$.

Reciprocally, if $w' \in [q]_\Sigma$, it can be written

$$w' = v'_1 h^{-1}(v_1 u'_0 d_1) \psi'(Y_1) h^{-1}(e_1 u'_1 d_2) \ldots \psi'(Y_s) h^{-1}(e_s u'_s v_2) v'_2$$

with $\psi'\colon \Delta' \to \Sigma^+$, and then $h(w') \in [p]_{\Sigma'}$ as $h(w')$ contains $\varphi'(p)$ where $\varphi'\colon \Delta \to \Sigma'^+$ is defined by

$$\begin{cases} \varphi'(X) & = & \varphi(X) & \text{if } X \notin \Delta' \\ \varphi'(X) & = & d_{\varphi(X)} h\left(\psi'(X)\right) e_{\varphi(X)} & \text{if } X \in \Delta'. \end{cases}$$

Let us denote by Q the set of all the patterns q obtained by this method for all possible choices of w (this set can *a priori* be infinite): *(i)* is clearly verified.

Let us now prove *(ii)*. According to the construction of q,

$$\begin{aligned} |q|_\Delta &= s \leqslant r = |p|_\Delta \\ \|q\|_\Sigma &= \left\|h^{-1}\left(v_1 u_0' d_1 e_1 u_1' d_2 e_2 u_2' \ldots d_s e_s u_s' v_2\right)\right\|_\Sigma \\ &\leqslant \frac{1}{m} \|v_1 u_0' d_1 e_1 u_1' d_2 e_2 u_2' \ldots d_s e_s u_s' v_2\|_{\Sigma'} \\ &\leqslant \frac{1}{m}\Big(\|v_1 v_2\|_{\Sigma'} + \|d_1 e_1 d_2 e_2 \ldots d_s e_s\|_{\Sigma'} + \|u_0 u_1 \ldots u_r\|_{\Sigma'} \\ &\qquad + \sum_{\substack{1 \leqslant i \leqslant r \\ X_i \notin \Delta'}} \|\varphi\left(X_i\right)\|_{\Sigma'} \Big) \\ &\leqslant \frac{1}{m}(2M + 2sL + \|p\|_{\Sigma'} + (r-s)2L) \end{aligned}$$

hence *(ii)* is verified.

As the number of patterns verifying *(ii)* is finite, the set Q is necessarily finite. ■

The set Q is denoted $Q = h_U^{-1}(p)$, or simply $Q = h^{-1}(p)$, and is called the *inverse image* of the pattern p by the morphism h in U. This set is not uniquely defined, but what is important is that we have an effective method to construct such a set. In particular, if an element of Q is contained in another element of Q, it is possible to suppress the latter without changing the properties *(i)* and *(ii)*. In practice, we will try to minimize the size of Q to spare computations.

If P is a finite set of patterns with constants in Σ', we naturally denote by $h_U^{-1}(P)$ the set $h_U^{-1}(P) = \bigcup_{p \in P} h_U^{-1}(p)$, which is also a finite set, and by $[P]_{\Sigma'}$ the set $\bigcup_{p \in P} [p]_{\Sigma'}$.

5 The algorithm

We can now state our algorithm. Recall that the input is:

- $G = (\Sigma, h, w_0, \Sigma', g)$, a circular and expansive HDOL-system;
- $p \in (\Delta \cup \Sigma')^*$, a pattern with constants.

and that the algorith has to determine if the HDOL-language generated by G avoids or contains p. In this section, it will be understood that all inverse images of patterns by morphisms are in $U = F(L(\Sigma, h, w_0))$.

For every integer j, we denote by Q_j the finite set $Q_j = h^{-j}\left(g^{-1}(p)\right)$, which can be computed starting with $Q_0 = g^{-1}(p)$ by mean of the recurrence formula $\forall j \in \mathbb{N}, Q_{j+1} = h^{-1}(Q_j)$. Let $Q = \bigcup_{j=0}^{\infty} Q_j$, so that

$$[Q]_\Sigma \cap U = \bigcup_{j=0}^{\infty} h^{-j}\left(g^{-1}\left([p]_{\Sigma'}\right)\right) \cap U \ .$$

Using the fact that the HDOL-system G is expansive, we can prove that the set Q is always finite: Let us first suppose that h is strictly growing: let m and M be integers such that for every letter $x \in \Sigma$, $2 \leqslant m \leqslant |h(x)| \leqslant M$, and let

$$\alpha = \max_{q \in Q_0} |q|_\Sigma$$

and

$$\beta = \max\left(\alpha, \frac{2M + 2L\,|p|_\Delta}{m-1}\right) \ .$$

We prove by induction on j that

$$\forall j \in \mathbb{N}, \forall q \in Q_j, |q|_\Delta \leqslant |p|_\Delta \text{ and } |q|_\Sigma \leqslant \beta \ .$$

This is obvious when $j = 0$, and if this inequality holds for j, then according to inequality *(ii)* of Proposition 3, with the length as valuation on Σ, we have for all $q \in Q_{j+1} = h^{-1}(Q_j)$:

$$|q|_\Delta \leqslant |p|_\Delta \text{ and } |q|_\Sigma \leqslant \frac{2M}{m} + \frac{1}{m}\beta + \frac{2L}{m}|p|_\Delta \leqslant \beta \ .$$

Therefore

$$\forall q \in Q, |q|_\Delta \leqslant |p|_\Delta \text{ and } |q|_\Sigma \leqslant \beta \ ,$$

which implies that Q is finite.

If G is expansive but h is not strictly growing, there exists a valuation on Σ such that $\|h(x)\| > \|x\|$ for every letter x. Indeed, there exists an integer n such that h^n is strictly growing. For every $x \in \Sigma$, there exists an integer n_x between 0 and $n-1$ such that $|h^{n_x}(x)| = 1$ and $|h^{n_x+1}(x)| \geqslant 2$. We define a valuation on Σ by $\|x\| = 1 - \frac{n_x}{2n}$. If $n_x > 0$,

$$\|h(x)\| = 1 - \frac{n_{h(x)}}{2n} = 1 - \frac{n_x - 1}{2n} = \|x\| + \frac{1}{2n}$$

and if $n_x = 0$, the word $h(x)$ contains at least two letters y and z and

$$\|h(x)\| \geqslant \|y\| + \|z\| = 2 - \frac{n_y + n_z}{2n} \geqslant 2 - \frac{2(n-1)}{2n} = 1 + \frac{1}{n} \ .$$

In both cases, we have $\|h(x)\| \geqslant m\,\|x\|$ with $m = 1 + \frac{1}{2n}$. We can the proceed as above, using the valuation we have just defined instead of length. As $m > 1$, we can also conclude that Q is finite.

To construct the set Q, we start with Q_0 and compute successively Q_1, Q_2, etc. using the recurrence formula $Q_{j+1} = h^{-1}(Q_j)$, until we find that $Q_{j+1} \subset \bigcup_{i=0}^{j} Q_i$, and then $Q = \bigcup_{i=0}^{j} Q_i$.

If a pattern $q \in Q$ is contained in w_0, it means that there exists j such that $w_0 \in h^{-j}(g^{-1}([p]_{\Sigma'}))$. Then $g(h^j(w_0))$ contains p, and we can conclude that $L(G)$ does not avoid p. Reciprocally, if w_0 avoids every element of Q, it means that none of the words $g(h^j(w_0))$ contains p, i.e. that $L(G)$ avoids p. Therefore, as Q is finite, we can effectively test whether $L(G)$ avoids p.

```
algorithm test-circular-HDOL
input : a circular expansive HDOL-system G = (Σ, h, w₀, Σ', g)
        a pattern with constants p ∈ (Δ ∪ Σ')*
output : a boolean, indicating whether the HDOL-language L(G) avoids p
variables : two sets of patterns with constants Q and R

begin
    Q ← ∅
    R ← g⁻¹(p)
    while R ≠ ∅ do
        choose q ∈ R and remove q from R
        if q is divisible by no other element of Q, then
            if w₀ contains q,
                then return false
                else append q to Q and h⁻¹(q) to R
    return true
End.
```

Figure 1: Main algorithm

In practice, we can proceed as indicated on Figure 1. The variable R is a stack containing patterns waiting to be processed. At the end of the loop, i.e. when there is no pattern left in R, the variable Q contains the set $Q = \bigcup_{i=0}^{\infty} Q_i$ defined above. We added two ameliorations: first, if a pattern is divisible by a pattern already in Q, it can be discarded as it cannot yield any new pattern not divisible by a pattern in Q (for this to be efficient, we need a strategy to select patterns in R that advantages those which are more likely to divide other patterns; the simplest such strategy is to choose small patterns first); second, if we find a pattern contained in w_0, it is useless to compute the set Q entirely. In Appendix 7, we demonstrate how this algorithm works on a simple example, in particular how inverse images can be computed.

6 Reduction to expansive morphisms

It is sometimes useful to consider HDOL-systems $G = (\Sigma, h, w_0, \Sigma', g)$ that are not expansive; a common case is when there is a letter $b \in \Sigma$ such that $h(b) = b$. Of course, the algorithm may still be used, but then there is no warranty that it will ever stop. Another solution is to construct a new HDOL-system $\widetilde{G} = (\widetilde{\Sigma}, \widetilde{h}, \widetilde{w}_0, \Sigma', \widetilde{g})$ which is expansive and equivalent to G, in the sense that the generated languages have the same factors, and to apply the algorithm to this new system.

We shall assume that the circular HDOL-system G verifies the following additional conditions:

- w_0 is a letter which is not of rank 0 (a letter of rank 0 is a letter a such that the set $L(\Sigma, h, a)$ is finite; only non-expansive L-systems have such letters).
- G is not pushy, i.e. the set of words in $L(\Sigma, h, a)$ consisting entirely of letters of rang 0 is finite.

The first restriction is not very important, as HDOL-avoidable patterns can always be avoided by this kind of HDOL-languages [6, Theorem 3.2]. The second one is always satisfied for useful HDOL-systems, because a pushy HDOL-system is always repetitive [8] and avoids no pattern without constants.

First, here are two useful properties of circular morphisms.

Lemma 1. *Let $f: \Sigma^* \to \Sigma'^*$ and $g: \Sigma'^* \to \Sigma'^*$ be two morphisms. Suppose that f is circular on a factorial language $U \in \P(\Sigma^*)$ and that g is circular on $V = F(f(U))$. Then $g \circ f$ is circular on U.*

Proof: Let M be the maximal length of the image of a letter by g, and L_f and L_g be the synchronization delays of f and g on U and V, respectively. We shall proove that $g \circ f$ is circular on U with synchronization delay $L = 2L_g + ML_f$.

Let w be a word in Σ'^L. We have: $w = d_w|_V g(c_w)|_V e_w$ with $|d_w| \leqslant L_g$, $|e_w| \leqslant L_g$ and $|c_w| \geqslant L_f$. The word c_w has therefore a synchronization point $c_w = u_1|_U u_2$ for f, and from this we deduce easily that w has a synchronization point $w = d_w g(u_1)|_U g(u_2) e_w$ for $g \circ f$. As $g \circ f$ is injective on U, this shows that $g \circ f$is circular on U. ■

Lemma 2. *Let $f: \Sigma^* \to \Sigma'^*$ and $g: \Sigma'^* \to \Sigma'^*$ be two morphisms. Suppose that $g \circ f$ is circular on a factorial language $U \in \P(\Sigma^*)$ and that g is injective on $V = F(f(U))$. Then f is circular on U and g is circular on V.*

Proof: Let L be the synchronization delay of $g \circ f$ on U. Let u be a word in Σ^* long enough so that $|g(f(u))| \geqslant L$; the word $g(f(u))$ than has a synchronization point $g(f(u)) = z_1|_U z_2$ for $g \circ f$, hence u can be factorized as $u = u_1 u_2$ in such a way that $z_i = g(f(u_i))$.

We then have $g(f(u)) = z_1|_V z_2$ for g. Suppose indeed that $v \in V$ and $w_1, w_2 \in W = F(g(v))$ are such that $g(v) = w_1 g(f(u)) w_2$; v itself is a factor of a word in $f(U)$, say $f(u') = v_1 v v_2$. Then $g(f(u')) = g(v_1) w_1 z_1 z_2 w_2 g(v_2)$, hence there exist two words $u'_1, u'_2 \in U$ such that $u' = u'_1 u'_2$, $g(f(u'_1)) = g(v_1) w_1 z_1$ and $g(f(u'_2)) = z_2 w_2 g(v_2)$. The length of $f(u'_i)$ is greater than or equal to that of v_i, so there exist two words $v'_1, v'_2 \in V$ such that $f(u'_1) = v_1 v'_1$ and $f(u'_2) = v'_2 v_2$. Finally, we have $g(v'_1) = w_1 z_1$ and $g(v'_2) = z_2 w_2$ with $v = v'_1 v'_2$, which is what we wanted to prove.

If now w is a long enough word in Σ'^*, either w is not in W and we can insert synchronization points for g on V everywhere in w, or w has a factor of the form $g(f(u))$ which has, as we have seen, a synchronization point for g. The morphism g is therefore circular on V.

Similarly, let us show that $f(u) = f(u_1)|_U f(u_2)$ for f. Suppose that $u' \in U$ and $v_1, v_2 \in V$ are such that $f(u') = v_1 f(u) v_2$. Then $g(f(u')) = g(v_1) g(f(u)) g(v_2)$ hence there exist two words $u'_1, u'_2 \in U$ such that $u' = u'_1 u'_2$, $g(f(u'_1)) = g(v_1) g(f(u_1))$ and $g(f(u'_2)) = g(f(u_2)) g(v_2)$. As g is injective on V, we deduce that $f(u'_1) = v_1 f(u_1)$ and $f(u'_2) = f(u_2) v_2$. As above, we conclude that f is circular on U. ■

We can now construct the new system $\widetilde{G}$, and prove that it is circular.

Let Σ_0 be the set of letters of rank 0, and Σ_1 its complement. We denote by G^0 the DOL-system underlying G. Consider the set $\widetilde{\Sigma}$ of words in $\Sigma_1 \Sigma_0^*$ that appear in $L(G^0)$, either at the end of a word, or followed by a letter in Σ_1, which is finite according to the hypotheses. This set will be taken as a new alphabet; it can be much larger than Σ. To distinguish elements of $\widetilde{\Sigma}$ (considered as letters) and the associated words in Σ^*, the former will be denoted by $[av]$ with $a \in \Sigma_1$ and $v \in \Sigma_0^*$.

Let i be the morphism defined by

$$i\colon \begin{array}{ccc} \widetilde{\Sigma}^* & \longrightarrow & \Sigma^* \\ [av] & \longmapsto & av \end{array} .$$

Then there exist morphisms $\widetilde{h}\colon \widetilde{\Sigma}^* \to \widetilde{\Sigma}^*$ and $\widetilde{g}\colon \widetilde{\Sigma}^* \to \Sigma'^*$ such that the following diagram is commutative:

$$\begin{tikzcd} \widetilde{\Sigma}^* \arrow[r, "\widetilde{h}"] \arrow[d, "i"] & \widetilde{\Sigma}^* \arrow[d, "i"] \arrow[dr, "\widetilde{g}"] & \\ \Sigma^* \arrow[r, "h"] & \Sigma^* \arrow[r, "g"] & \Sigma'^* \end{tikzcd}$$

(they are defined by $\widetilde{h}([av]) = i^{-1}(h(av))$ and $\widetilde{g} = g \circ i$). If $\widetilde{w}_0 = [w_0]$, it is clear that $\widetilde{g}(\widetilde{h}^n(\widetilde{w}_0)) = g(h^n(w_0))$, hence $L(\widetilde{G}) = L(G)$ with $\widetilde{G} = (\widetilde{\Sigma}, \widetilde{h}, \widetilde{w}_0, \Sigma', \widetilde{g})$.

The new morphism $\widetilde{h}$ is expansive: for every letter $[av] \in \widetilde{\Sigma}$, the set of factors of $i(L(\widetilde{\Sigma}, \widetilde{h}, [av]))$ contains $L(\Sigma, h, a)$ which is infinite, hence $L(\widetilde{\Sigma}, \widetilde{h}, [av])$ is infinite too.

The morphism i is clearly circular (on $\widetilde{\Sigma}^*$), because letters of Σ_1 act as markers of the boundaries between images. After Lemma 1, the morphisms $i \circ \widetilde{f} = f \circ i$ and $\widetilde{g} = g \circ i$ are circular on $\widetilde{U} = F(L(\widetilde{G}^0))$. We can then apply Lemma 2 to the morphisms $\widetilde{f}$ and i (i is injective) to show that $\widetilde{f}$ is circular on $\widetilde{U}$.

7 Conclusion

We have constructed an algorithm that works for a large class of HDOL-systems. At the price of a certain complication due to the necessity of always precising the set U, we have been able to suppress some limitations on the first morphism (recall however that it has to be expansive if the pattern contains constants or the axiom is not a letter). This allows to process common morphisms that are not circular, as Thue-Morse's. In this particular case, it is also possible to circumvent the difficulty by finding a new HDOL-system $\widetilde{G} = (\Sigma_3, \mu, a, \Sigma_2, \pi)$ with circular morphisms equivalent to $G = (\Sigma_2, \theta, a)$:

$$\begin{array}{rrcl} \mu\colon & \Sigma_3^* & \longrightarrow & \Sigma_3^* \\ & a & \longmapsto & abc \\ & b & \longmapsto & ac \\ & c & \longmapsto & b \end{array}$$

and

$$\begin{array}{rrcl} \pi\colon & \Sigma_3^* & \longrightarrow & \Sigma_2^* \\ & a & \longmapsto & abb \\ & b & \longmapsto & ab \\ & c & \longmapsto & a \end{array} .$$

($\widetilde{G}$ is expansive, because h^2 is strictly growing). Maybe such a construction is possible for every DOL-system, using techniques similar to that of Section 6.

The remaining restrictions on the second morphism (it has to be circular on the set U of factors of the underlying DOL-language) are more annoying. Indeed, the most used HDOL-systems are CDOL-systems, where the second morphism is letter to letter and cannot be injective on U. For these systems, it is often possible to find an equivalent circular HDOL-system (using for instance the fact that $(\Sigma, h, w_0, \Sigma', g)$ is equivalent to $(\Sigma, h, w_0, \Sigma', g \circ f)$). It would be interesting to be able to make such a construction in all cases.

References

[1] K. A. Baker, G. F. McNulty, and W. Taylor, Growth problems for avoidable words, *Theoret. Comput. Sci.* **69** (1989), 319–345.

[2] D. R. Bean, A. Ehrenfeucht, and G. F. McNulty, Avoidable patterns in strings of symbols, *Pacific J. of Math.* **85** (1979), 261–294.

[3] J. Berstel and D. Perrin, *Theory of Codes*, Academic Press, 1985.

[4] A. Carpi, On representation systems, Preprint I.C.-R.I. 104/93, Istituto di Cibernetica, Arco Felice (NA), Italia, 1993.

[5] J. Cassaigne, Unavoidable binary patterns, *Acta Inf.* **30** (1993), 385–395.

[6] J. Cassaigne, *Motifs évitables et régularités dans les mots*, Thèse de Doctorat, Université Paris 6, 1994. Rapport LITP TH94-04.

[7] J. D. Currie, Open problems in pattern avoidance, *Amer. Math. Monthly* **100** (1993), 790–793.

[8] A. Ehrenfeucht and G. Rozenberg, Repetition of subwords in DOL languages, *Inform. and Control* **59** (1983), 13–35.

[9] F. Mignosi and P. Séébold, If a DOL language is k-power-free then it is circular, in *ICALP '93*, Lund, 1993, no. 700 in *Lect. Notes Comp. Sci.*, Springer-Verlag.

[10] B. Mossé, Puissances de mots et reconnaissabilité des points fixes d'une substitution, *Theoret. Comput. Sci.* **99** (1992), 327–334.

[11] P. Roth, Every binary pattern of length six is avoidable on the two-letter alphabet, *Acta Inf.* **29** (1992), 95–106.

[12] A. Thue, Über die gegenseitige Lage gleicher Teile gewisser Zeichenreihen, *Norske Vid. Selsk. Skr., I. Mat. Nat. Kl., Christiana* **1** (1912), 1–67.

[13] A. I. Zimin, Blocking sets of terms, *Math. USSR Sbornik* **47**,2 (1984), 353–364. English translation.

Appendix A: Example

The purpose of this appendix is to demonstrate how our algorithm can be applied to an example: show that the DOL-language $L(\Sigma_3, \mu, a)$ avoids the pattern $p = AA$, where μ is the expansive morphism

$$\begin{array}{rrcl} \mu\colon & \Sigma_3^* & \longrightarrow & \Sigma_3^* \quad . \\ & a & \longmapsto & abc \\ & b & \longmapsto & ac \\ & c & \longmapsto & b \end{array}$$

A.1 μ is circular, with synchronization delay 2

Indeed, we have the following synchronization points:

$$|a,\ c|,\ |b|b|.$$

and every word of length at least 2 contains a, c or bb as a factor.

A.2 The set S_μ

S_μ contains 9 words with 0 or 1 synchronization point:

$$|a,\ b,\ c|,\ |ab,\ bc|,\ c|a,\ bc|a,\ c|ab,\ bc|ab.$$

A.3 The complement of S_μ

The other words which are factors of some word in C^* (with $C = \mu(\Sigma_3)$) can be factored in one of the following 9 forms:

$$|\mu(w)|,\ |\mu(w)|a,\ |\mu(w)|ab,\ c|\mu(w)|,\ c|\mu(w)|a,$$
$$c|\mu(w)|ab,\ bc|\mu(w)|,\ bc|\mu(w)|a,\ bc|\mu(w)|ab.$$

A.4 Main loop

Current pattern: AA

1. We rewrite each variable (here only A) in each of the 18 ways determined in A.2 and A.3:

$$|a|a,\ bb,\ c|c|,\ |ab|ab,\ bc|bc|,$$
$$c|ac|a,\ bc|abc|a,\ c|abc|ab,\ bc|abbc|ab,$$
$$|A|A|,\ |A|a|A|a,\ |A|ab|A|ab,\ c|A|c|A|,\ c|A|ac|A|a,$$
$$c|A|abc|A|ab,\ cb|A|bc|A|,\ bc|A|abc|A|a,\ bc|A|abbc|A|ab.$$

2. We compute the (ordinary) inverse images by μ of these words. There are possibly several such inverse images (because of ambiguities at the beginning and the end of the words) or none (because of forbidden factors, i.e. words that are not factors of any word in C^*):

$$\emptyset,\ \{cc\},\ \emptyset,\ \emptyset,\ \emptyset,$$
$$\{aba, abb, bba, bbb\},\ \{aaa, aab\},\ \{aaa, baa\},\ \emptyset,$$
$$\{AA\},\ \emptyset,\ \emptyset,\ \emptyset,\ \{aAbAa, aAbAb, bAbAa, bAbAb\},$$
$$\{aAaAa, bAaAa\},\ \emptyset,\ \{aAaAa, aAaAb\},\ \emptyset.$$

3. We eliminate patterns that have already been found (AA), as well as patterns containing them (e.g. $aAbAb$); what remains is:

$$aba,\ aAbAa.$$

   Now we have to run A.4 for each of these patterns.

Current pattern: aba

1. No variable, hence no rewriting: we keep aba.
2. No inverse image, as aba is a forbidden factor.
3. Therefore we get no new pattern.

Current pattern: aAbAa

1. We rewrite the variable A :

$$\begin{gathered} a|ab|aa\,,\; abbba\,,\; ac|bc|a\,,\; a|abb|aba\,,\; abc|bbc|a\,, \\ ac|abc|aa\,,\; abc|abbc|aa\,,\; ac|abbc|aba\,,\; abc|abbbc|aba\,, \\ a|A|b|A|a\,,\; a|A|ab|A|aa\,,\; a|A|abb|A|aba\,,\; ac|A|bc|A|a\,,\; ac|A|abc|A|aa\,, \\ ac|A|abbc|A|aba\,,\; abc|A|bbc|A|a\,,\; abc|A|abbc|A|aa\,,\; abc|A|abbbc|A|aba\,. \end{gathered}$$

2. No inverse image.
3. Therefore we get no new pattern.

A.5 Result

We obtained the following patterns:

$$AA\,,\; aba\,,\; aAbAa\,.$$

This means that $Q = \{AA, aba, aAbAa\}$ is invariant by μ^{-1}. As the letter a avoids all these patterns, we can conclude:

$$L(\Sigma_3, \mu, a) \text{ avoids } p\,.$$