Undecidability of existential properties in picture languages*

D. Robilliard† D. Simplot‡

January 9, 1998

Abstract

We are interested in description of set of pictures by string languages by using several semantics: segments [12], segments with blank moves [8] and pixels [10]. We give a method to code the Post correspondence problem in rational picture languages in order to show the undecidability of existence of words satisfying a given property in a rational language.

1 Introduction

In order to use the well-known tools of theory of formal languages to manipulate multidimensional objects such like pictures, we try to lead to the free monoid by describing this objects with words. This is the subject of the "picture languages theory" introduced by Maurer, Rozenberg and Welzl [12] who use the restricted Freeman's code to describe pictures made of segments. The aim is to assimilate a word with a sequence of orders to a plotter where the commands are limited to the four orders "up", "right", "down" and "left" designed respectively by the letters u, r, d and l. Thus the words over the alphabet Π — called *picture words* — represent pictures with two distinguished points: the start point and the end point.

We are interested in the description of picture sets by (string) languages classified according to the Chomsky's hierarchy. Thus we know that the membership problem — i.e. to know if a given picture is represented in a picture of Π^* — is decidable but NP-complete for rational languages [9]. We also know how to decide if a context-free language describes a finite or an infinite set of pictures [12].

In this line of decision problems, in this paper we are interested in existential properties. Let p be a predicate over picture words, which can be about pictures or words. We would like to know whether we can decide the problem:

> *Let L be a picture word language. Is there in L a word which satisfies the property p?*

We give here a method to show the undecidability of this kind of problems for several properties p such as "is a self-avoiding word", "is a contour word of polyomino" for rational picture languages as well as for several semantics: segments like described above [12], segments with blank moves [8] and pixels [10].

2 Preliminaries and notations

We assume the reader to be familiar with the theory of formal languages and we give only some notations. For more precisions, see the books of J. Berstel [3] or of S. Eilenberg [7].

*This work was partially supported by "PRC/GDR Mathématiques et Informatique" and ESPRIT Basic Research Action 6317 ASMICS 2

†L.I.L., Univ. du Littoral, B.P. 719, 62228 Calais Cedex, France. Email: robillia@lil.univ-littoral.fr

‡L.I.F.L., Univ. Lille 1, Bât. M3, 59655 Villeneuve d'Ascq Cedex, France. Email: simplot@lifl.fr WWW home page: http://www.lifl.fr/~simplot

Let Σ be an alphabet, the set Σ^* represents the free monoid generated by the alphabet Σ, we note ε the empty word.

For a word $\omega \in \Sigma^*$, $|\omega|$ denotes its length and $|\omega|_a$ the number of occurrences of the letter a in ω. We design by $\tilde{\omega}$ the mirror image of ω. For two words $\alpha, \beta \in \Sigma^*$, we note $\alpha \leq \beta$ the fact that α is a prefix of β, i.e. there is $\alpha' \in \Sigma^*$ such that $\beta = \alpha.\alpha'$.

The wording of the *Post correspondence problem* (noted PCP), is a couple (U, V) with $U = (u_1, \ldots, u_k)$ and $V = (v_1, \ldots, v_k)$ with $k > 1$ and $\forall 1 \leq i \leq k \ u_i, v_i \in \{0,1\}^+$. A solution of the PCP is a sequence $i_1, \ldots, i_n \in \{1, \ldots, k\}$ with $n \geq 1$ such that:

$$u_{i_1}.\ldots.u_{i_n} = v_{i_1}.\ldots.v_{i_n}$$

A variant of the PCP (noted VPCP) is to find a sequence $i_1, \ldots, i_n \in \{1, \ldots, k\}$ with $n \geq 1$ such that:

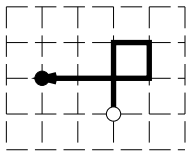
$$u_{i_1}.\ldots.u_{i_n} = v_{i_1}.\ldots.v_{i_n}$$

and

$$\forall 1 \leq j \leq n \quad |u_{i_1}.\ldots.u_{i_j}| \geq |v_{i_1}.\ldots.v_{i_j}|$$

This condition imposes the u_i always being "ahead" of the v_i. The PCP and the VPCP are well known to be undecidable problems [13, 11].

A *picture word* is a word over the alphabet $\Pi = \{u, r, d, l\}$ where each letter codes a unit move up for u, to the right, down and to the left for r, d and l respectively. For each unit move we plot the covered segment [12]. For instance, the word $\omega = u^2rdl^3$ describes the picture:



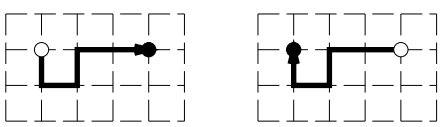
The white circle designs the start point of the picture and the black one the end point. The vector between this two points is called the *shift* of the word and is denoted by $sh(\omega)$:

$$sh(\omega) = (|\omega|_r - |\omega|_l, |\omega|_u - |\omega|_d)$$

We say that a word represents a *loop* if its shift is nil (start point=end point).

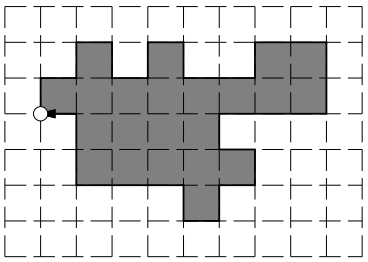
The *inverse* of a picture word ω is denoted $\overline{\omega}$ and is its image by the anti-morphism Inv from Π^* into Π^* such that $Inv(u) = d$, $Inv(r) = l$, $Inv(d) = u$ and $Inv(l) = r$. For instance:

$\omega = drur^2$ $\qquad$ $\overline{\omega} = l^2dlu$



A picture word is *self-avoiding* (we say also self-avoiding walk) if all its non-empty factors are not loop. A picture word is a *contour word of polyomino* [4] if it is a loop and there is no proper factor which is a loop. Such words describe polyominoes by the contour. Example of contour word of polyomino:

$$ururdrurdr^2ur^2d^2l^3drdldlul^3u^2l$$



We say that a polyomino is *row (respectively column) convex* if each row (resp. column) is convex. A polyomino is *convex* if it is row and column convex.

3 Method and example for rational languages

The coding of the PCP in linear picture languages can be made by using the possibility of generating palindromes [5]. To consider rational languages instead of linear languages deprives us of this possibility which seems, a priori, to be necessary to code the PCP in languages. We give here a method, for rational languages, which is based on the variant of the PCP noted VPCP.

We illustrate this method on an example and we give it formally at the end of the section. The example we deal with concerns self-avoiding walk.

Proposition 3.1 *Let L be a rational language over Π. It is undecidable to know whether or not L contains a self-avoiding word.*

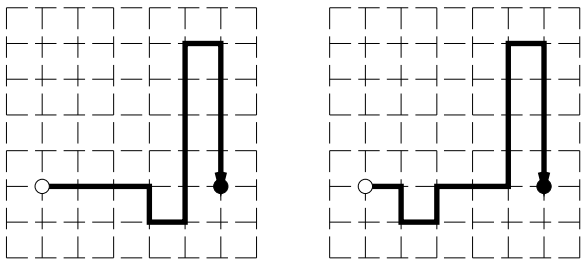
Proof: In order to obtain this result, we show that if this problem is decidable, the VPCP is decidable too. Thus, for every instance of the VPCP, we construct a regular language of Π^* which contains a self-avoiding walk if and only if the instance of the VPCP have a solution.

Let (U, V) be an instance of the VPCP with $U = (u_1, ..., u_k)$ and $V = (v_1, ..., v_k)$. We note K the rational language that we build. A word ω of K is an attempt of correspondence for a sequence $i_1, ..., i_n \in \{1, ..., k\}$ with $n \geq 1$. By giving the construction of K, we show that the only way to have a self-avoiding word is that $i_1, ..., i_n$ is a solution of the VPCP. By construction, it will be easy to see that for a solution we can find a self-avoiding word of K. The words of K have the following scheme:

$$\omega = \mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1}).\mu_1 \ldots \mu_{n-1}.\varphi(u_{i_n}).\lambda_n.\psi(v_{i_n}).\rho$$

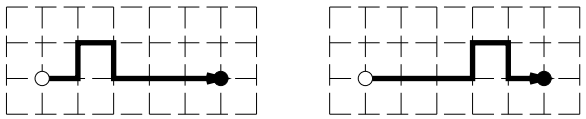
with φ and ψ two morphisms from $\{0,1\}^*$ into Π^* defined by:

$\varphi(1) = r^3dru^5rd^4$ $\qquad$ $\varphi(0) = rdrur^2u^4rd^4$



and

$\psi(1) = rurdr^3$ $\psi(0) = r^3urdr$



and the μ_i, λ_i and ρ some words of Π^* we will define below.

The patterns $\varphi(0)$, $\varphi(1)$, $\psi(0)$ and $\psi(1)$ have been designed such that if we place a pattern $\psi(0)$ (respectively $\psi(1)$) "under" (shifted two units down) a pattern $\varphi(0)$ (resp. $\varphi(1)$) we have no intersection and we have an intersection in the other cases. Notice that the significant part, which "codes" the information 0 or 1, of the patterns $\varphi(0)$ and $\varphi(1)$ is under the horizontal axis since the upper part is used for control as we will see below. For the patterns $\psi(0)$ and $\psi(1)$ the significant part is above the horizontal axis.

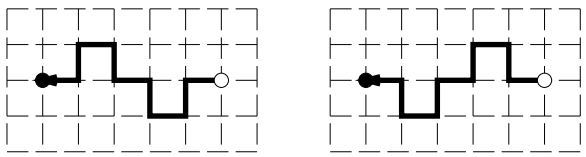
Thus, if we place $\psi(v_{i_1})$ under (shifted two units down) $\varphi(u_{i_1})$, we get a pattern without intersection only if v_{i_1} is a prefix of u_{i_1} or vice-versa. The role of λ_1 in ω is to "place the pen" below $\varphi(u_{i_1})$. The word λ_1 have the scheme:

$$\lambda_1 \in \delta.(\zeta_0 + \zeta_1)^*.\eta$$

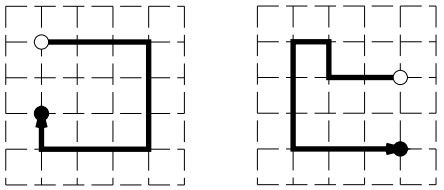
where δ, ζ_0, ζ_1 and η are words over Π.

Since it is impossible to place the pen at the right place without perturbing the comparison, we copy the pattern of $\varphi(u_{i_1})$ by using ζ_0 and ζ_1.

$\zeta_1 = ldlululdl$ $\zeta_0 = luldldlul$

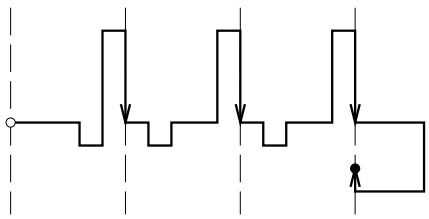


$\delta = r^3d^3l^3u$ $\eta = l^2uld^3r^3$

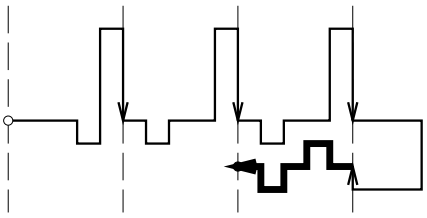


If we consider the morphism h from $\{0,1\}^*$ into Π^*, which associate with 0 the word ζ_0 and with 1 the word ζ_1, we want to show that there is no intersection in the word $\mu_0.\varphi(u_{i_1}).\lambda_1$ only if $\lambda_1 = \delta.h(\widetilde{u_{i_1}}).\eta$.

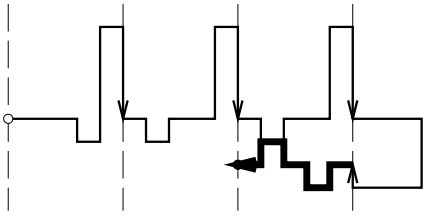
If for instance, we take $u_{i_1} = 100$, the picture described by $\varphi(u_{i_1}).\delta$ is the next one:



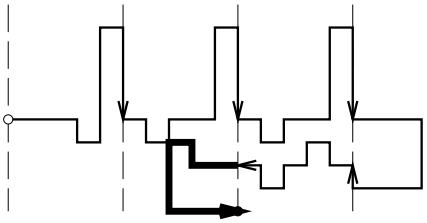
The patterns ζ_0 and ζ_1 have been designed to create cycles if they are placed under a 1 and a 0 respectively, but no intersection in the other cases. Thus, $\varphi(u_{i_1}).\delta.\zeta_0$ have no cycle:



while $\varphi(u_{i_1}).\delta.\zeta_1$ contains one:

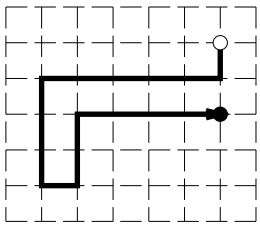


This assures us that the "copy", i.e. the part $(\zeta_0 + \zeta_1)^*$ of λ_1, corresponds to the pattern created by the φ. But we have to check that the copy does not stop before ending and that it is not too long. We are sure that the copy is not too short otherwise the pattern η creates a cycle with non-copied patterns:

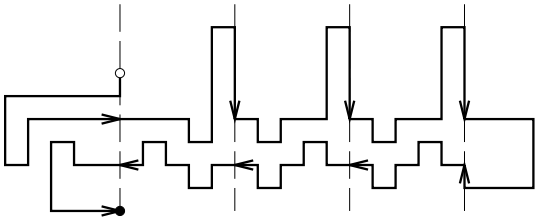


To check that the copy is not too long, we place before $\varphi(u_{i_1})$, by stating $\mu_0 = \beta$ a pattern which create a cycle. This pattern β is:

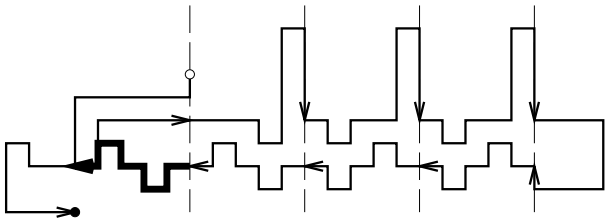
$$\beta = dl^5d^3ru^2r^4$$



Thus, if in λ_1 the copy is correct and have the good length, there is no intersection in $\mu_0.\varphi(u_{i_1}).\lambda_1$:

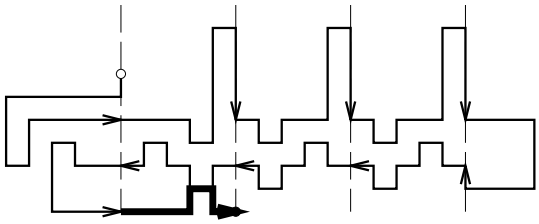


On the other hand, if the copy is too long a cycle is created with β (example where we add a ζ_1):

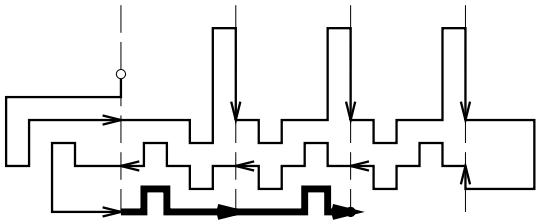


This assures us that $\lambda_1 = \delta.h(\widetilde{u_{i_1}}).\eta$ and thus under a pattern $\varphi(0)$ we have a ζ_0 and under a pattern $\varphi(1)$ we have a ζ_1. We have now to place $\psi(v_{i_1})$ and to show that there is no cycle only if v_{i_1} is a prefix of u_{i_1}. The interaction between the patterns ζ_0, ζ_1 and the patterns $\psi(0)$ and $\psi(1)$ is the same as between φ and ψ, i.e. there is no cycle if we place a $\psi(0)$ (respectively $\psi(1)$) under a ζ_0 (resp. ζ_1) and there is intersection otherwise. Since the ζ_0 correspond to $\varphi(0)$ of u_{i_1} and that similarly the ζ_1 correspond to $\varphi(1)$ of u_{i_1}, if there is no intersection, there is correspondence between the letters of v_{i_1} and the letters of u_{i_1}.

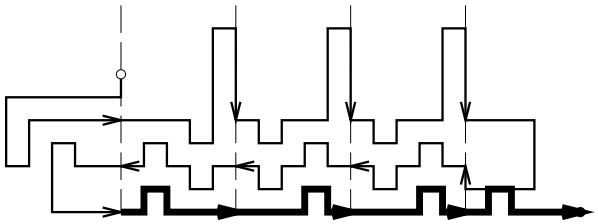
For instance, always with $u_{i_1} = 100$, even if λ_1 copies correctly u_{i_1}, we have necessary an intersection in $\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1})$ if $v_{i_1} = 0$:



On the other hand, if $v_{i_1} = 10$, there is no cycle:



The VPCP imposes "the u_i to be ahead of the v_i", i.e. for a solution $i_1, \ldots, i_n$, we have $\forall 1 \leq j \leq n \ |u_{i_1} \ldots u_{i_j}| \geq |v_{i_1} \ldots v_{i_j}|$. It is the role of δ (appearing in λ_1) to create a cycle if the v_i are too long. For instance, if $v_{i_1} = 1001$, we have a cycle:



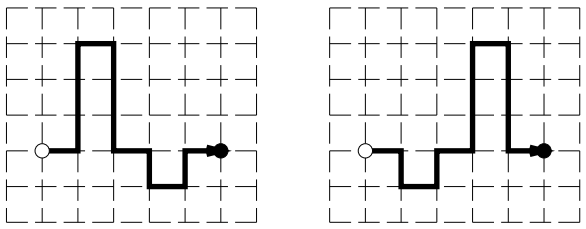
We note ν_i, for $1 \leq i < n$, the words of Π^* such that $\forall 1 \leq j < n \ u_{i_1} \ldots u_{i_j} = v_{i_1} \ldots v_{i_j} \nu_j$. We state $\nu_0 = \varepsilon$.

In the word ω which have the scheme imposed above, if we have no intersection in the factor $\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1})$ (with $\mu_0 = \beta$ et $\lambda_1 \in \delta.(\zeta_0 + \zeta_1)^*.\eta$), this means that v_{i_1} is a prefix of u_{i_1}. The word ν_1 is the part of u_{i_1} which have not been compared with the word made by the v_i. Before to add $\varphi(u_{i_2})$, we have to copy ν_1. This operation is similar with the copy in λ_1 and is done by μ_1 which have the scheme:

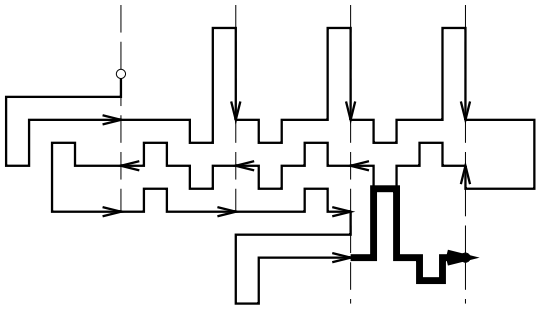
$$\mu_1 \in \beta.(\gamma_0 + \gamma_1)^*$$

Where β is the word used to define μ_0 and:

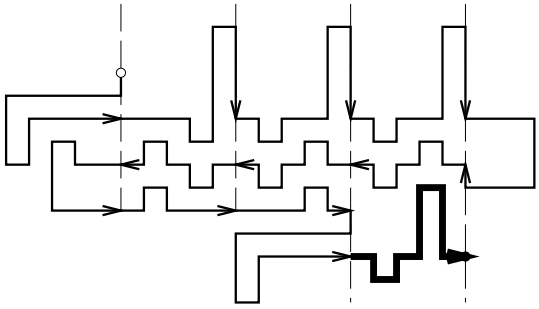
$\gamma_1 = ru^3rd^3rdrur$ $\qquad$ $\gamma_0 = rdruru^3rd^3r$



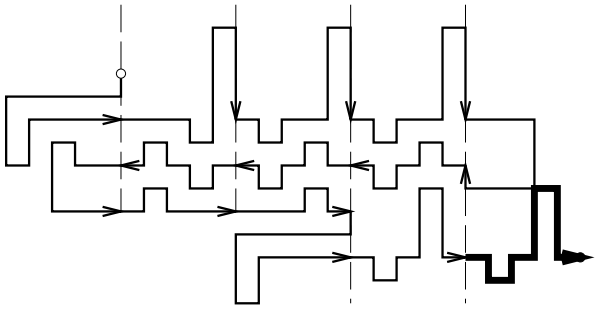
The factor $(\gamma_0 + \gamma_1)^*$ of μ_1 corresponds to the copy of ν_1 in order to be compared with the next v_i. If we consider the morphism g from $\{0,1\}^*$ into Π^* which associate 0 with the word γ_0 and 1 with the word γ_1, we want to show that if there is no cycle, then, $\mu_1 = \beta.g(\nu_1)$. In a same way as with ζ_0 and ζ_1, if we want no cycle, we have to put a γ_0 under a ζ_0 and a γ_1 under a ζ_1. Thus, with the example having $u_{i_1} = 100$ and $v_{i_1} = 10$, $\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1}).\beta.\gamma_1$ contains a cycle:



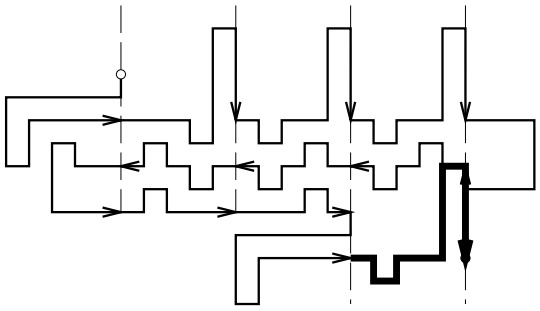
while $\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1}).\beta.\gamma_0$ have no cycle:



This assures us that ν_1 is correctly copied. If the copy is too long the pattern γ_0 or γ_1 collides with δ:



To verify that the copy is not too short, we use the fact that a μ_i is followed by a $\varphi(u_i)$. If we place a $\varphi(0)$ or a $\varphi(1)$ since ν_1 is not entirely copied, there is intersection:

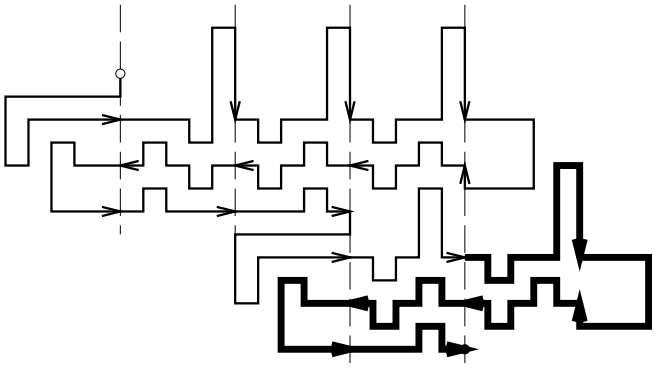


Then, the word μ_1 is the exact copy of ν_1 and we have $\mu_1 = \beta.g(\nu_1)$.

Next, in ω, we place $\varphi(u_{i_2})$ and a word λ_2, which like λ_1 belongs to $\delta.(\zeta_0 + \zeta_1)^*.\eta$. In the same way than with λ_1, the only way to have no cycle is that $\lambda_2 = \delta.h(\widetilde{\nu_1.u_{i_2}}).\eta$. Then, we place $\psi(v_{i_2})$ and we show like above that there is no cycle only if v_{i_2} is a prefix of $\nu_1.u_{i_2}$. If in the previous example we take $u_{i_2} = 0$ and $v_{i_2} = 0$, the word (without cycle):

$$\mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1}).\mu_1.\varphi(u_{i_2}).\lambda_2.\psi(v_{i_2})$$

represents the picture:



And so on. In fact, we can show by induction over n that if we have:

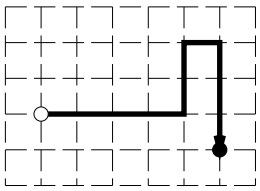
$$\begin{array}{rrcl} & \mu_0 & = & \beta \\ \forall 1 \leq j < n & \mu_j & \in & \beta.(\gamma_0 + \gamma_1)^* \\ \forall 1 \leq j < n & \lambda_j & \in & \delta.(\zeta_0 + \zeta_1)^*.\eta \end{array}$$

then:

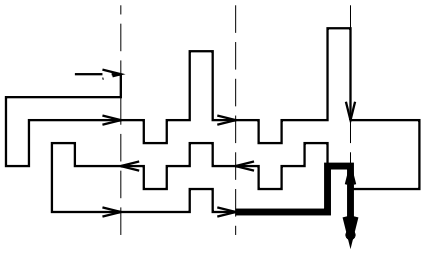
$$\begin{array}{c} \mu_0.\varphi(u_{i_1}).\lambda_1.\psi(v_{i_1}).\mu_1 \ldots \mu_{n-1}.\varphi(u_{i_n}).\lambda_n.\psi(v_{i_n}) \text{ is self-avoiding} \\ \Updownarrow \\ \begin{array}{rrcl} \forall 1 \leq j < n & \mu_j & = & \beta.g(\nu_j) \\ \forall 1 \leq j < n & \lambda_j & = & \delta.h(\widetilde{\nu_{j-1}.u_{i_j}}).\eta \\ \forall 1 \leq j \leq n & v_{i_j} & \leq & \nu_{j-1}.u_{i_j} \end{array} \end{array}$$

After this, we have to show that if ω is self-avoiding then v_{i_n} is exactly $\nu_{n-1}.u_{i_n}$. For this, we just take $\rho = \theta$ with:

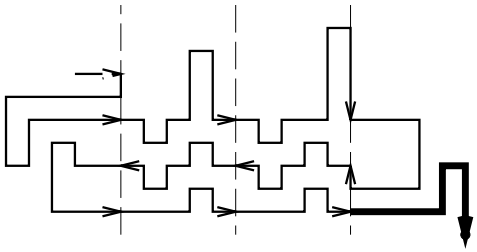
$$\theta = r^4 u^2 r d^3$$



If the last comparison ends too soon, there is a cycle with θ:



On the other hand, if the comparison have the good length, no cycle is created:

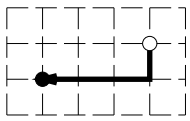


With the construction of ω, this word is self-avoiding only if the sequence $i_1, \ldots, i_n$ is a solution of the VPCP. In order to contain all the possible sequences, we consider the language K:

$$K = \alpha \left\{ \biguplus_{i=1}^{k} \underbrace{\beta(\gamma_0 + \gamma_1)^*}_{\text{“}\mu\text{”}} \varphi(u_i) \underbrace{\delta(\zeta_0 + \zeta_1)^* \eta}_{\text{“}\lambda\text{”}} \psi(v_i) \right\}^+ \theta$$

where α is the word:

$$\alpha = dl^3$$



which forces the first “μ” to be β (i.e. μ_0 seen above).

Thus, if K contains a self-avoiding word then the VPCP have a solution. Now, if the VPCP have a solution, we can easily associate with it a self-avoiding word of K. □

Our method consists to associate with every wording of the VPCP a rational picture word language which contains a word with the wished property — being self-avoiding in the previous proposition — if and only if the VPCP have a solution.

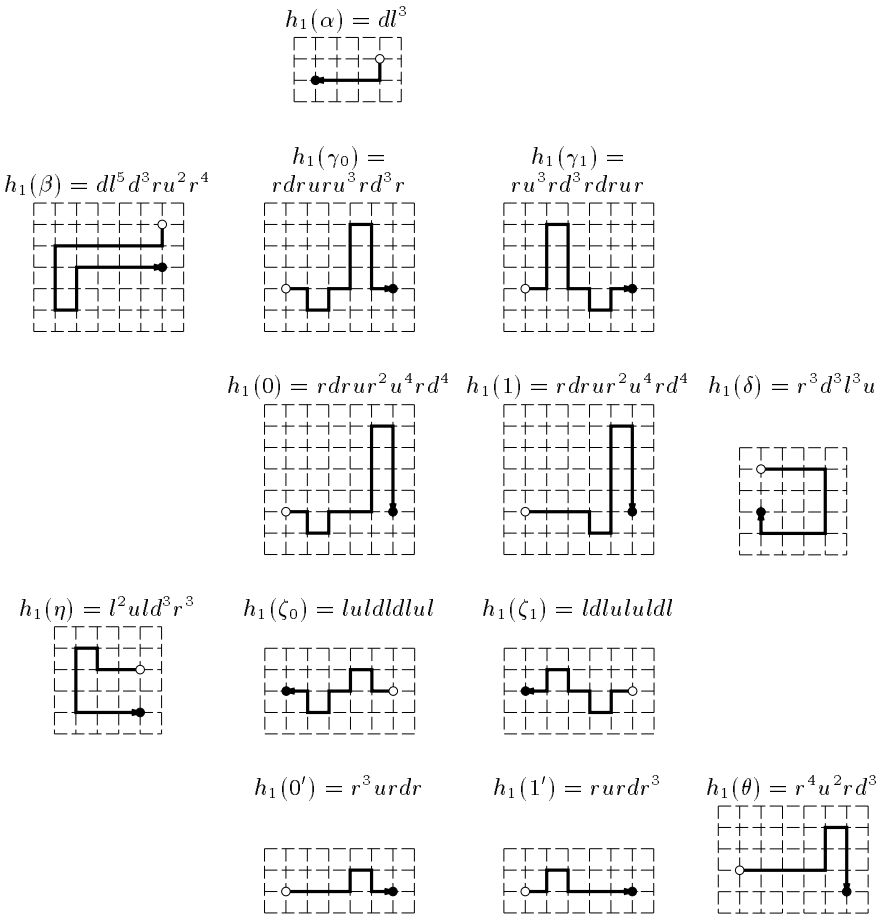


Figure 1: Morphism h_1 *(existence of a self-avoiding word)*

Definition 3.2 *Let (U, V) be an instance of the VPCP with $U = (u_1, \ldots, u_k)$ and $V = (v_1, \ldots, v_k)$. We associate with (U, V) the language $K_{(U,V)}$ over the alphabet Σ :*

$$\Sigma = \{0, 1, 0', 1', \alpha, \beta, \gamma_0, \gamma_1, \delta, \zeta_0, \zeta_1, \eta, \theta\}$$

$$K_{(U,V)} = \alpha \left\{ \biguplus_{i=1}^{k} \beta(\gamma_0 + \gamma_1)^* u_i \delta(\zeta_0 + \zeta_1)^* \eta . f(v_i) \right\}^+ \theta$$

where f is the morphism from $\{0, 1\}^*$ into Σ^* such that $f(0) = 0'$ and $f(1) = 1'$.

We just have to apply to $K_{(U,V)}$ the suitable morphism from Σ^* into Π^* to obtain the result. For instance for the Proposition 3.1, we have applied the morphism h_1 illustrated in figure 1.

4 Other results

This method allows us to re-show an undecidability result of Beauquier on the existence of a contour word of polyomino in rational languages [1] and a result of Dassow and Hinz on the existence of a tree (graph without cycle) in a rational language [6].

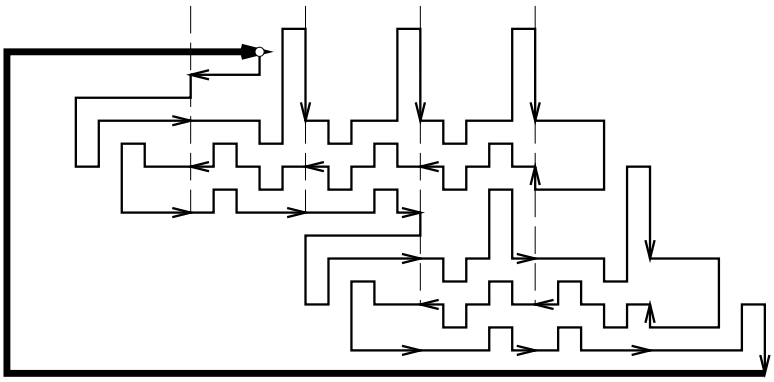


Figure 2: Contour word of polyomino in $K.l^+.u^+.r^+$

Theorem 4.1 (Beauquier 91 [1]) *Let L be a rational language over Π. It is undecidable to know whether or not L contains a contour word of polyomino.*

Proof: We consider an instance (U, V) of the VPCP and we use the language $h_1(K_{(U,V)})$ of the proof of the Proposition 3.1. It is easy to see that the language $K' = K.l^+u^+r^+$ contains a contour word of polyomino if and only if K contains a self-avoiding word (see figure 2). □

Theorem 4.2 (Dassow, Hinz 93 [6]) *Let L be a rational language over Π. It is undecidable to know whether or not L contains a tree.*

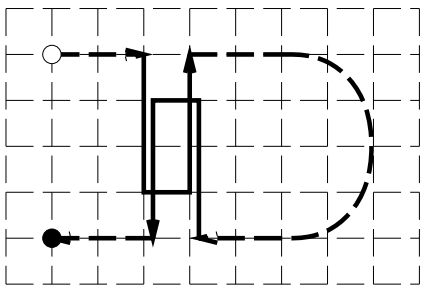
Proof: Let (U, V) be an instance of the VPCP. We still use the language $L = h_1(K_{U,V})$ of the Proposition 3.1 and we easily see that a self-avoiding word is a tree and that a word of L which is not self-avoiding have a cycle. □

The method allows to show some results which are not based on geometrical properties. It is the case of the existence of an *optimal word* or of a *minimal word*. We say that a word is optimal if it draws only one time each segment of the picture it describes. A word is said minimal if there is no shorter word which describes the same picture.

Proposition 4.3 *Let L be a rational language over Π.*

1. *It is undecidable to know whether or not L contains an optimal word.*
2. *It is undecidable to know whether or not L contains a minimal word.*

Proof: Let (U, V) be an instance of the VPCP. We consider the language $L = h_2(K_{(U,V)})$ where h_2 is the morphism illustrated in figure 3. The construction of h_2 is a variant of h_1. With h_2, an intersection have the scheme (the second drawing is slightly shifted to distinguish the moves):



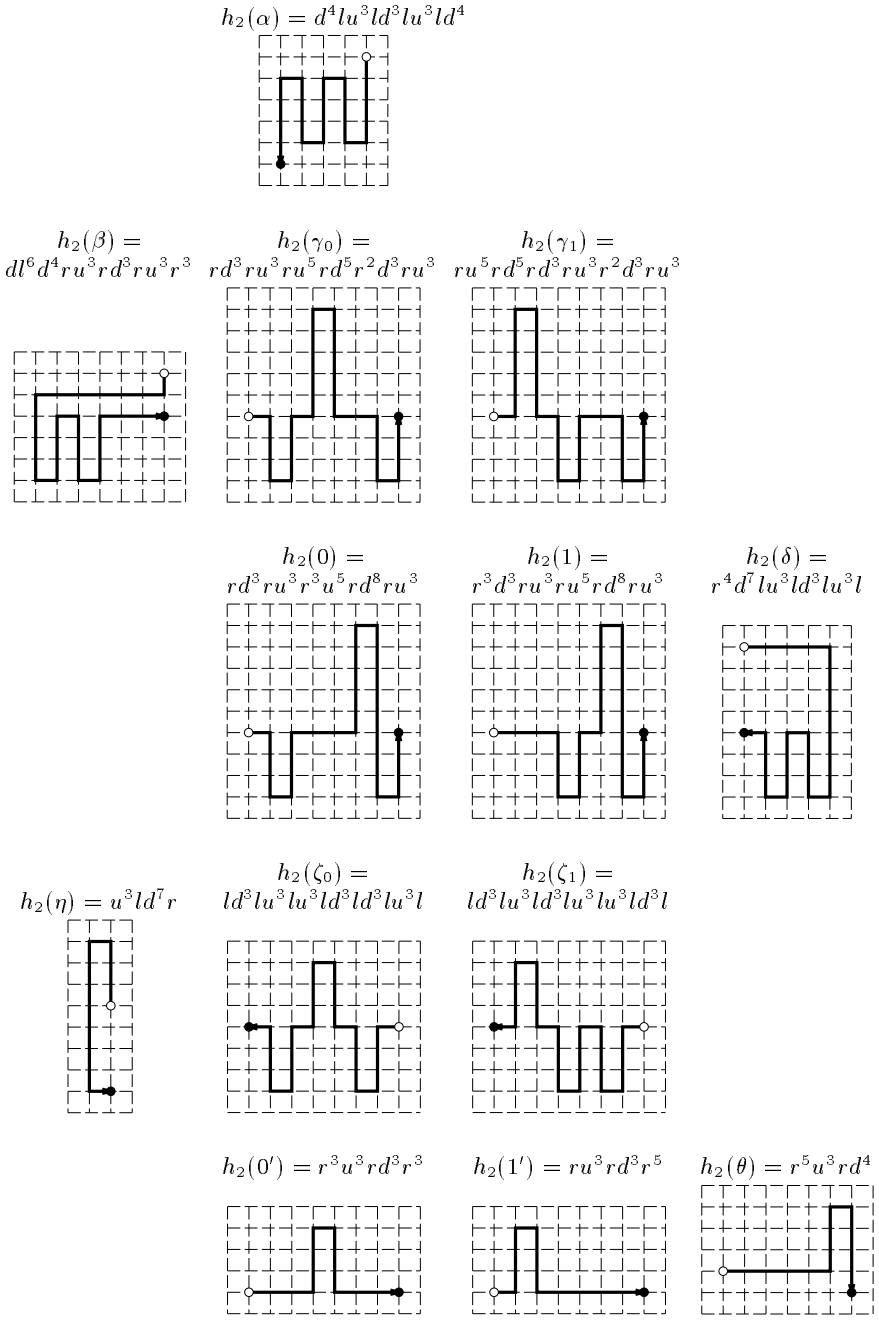
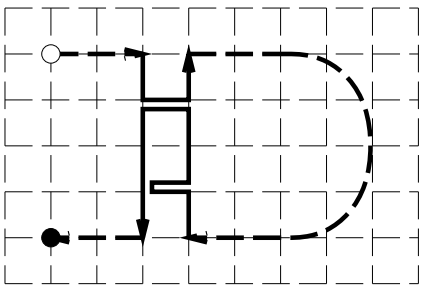


Figure 3: Morphism h_2 *(existence of a minimal word)*

For the point 1, we just have to notice that every self-avoiding word is optimal and that every word of L which is not self-avoiding have a segment plotted twice.

For the point 2, we note that to draw the non-dotted part a word of L uses 14 moves (d^3ru^3 and u^3ld^3) while we need only 9 letters (dru and $ulru^2ld^3$ for instance):



Thus, a word of L which contains an intersection is not minimal while a self-avoiding word is minimal. The instance (U, V) have a solution if and only if L contains a minimal word. □

5 Picture languages with blank moves

The use of the alphabet Π to code pictures imposes these ones to be connected. It is interesting to extend this alphabet with letters allowing to move without plotting segments. We consider the alphabet $\Pi_i = \{u, r, d, l, u', r', d', l'\}$ where u, r, d and l have the same meaning than before and where the primed letters induce an invisible move [8]. This allows to describe connected and non-connected pictures.

It is then natural to ask if we can decide if all the pictures of a language are connected or all non-connected. We answer by the negative to this two questions with to two following propositions.

Proposition 5.1 *Let L be a rational language over Π_i. It is undecidable to know whether or not L contains a word describing a connected picture.*

Proof: Let (U, V) be an instance of the VPCP. We consider the language $L = h_3(K_{(U,V)})$ where h_3 is the morphism from Σ^* into Π_i^* described in figure 4.

The idea is to code the information "0" or "1" in non-connected parts of $h_3(0)$ and $h_3(1)$. Only a correct copy will connect this parts (see $h_3(0')$ and $h_3(1')$). An example of correspondence is given in picture 5. □

Proposition 5.2 *Let L be a rational language over Π_i. It is undecidable to know whether or not L contains a word describing a non-connected picture.*

Proof: Let (U, V) be an instance of the VPCP. We consider the language $L = h_4(K_{(U,V)})$ where h_4 is the morphism from Σ^* into Π_i^* defined in figure 6.

Here, we construct simultaneously two disjoint connected component and mistakes in copies will connect them. An example of correspondence is given in picture 7. □

The method seems to be efficient enough to be extended to other semantics. It is the purpose of the last section with "pixel" semantic.

6 Pixel picture languages

By changing the semantic of the alphabet Π, we are interested in description of pictures made of pixels (unit square) instead of segments. For this, the letters of Π induce always unit move but

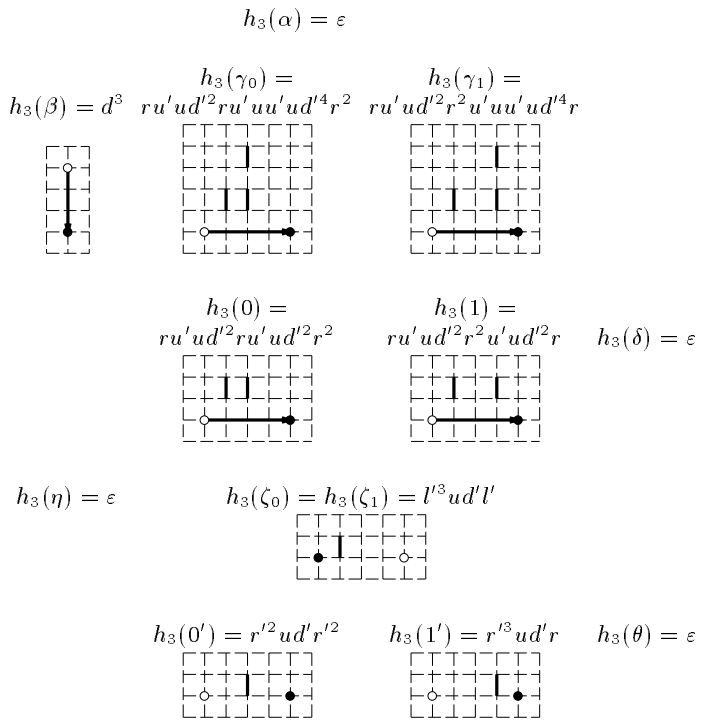


Figure 4: Morphism h_3 *(existence of a word describing a connected picture)*

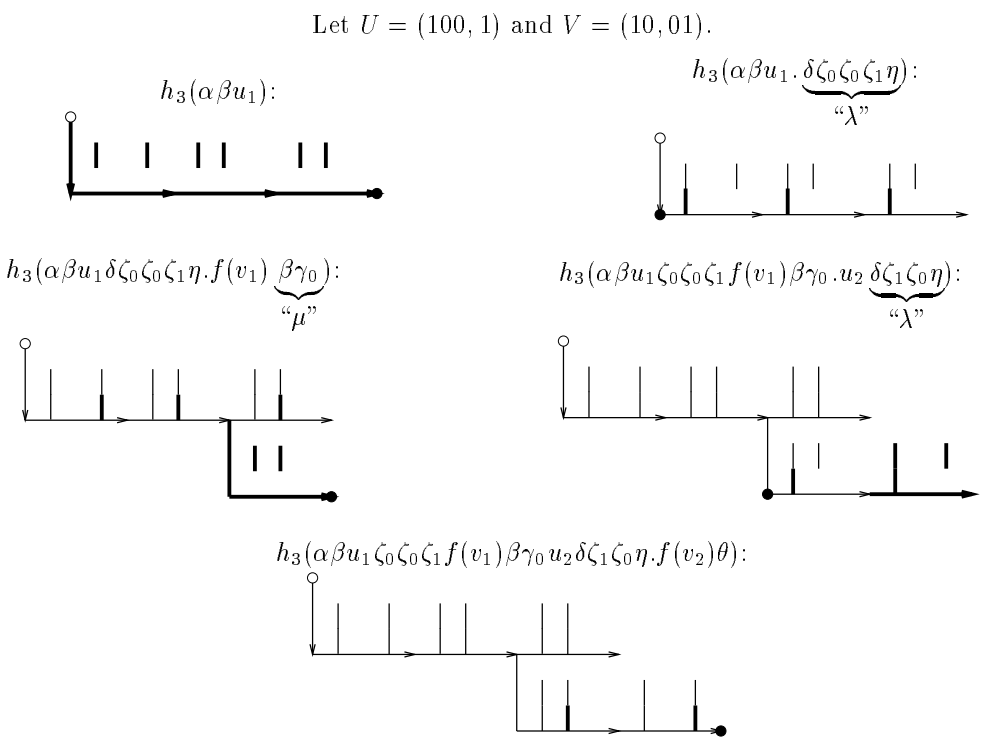


Figure 5: Example of correspondence for the existence of a connected picture

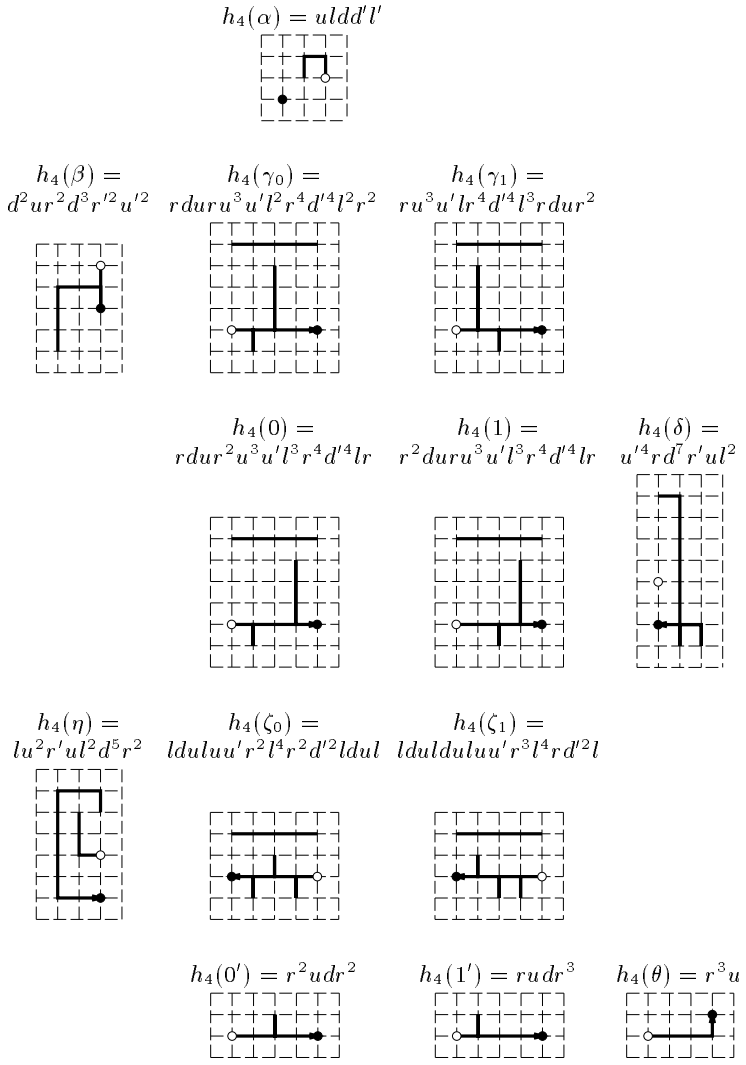


Figure 6: Morphism h_4 *(existence of a word describing a non-connected picture)*

Let $U = (100, 1)$ and $V = (10, 01)$.

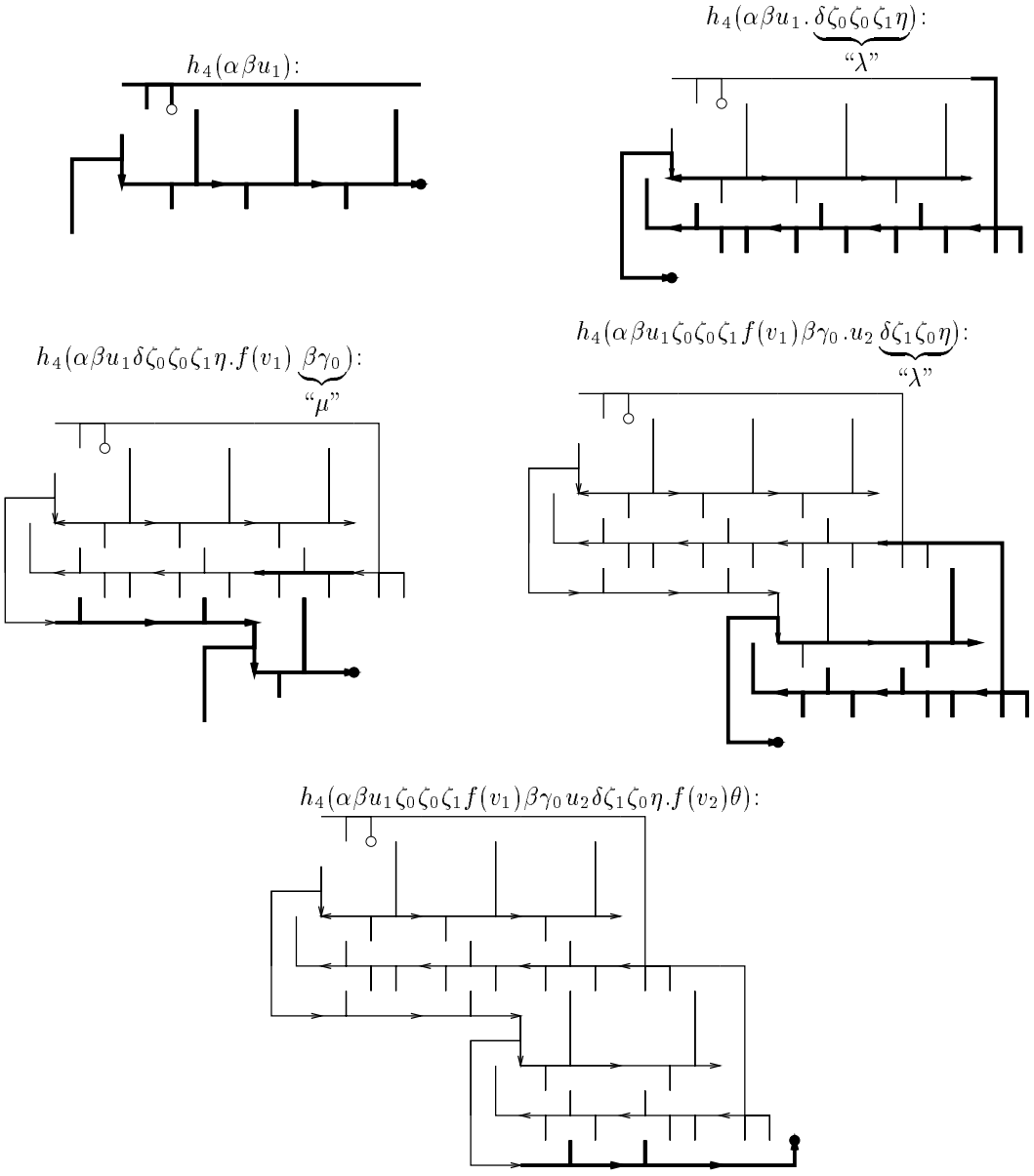


Figure 7: Example of correspondence for the existence of a non-connected picture

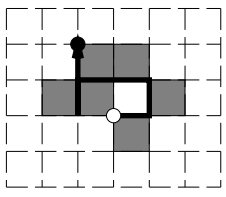


Figure 8: Picture described by *rullduu*.

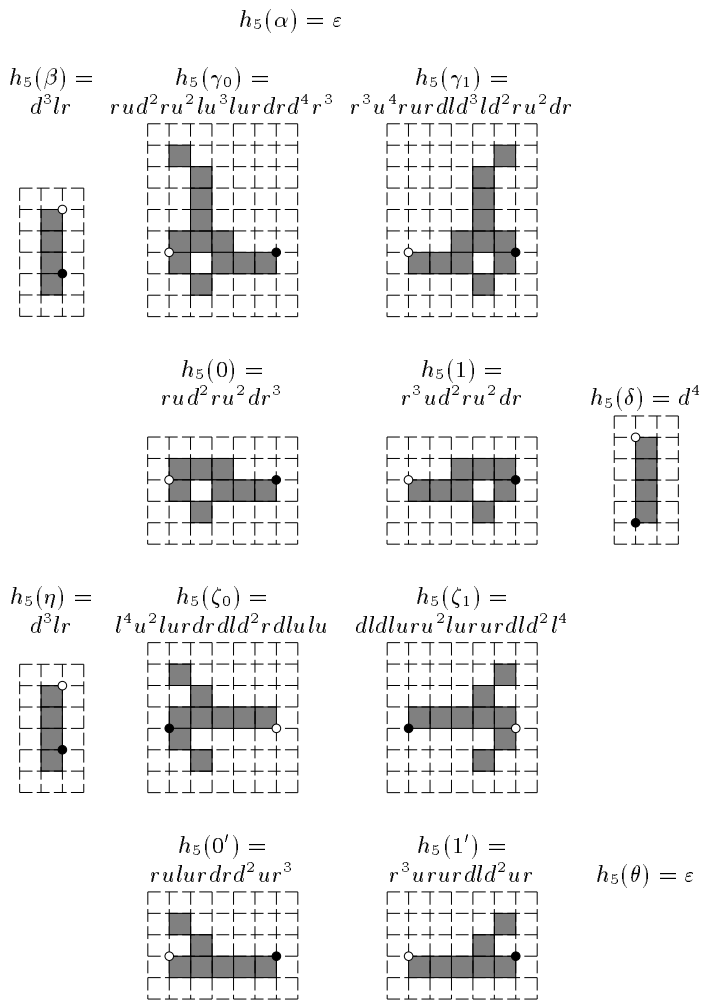


Figure 9: Morphism h_5 *(existence of a word describing a polyomino)*

we do not plot the segment under the move but the pixel in the right of the move [10]. Thus the word $rullduu$ describes the picture given in Figure 8.

This semantic can be used to describe polyominoes with or without holes since contour words of polyomino defined above can only describe polyominoes without hole (we said *polygons*).

Proposition 6.1 *Let L be a rational language over Π. It is undecidable to know whether or not L contains a word describing a polyomino.*

Proof: Let (U, V) be an instance of the VPCP. We consider the language $L = h_5(K_{(U,V)})$ where h_5 is the morphism from Σ^* into Π^* illustrated in figure 9.

The principle is similar than for Proposition 5.1. The information is coded with non-4-connected parts of $h_5(0)$ and $h_5(1)$ and only exact copies will connect them (see $h_5(\gamma_i)$, $h_5(0')$ and $h_5(1')$. An example of correspondence is given in figure 10. □

Compared with Theorem 4.1 this is not very surprising. More surprising is the next result which states the undecidability of the existence of a convex polyomino in a rational language, since we know that this problem is decidable for contour words of convex polyomino [2].

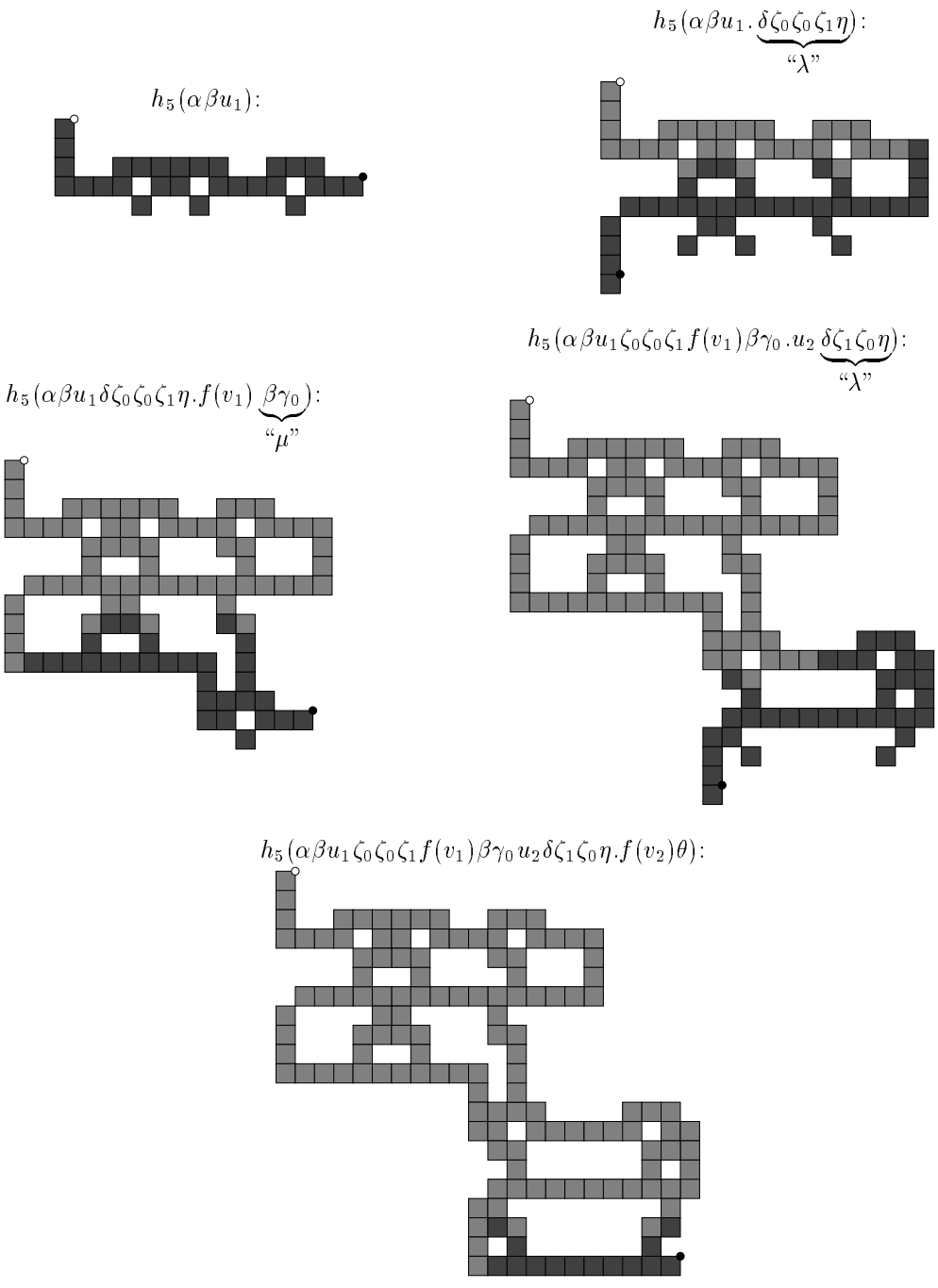


Figure 10: Example of correspondence for the existence of a polyomino

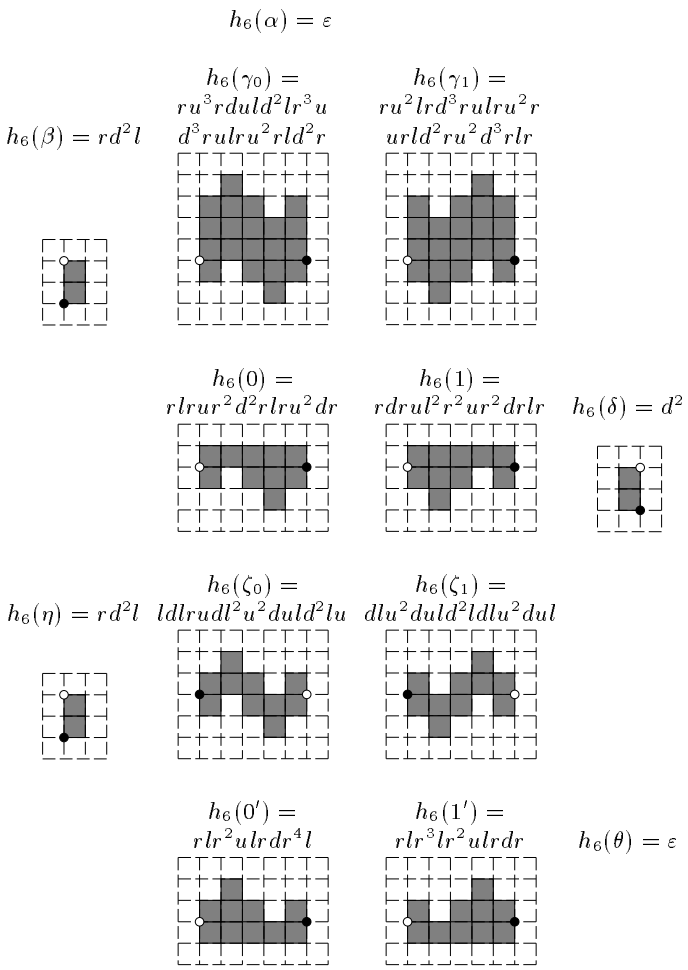


Figure 11: Morphism h_6 *(existence of a word describing a convex polyomino)*

Proposition 6.2 *Let L be a rational language over* Π. *It is undecidable to know whether or not L contains a word describing a convex polyomino.*

Proof: Let (U, V) be an instance of the VPCP. We consider the language $L = h_6(K_{(U,V)})$ where h_6 is the morphism defined in figure 11.

The idea is to construct a polyomino which contains no hole if and only if copies are correct (to place $h_6(\gamma_1)$ under a $h_6(0)$ creates a hole). And the polyomino is not row convex if a copy is too short or too long). An example of correspondence is given in figure 12. □

With similar morphisms, we can also show the undecidability of the existence of a row (column) convex polyomino, of a polygon or also, like in Proposition 4.3, of the existence of a minimal word in a rational pixel picture language.

7 Conclusion

The method we give allows to show the undecidability of numerous problems of existential properties in several semantics and this with the image of the same rational language $K_{(U,V)}$ by different morphisms.

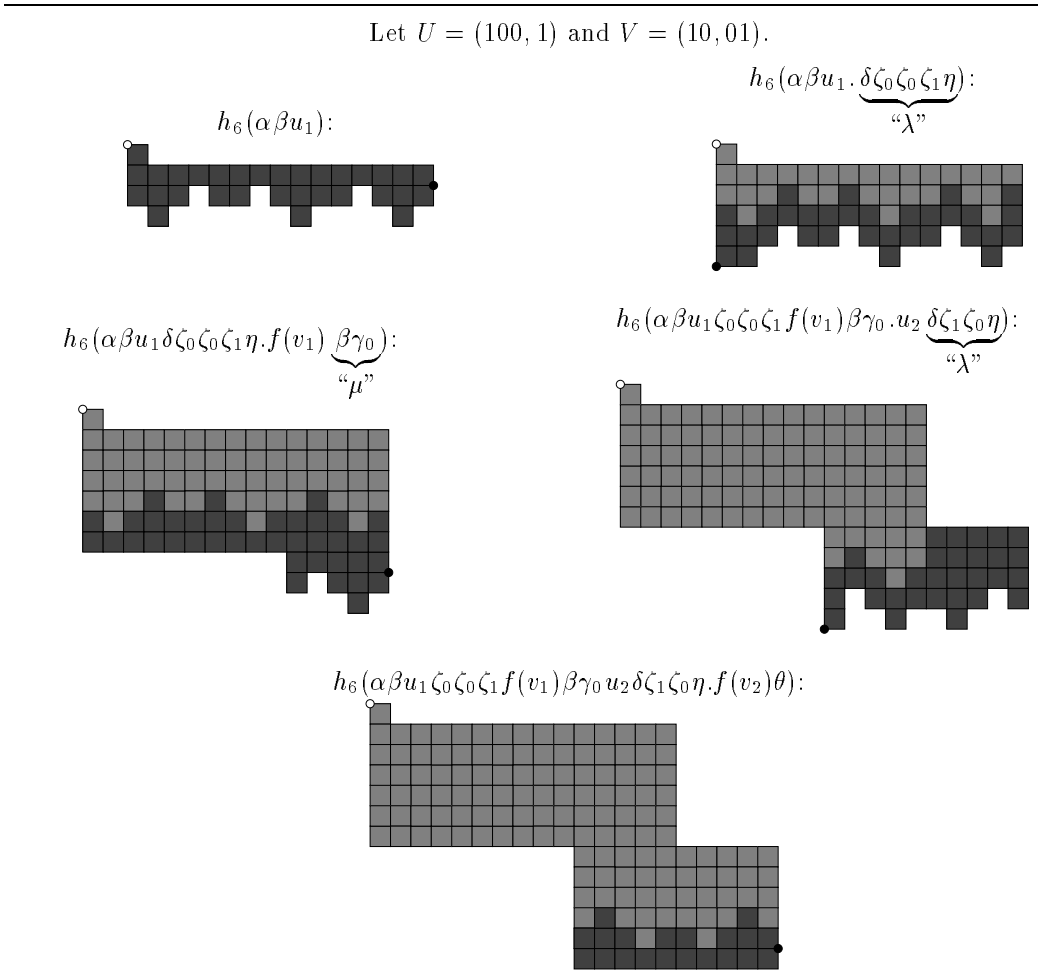


Figure 12: Example of correspondence for the existence of a convex polyomino

Notice that this method cannot be applied to "stripe languages" [15] — i.e. picture languages where every picture can be drawn between two given parallel lines — since the languages we produced cannot be bounded in any direction. It is also interesting to notice that, in the case of stripe languages, some of this problems become decidable, like for instance containing a contour word of polyomino.

It is also interesting to notice the differences between segment picture languages and pixel picture languages. We have seen that the existence of a (contour word of) polyomino, as well as the existence of a minimal word are undecidable in the two semantics, but the existence of a (contour word of) convex polyomino is decidable in segments but undecidable in pixels. Last, we have showed that the existence of an optimal word is undecidable in segment and we can show that this problem is decidable in pixels [14].

Acknowledgments

The authors want to thank M. Latteux for his helpful comments which allow to clarify the explained method.

References

[1] D. Beauquier. An undecidable problem about rational sets and contour words of polyominoes. *Information Processing Letters*, 37:257–263, 1991.

[2] D. Beauquier, M. Latteux, and K. Slowinski. A decidability result about convex polyominoes. In I. Simon, editor, *Proc. Latin American Theoretical Informatics (LATIN'92)*, volume 583 of *Lecture Notes in Computer Science*, pages 32–45, Sao Paulo, Brazil, 1992. Springer-Verlag, Berlin.

[3] J. Berstel. *Tranductions and Context-Free Languages.* Teubner Studienbücher, Stuttgard, 1979.

[4] M. Bousquet-Mélou. Polyominoes and polygons. *Contemporary Mathematics*, 178:55–70, 1994.

[5] J. Dassow. Graph-theoretical properties and chain code picture languages. *Journal of Information Processing and Cybernetics EIK*, 25:423–433, 1989.

[6] J. Dassow and F. Hinz. Decision problems and regular chain code picture languages. *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 45, 1993.

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

[8] F. Hinz and E. Welzl. Regular chain code picture languages with invisible lines. Technical Report 252, I.I.G., Techn. Univ. Graz, Austria, 1988.

[9] C. Kim and I. H. Sudborough. The membership and equivalence problems for picture languages. *Theoretical Computer Science*, 52(3):177–191, 1987.

[10] M. Latteux, D. Robilliard, and D. Simplot. Figures composées de pixels et monoïde inversif. *Bulletin of the Belgian Mathematical Society – Simon Stevin*, 4:89–111, 1997. Special issue *Journées Montoises d'Informatique Théorique* (Mons, Belgique, 1994). in french.

[11] M. Latteux and P. Turakainen. On characterizations of recursively enumerable languages. *Acta Informatica*, 28(2):179–186, 1990.

[12] H. A. Maurer, G. Rozenberg, and E. Welzl. Using string languages to describe picture languages. *Information and Control*, 54(3):155–185, 1982.

[13] E. L. Post. Recursive unsolvability of a problem of Thue. *Journal of Symbolic Logic*, 12:1–11, 1947.

[14] D. Simplot. A decidability result about optimal words in pixel picture languages. Technical report, L.I.F.L., Univ. Lille 1, France. in preparation.

[15] I. H. Sudborough and E. Welzl. Complexity and decidability for chain code picture languages. *Theoretical Computer Science*, 36(2-3):173–202, 1985.