Discrete Applied Mathematics 45 (1993) 29–49
North-Holland

Decision problems and regular chain code picture languages

Jürgen Dassow

Faculty of Computer Science, Magdeburg University of Technology, Magdeburg, Germany

Friedhelm Hinz

Department of Mathematics and Computer Science, University of Trier, Trier, Germany

Received 6 July 1990
Revised 17 July 1991

Abstract

Dassow, J. and F. Hinz, Decision problems and regular chain code picture languages, Discrete Applied Mathematics 45 (1993) 29–49.

By interpretation of the letters u, d, r, l, $\uparrow$, $\downarrow$ as "move up, (down, right, left)" with the pen, "lift" and "sink" the pen of a plotter we can associate a picture with a word. The set of pictures associated with the words in a regular (context-free) string language is called a regular (context-free) picture language or chain code picture language.

In this paper we discuss the decidability status of the following problems for regular and context-free picture languages:

- Is some picture a subpicture of all pictures of the language?
- Does the language contain a picture with a given property?
- Have all pictures of the language a given property?

1. Introduction

In the last three decades many different approaches have been introduced in order to solve problems in picture processing. One of the approaches uses chain codes which give a connection between pictures and strings describing pictures. In [8] Freeman gave a survey on this work which can be used in various fields as, e.g., picture generation and pattern recognition. Whereas Freeman used eight basic directions we restrict to four directions but our results will hold in the Freeman case, too.

Correspondence to: Professor J. Dassow, Fakultät für Informatik, Technische Universität Magdeburg, PSF 4120, 0-3010 Magdeburg, Germany.

We interpret the letters $u, d, r, l, \uparrow, \downarrow$ as movements of a plotter in the directions up, down, right, left and lift the pen and sink the pen. Thus with a word we associate a picture, and a set of pictures, called chain code picture language in [18], corresponds to a language of words.

One of the well-investigated decision problems in the theory of (chain code) picture languages is the subpicture problem to decide whether or not a given picture is a subpicture of some picture in a given language. This problem is shown to be decidable for context-free picture languages in [18] and to be NP-complete for regular picture languages in [13]. In this paper we study a modification of this problem which we call the universal subpicture problem: Decide whether or not a given picture is a subpicture of all pictures in a given language. We shall prove its undecidability for almost all pictures and regular languages. In [5,6] the following decision problems are studied for some geometrical or graph-theoretical properties P.

Q1: Decide whether or not the picture language contains a picture with P.

Q2: Decide whether or not all pictures in the picture language have property P.

The undecidability of Q1 for linear chain code picture languages is shown in [5,6].

In this paper we shall prove that in most cases undecidability will already hold for regular picture languages. In order to do this we use the same scheme: we simulate a linearly bounded automaton by a regular grammar in such a way that the pictures corresponding to a computation have some property if they do not have certain subpictures. This is a reduction of Q1 to some special universal subpicture problem.

With respect to Q2 we extend the list of decidable cases.

In [1,2] these questions are studied for a concept of simplicity of pictures which slightly differs from our concept.

The paper is organized as follows. In Section 2 we present the necessary notions on picture languages. In Section 3 we give a general scheme which can be used in order to prove the undecidability of the universal subpicture problem and problems of the existence of pictures with a given property in a regular picture language. In Section 4 we apply this scheme to problems of connectedness for generalized picture languages. In Section 5 we use the result to prove that there is a fixed linear language L such that "$R \subseteq L$?" is undecidable for regular languages R as a contribution to classical formal language theory. In Section 6 we discuss some cases where the above questions Q1 and Q2 are decidable. Section 7 contains some concluding remarks.

2. Definitions

Throughout the paper we assume that the reader is familiar with the basic concepts of formal language theory as regular and context-free grammars and languages and their properties (see, e.g., [14]). Only in one proof in Section 6, in addition, we need some knowledge on matrix grammars (see [7]). We give here only the definitions and notations for picture languages.

Let (m,n) be a point of the grid $\mathbb{Z}^2$ over the set $\mathbb{Z}$ of integers, and let $b \in \pi = \{u,d,r,l\}$. Then we set

$$u((m,n)) = (m,n+1), \qquad d((m,n)) = (m,n-1),$$

$$r((m,n)) = (m+1,n), \qquad l((m,n)) = (m-1,n).$$

For $z \in \mathbb{Z}^2$, we denote by $\langle z, b(z)\rangle$ the (undirected) unit line connecting z and $b(z)$. By a picture we mean a finite set of such unit lines in the grid. By this set-theoretic point of view a subpicture (superpicture) of the picture p is understood as a subset (superset) of p.

Now we associate with any word over $\pi_\updownarrow = \{u,d,r,l,\uparrow,\downarrow\}$ a triple (p,z,s), where the first component is a picture, the second component is a point, and the third component gives the state pen-up or pen-down in the following inductive way:

(i) $t(\lambda) = (\emptyset,(0,0),\downarrow)$,

(ii) if $t(w) = (p,z,s)$, then

$$t(wb) = \begin{cases} (p,z,\downarrow), & \text{if } b = \downarrow, \\ (p,z,\uparrow), & \text{if } b = \uparrow, \\ (p,b(z),\uparrow), & \text{if } s = \uparrow \text{ and } b \in \pi, \\ (p \cup \langle z,b(z)\rangle, b(z),\downarrow), & \text{if } s = \downarrow \text{ and } b \in \pi. \end{cases}$$

If $t(w) = (p,z,s)$, then p is the picture of w, which we denote by $pic(w)$, and z is its endpoint. By definition, the drawing of p starts in $(0,0)$.

In the sequel we are not interested in the exact position of the picture in the grid, i.e., we shall identify pictures which can be transformed to each other by shifts, and in notation we shall not distinguish between a picture and its equivalence class with respect to shifts. There are different possibilities to traverse a picture. Hence, for every nonempty picture, there is more than one description. Especially, given a traversal by some word a second traversal of the same picture can be given by inverting the direction and order of lines. Formally, we define the operator *inv* inductively by

$$inv(\lambda) = \lambda,\ inv(u) = d,\ inv(d) = u,\ inv(r) = l,\ inv(l) = r,$$

$$inv(wa) = inv(a)inv(w).$$

Obviously, $pic(w) = pic(inv(w))$.

Let G be a grammar such that the generated language $L(G)$ is contained in $\pi_\updownarrow^*$. Then we set

$$Pic(G) = \{pic(w):\ w \in L(G)\}.$$

A picture language B is called regular (context-free, etc.) if $B = Pic(G)$ for some regular (context-free, etc.) grammar G satisfying $L(G) \subseteq \pi^*$. If $L(G) \subseteq \pi_\updownarrow^*$, then we call the associated (regular, etc.) picture language generalized. All languages and picture languages considered in this paper will be represented by a grammar.

For a picture language B, we denote by $SUPER(B)$ the set of all pictures p such

that p is a superpicture of some $q \in B$. For a regular picture language B, $SUPER(B)$ is also regular. p is called a universal subpicture of B if p is a subpicture of all $q \in B$.

Sometimes we shall consider the pictures as (undirected) graphs, where the set of nodes is given by the set of points of $\mathbb{Z}^2$ belonging to the picture, and the edges correspond to the unit lines of the picture. For graph-theoretical concepts we refer to [3].

Note that all graphs generated by some grammar are bipartite, and therefore the maximal degree of the nodes in the graph gives the minimal number of colours which are necessary for an edge-colouring.

In this paper among other things we shall consider the following properties of pictures or graphs. p is called a simple curve if each node of p has at most degree 2, and it is called a simple closed curve iff all nodes of p have degree 2. p is called convex if there is a picture q such that $p \cup q$ is a closed simple curve and the intersection of the inner part of $p \cup q$ with any straight line which is parallel to one of the axes is a finite straight line.

3. A scheme for undecidability proofs for regular picture languages

In this section we present a method to obtain undecidability results for regular picture languages. This method is similar to the technique used in [17] to show the undecidability of the equivalence problem for regular languages. The basic idea is to simulate a run of a linearly bounded automaton by a regular grammar. This will be done in three steps. First we present a normal form for linearly bounded automata. Then we introduce usual Turing tapes and Turing tapes with defects such that the existence of a run without defects corresponds to the nonemptiness of the accepted language. Finally we present a regular grammar such that every word generated by the grammar corresponds to a run of the linearly bounded automaton on a probably defect Turing tape.

The picture drawn by a word has a certain property if no defects of the Turing tape occur in the corresponding run, and it fails to have this property if a serious defect occurs in the corresponding run of the linearly bounded automaton. As soon as this correspondence is established we have reduced the question, whether or not

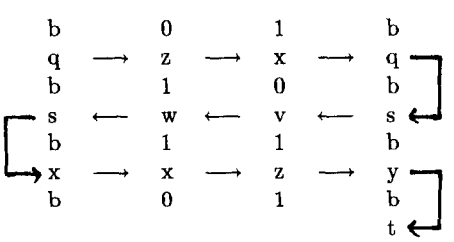


Fig. 1. t is a final state.

a regular grammar generates no description of a picture with a certain property, to the undecidable question whether or not all runs use defect tapes.

We now present the normal form of a linearly bounded automaton which can be assumed without loss of generality. In the beginning the machine scans the left endmarker. Then from left to right it starts to scan (and to rewrite) the complete input until it finds the endmarker. Then it performs one stationary step and starts to scan the take from right to left until it finds the left endmarker. Again, it performs a stationary step and scans up to the right endmarker. The machine finishes in a final state or repeats the process arbitrarily often. The control is transformed to a final state only when the right endmarker is the input, and the machine halts only in a final state.

A run of such a machine can be written into a rectangle each line of which corresponds to one scan, i.e., the horizontal width is determined by the length of the input word and the vertical length is determined by the number of scans in the run. The first line contains the input word. We note the letter which is actually read above the finite control and the letter that is actually written below the finite control. The arrows indicate the direction in which the input head moves. A representative example is given in Fig. 1.

For our purpose it is technically useful to have a Turing tape with no more than two symbols. By this we avoid the terminators and expect a tape that is arbitrarily filled with letters from $\{0, 1\}$. Of course, we cannot recognize an arbitrary context-sensitive language L on such a tape since we would be unable to identify the end of the input word. Instead of L we accept the language $011h(L)011$ where the encoding h is given by

$$h(0) = 110, \qquad h(1) = 101.$$

It is not difficult to transform a linearly bounded automaton for L into a linearly bounded automaton for $011h(L)011$ (where 011 is used as a "software endmarker"). This is shown in Fig. 2.

We find it convenient to use a transition function

$$\delta : Q \times \{0, 1\} \to 2^{\{R, L, RL, LR\} \times \{0, 1\} \times Q}$$

where R means move to the right, L means move to the left, and both RL and LR

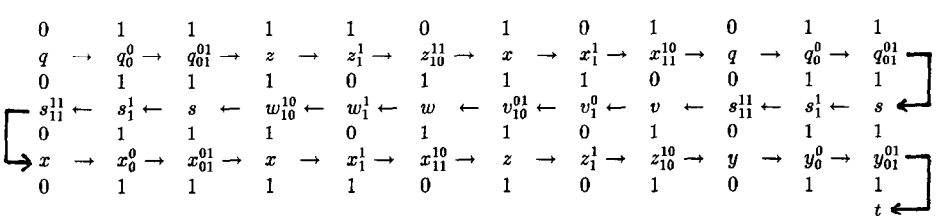
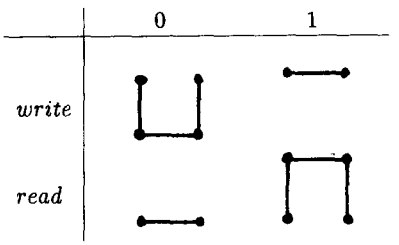


Fig. 2. In the upper index we store up to two letters that have just been read and in the lower index we store up to two letters that have just been written.



a

	0	1
$\overrightarrow{write}$	dru	r
$\overrightarrow{read}$	r	urd
$\overleftarrow{write}$	dlu	l
$\overleftarrow{read}$	l	uld

b

$right-to-left$	$rddl$
$left-to-right$	$lddr$

c

Fig. 3.

mean keep stationary. We use four symbols instead of the usual three symbols as we want to indicate whether a stationary step is after moving right and before moving left (RL) or after moving left and before moving right (LR).

A Turing tape with defect is a tape that may nondeterministically change the contents of cells in the absence of the read-write-head from 0 to 1 or from 1 to 0. We say that a serious defect occurs if both a 1 is changed to a 0 and a 0 is changed to a 1. If only 0s are changed to 1s this defect is considered to be nonserious, because the automaton can detect such a defect as it finds a block 111 where it expects to read a marker 011 or $h(0)=110$ or $h(1)=101$, and it refuses to enter a final state.

To be able to represent a run with defect in our rectangle scheme it is necessary to distinguish between reading a letter and drawing a letter. We use the functions of Fig. 3(a). The pictures of Fig. 3(a) are drawn from left to right if the machine

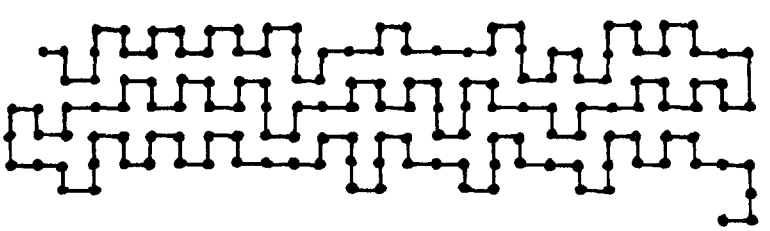


Fig. 4.

scans the tape from left to right, and they are drawn in the other direction otherwise. Thus we need four functions represented in Fig. 3(b). To change the direction and to start a new scan we need two constants as represented in Fig. 3(c), namely the constant *right-to-left* in order to change from the $(2i-1)$st scan to the $2i$th scan and the constant *left-to-right* in order to change from the $2i$th scan to the $(2i+1)$st scan. In this representation the run of Fig. 1 looks like Fig. 4.

A regular grammar to describe such a run of a linearly bounded automaton $M=(Q, \{0,1\}, q_0, \delta, F)$ in normal form is $G=(Q, \pi, q_0, P)$ where P contains the following rules

$$
\begin{aligned}
&q \to \lambda, && \text{for } q \in F,\\
&q \to \overrightarrow{read}(a)\,\overrightarrow{write}(b)\; q', && \text{if } (R, b, q') \in \delta(q, a),\\
&q \to \overrightarrow{read}(a)\,\overrightarrow{write}(b) \textit{right-to-left}\; q', && \text{if } (RL, b, q') \in \delta(q, a),\\
&q \to \overleftarrow{read}(a)\,\overleftarrow{write}(b)\; q', && \text{if } (R, b, q') \in \delta(q, a),\\
&q \to \overleftarrow{read}(a)\,\overleftarrow{write}(b) \textit{right-to-left}\; q', && \text{if } (RL, b, q') \in \delta(q, a).
\end{aligned}
$$

The above table does not only define a grammar G, but it also induces in an obvious way a bijective relation between a derivation in G and a run of M (on a possible defect tape). By the construction of G reading a cell of the tape in the ith scan is performed two unit lines below writing in the same cell in the $(i-1)$st scan (if that cell is visited in the $(i-1)$st scan). Thus if no defect occurs, the picture drawn by G contains only the pictures p_{00} and p_{11} of Fig. 5 as subpictures, but if a serious defect occurs it also contains the pictures p_{01} and p_{10} as subpictures.

As an accepting computation of the linearly bounded automaton corresponds to a run without defect and as the emptiness problem is undecidable for languages accepted by linearly bounded automata, we obtain the following theorem.

Theorem 3.1. *It is undecidable whether or not a regular picture language contains the above picture p_{01} as a universal subpicture.*

Before we consider universal subpictures different from p_{01} we apply Theorem 3.1 to obtain the undecidability of question Q1 for some graph-theoretical properties.

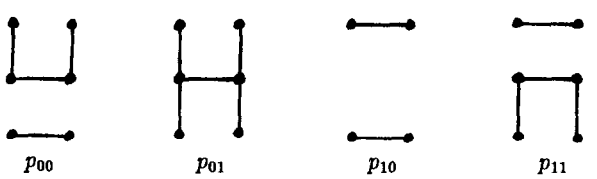


Fig. 5.

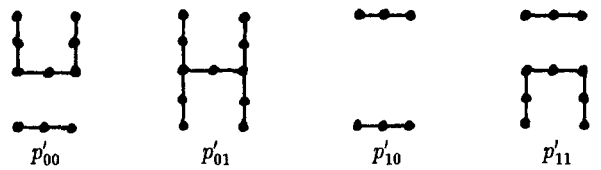
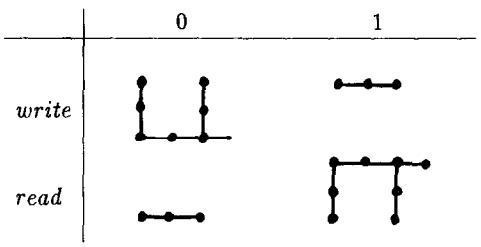


Fig. 6.

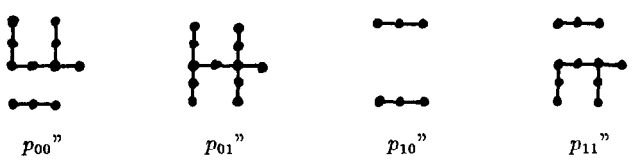
Theorem 3.2. *It is undecidable whether or not a regular picture language contains*

(i) *a simple curve,*
(ii) *a closed simple curve,*
(iii) *an Eulerian graph,*
(iv) *an Eulerian cycle,*
(v) *a tree,*
(vi) *a regular graph,*
(vii) *a Hamiltonian graph,*
(viii) *a Hamiltonian cycle,*
(ix) *a graph edge-colourable by three colours,*
(x) *a 2-connected graph.*



	0	1
$\overrightarrow{write}$	$d^2r^3lu^2$	r^2
$\overrightarrow{read}$	r^2	$u^2r^3ld^2$
right−to−left	$r^2d^4l^2$	
left−to−right	$l^2d^4r^2$	

a



b

Fig. 7.

Proof. Let p be a picture generated by the grammar G as above and corresponding to a run without defect. Then p is a simple curve (Eulerian, a tree) if and only if p does not contain the subpicture p_{01}. Now (i), (iii) and (v) follow from Theorem 3.1.

Obviously, if p is a simple curve, then it can be closed in the grid. This closed simple curve forms an Euclerian cycle which is contained in $SUPER(Pic(G))$. Furthermore, p_{01} is a universal subpicture of $Pic(G)$ if and only if it also is a universal subpicture of $SUPER(Pic(G))$. This implies (ii) and (iv).

As mentioned in [5] a graph consisting of at least two edges is regular if and only if it is a closed simple curve. Thus (vi) follows.

In order to prove (vii)-(x) we modify G. First we replace every letter $b \in \pi$ by the word bb. Thus we enlarge every picture by the factor two. Now the characteristic subpictures are those of Fig. 6. It is easy to see that a graph containing p_{01}' and having a start node of degree 1 cannot be Hamiltonian. On the other hand, a picture corresponding to a run without defect is still a simple curve. Therefore (vii) holds. (viii) follows as above since we can proceed from a curve to a cycle.

Note that, for every letter $a \in \pi$, we have

$$write(a) = inv(write(a)) \quad \text{and} \quad read(a) = inv(read(a)).$$

In all the following modifications we preserve this property, thus we only need to define *write* and *read* explicitly.

So far all pictures generated can be edge-coloured by at most three colours. Figure 7 presents a modification such that a picture can be edge-coloured by three colours if and only if it corresponds to a run without defect if and only if p_{01}'' is not a subpicture (see Fig. 7(b)). This implies (ix).

While up to now we used connected characteristic subpictures in case of a serious defect and disconnected characteristic subpictures in case of a run without defect we now do the opposite (see Fig. 8). Clearly, a run with a serious defect results in a picture that contains subpicture p_{01}''', and it is not 2-connected. However, a picture corresponding to a run without defect is not 2-connected since there are problems in the first row when we use letters from the input and there may arise problems in the last row when the machine prints letters that it will never read in a later scan.

The problem of the last row can easily be avoided by requiring that the linearly bounded automaton prints only 0 in the last scan of an arbitrary run which can be required without loss of generality. Furthermore, the suffix *right-to-left* has to be cancelled at every end of every word generated by the regular grammar.

$p_{00}''' = p_{10}''' = p_{11}'''$ $\qquad$ p_{01}'''

Fig. 8.

In order to solve the problem occurring in the first row we introduce a copy A' for every variable A used in the grammar so far. These primed versions will be used in the first row. Thus, if S is the start symbol, now S' is the start symbol. For every rule of the form

$$A \to \overrightarrow{read}(a)\,\overrightarrow{write}(b)B \quad \text{or} \quad A \to \overrightarrow{read}(a)\,\overrightarrow{write}(b)\textit{right-to-left}\ B$$

we add the rule

$$A' \to rr\overrightarrow{write}(b)B' \quad \text{or} \quad A' \to \overrightarrow{read}(a)\,\overrightarrow{write}(b)\textit{right-to-left}\ B,$$

respectively. By these modifications there is no one-to-one correspondence between a run of the machine and a derivation of the grammar; several runs correspond to one derivation, and several derivations may also describe the same picture since $p_{00}'''=p_{10}'''=p_{11}'''$. The crucial issue is that we are still sure that a subpicture p_{01}''' of p occurs if and only if there is a serious defect in every run that corresponds to some description of p. Every picture described by the grammar that does not contain p_{01}''' is 2-connected. Thus we obtain (x). □

Finally in this section let us return to the universal subpicture problem. Note that on the way we have proved that for p_{01}' and p_{01}'' it is undecidable whether or not the picture is a universal subpicture of a regular language. The same holds for all pictures shown in Fig. 9 since they are subpictures of p_{01}'' and do not appear in any picture drawn by a grammar used in the proof of Theorem 3.2(ix) and according to a run without defect.

It is not difficult to prove an undecidability result for any other picture that is large enough by a suitable modification of the grammar. For pictures of at most one unit line, i.e., $pic(\lambda)$, $pic(r)$, and $pic(u)$, the question is easily proved to be decidable. For the picture $pic(uu)$, the undecidability can be proved using the following constructors:

	0	1
$\overrightarrow{write}$	dur	r
$\overrightarrow{read}$	r	udr
right-to-left	$drdl$	
left-to-right	$ldldrr$	

	0	1
$\overrightarrow{write}$	rr	$durdurdu$
$\overrightarrow{read}$	$(u^3d^3r)^2u^3d^3$	$(u^2d^2r)^2u^2d^2$
right-to-left	d^3	
left-to-right	d^3	

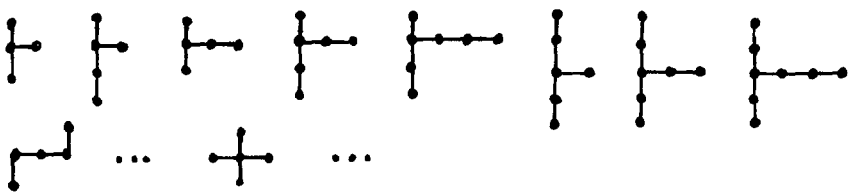


Fig. 9. Subpictures of p_{01}'' that are sufficient to detect a defect.

The characteristic subpictures for this construction are

$p_{00}^{(4)}$ $p_{01}^{(4)}$ $p_{10}^{(4)}$ $p_{11}^{(4)}$

and therefore *pic*(*uu*) is a subpicture of $p_{01}^{(4)}$. It can easily be seen that *pic*(*uu*) is not contained in a picture corresponding to a run without defect in this grammar.

As an open question we leave the universal subpicture problem concerning the picture *pic*(*ru*).

4. Undecidable problems for generalized picture languages

All undecidability results for regular picture languages trivially also hold for generalized picture languages. Yet there are some properties of pictures that are of interest only for generalized picture languages. It is shown in [6] that it is undecidable whether or not a linear generalized picture language

(1) contains no connected picture and

(2) contains no disconnected picture.

We apply the method presented in Section 3 to extend these results to the class of regular generalized picture languages.

Theorem 4.1. *It is undecidable whether or not a generalized regular picture language contains only*

(i) *disconnected pictures,*

(ii) *connected pictures.*

Proof. (i) We construct a regular grammar G_0 by

	0	1
$\overrightarrow{write}$	$du\uparrow r\downarrow$	$d^2u^2\uparrow r\downarrow$
$\overrightarrow{read}$	$ud\uparrow r\downarrow$	$\uparrow r\downarrow$

right-to-left	dd
left-to-right	$\uparrow dd\downarrow$

The characteristic subpictures are

$p_{00}^{(5)}$ $p_{01}^{(5)}$ $p_{10}^{(5)}$ $p_{11}^{(5)}$

Clearly, all columns of a picture described by G_0 form connected pictures (every column for itself), if the description of this picture corresponds to a run without defect. In order to connect the columns with each other we need the following modification: Provide a new variable A_0 and rules $A_0 \to rA_0$, $A_0 \to lA_0$, $A_0 \to \lambda$, and for every rule of the form $A \to \lambda$, we insert a rule $A \to A_0$. This construction allows to connect the columns in the last row. Thus, if every column is connected itself, the grammar generates a connected picture. But if a column is not connected itself, the modification will not allow to hide this defect, since there are no parallel horizontal lines besides that in the last row.

(ii) To prove this theorem we can basically use the same construction as in the proof of Theorem 3.1, especially the same write and read function. Yet if we finish the ith line, we do not connect it with the $(i+1)$st line but with the $(i+2)$nd line using *right-to-left* $= ru^4d^8u^2\uparrow l\downarrow$ for even i and *left-to-right* $= lu^4d^8u^2\uparrow r\downarrow$ for odd i. So far, for a picture corresponding to a run without defect there are two components, one of them consisting of the odd lines and the other one consisting of the even lines.

The discussion of a run with defect is a bit more complicated. Clearly, as soon as the first defect occurs an odd line and an even line will be connected. Therefore, if all odd lines are connected by subwords *left-to-right* and all even lines are connected by subwords *right-to-left* the whole picture forms one component. This works very well as long as the run keeps the form of a rectangle. But as a defect may cheat the machine concerning the correct position of an endmarker, a run with a defect need not obey the rectangle form. We now analyze the connection of two

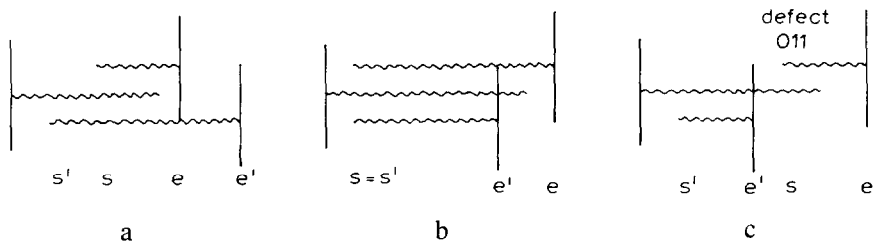


Fig. 10.

successive odd lines in the presence of such a defect. The connection of two successive even lines can be studied in the same way.

Let s be the horizontal component of the start point of the $(2i-1)$st line and e the horizontal component of its endpoint. Similarly, we define s' and e' for the $(2i+1)$st line. Note that $s' \leq e$ since the $2i$th line scans from right to left. We distinguish three cases concerning e'.

Case 1: $e' \geq e$ (Fig. 10(a)). For $s' < e < e'$ the subword *right-to-left* of line $2i-1$ draws a connection.

Case 2: $s \leq e' < e$ (Fig. 10(b)). The subword *right-to-left* of row $2i+1$ draws a connection.

Case 3: $e' < s$ (Fig. 10(c)). The subword *right-to-left* of line $2i+1$ draws a connection to line $2i$, but a direct connection to row $2i-1$ does not exist. Since $s < s'$ row $2i$ has overread the marker. Therefore there has to appear a serious defect which connects line $2i$ and line $2i-1$.

5. An application to classical formal language theory

In this section we state the undecidability of a problem in "classical" formal language theory but in the proof we shall use picture languages, and thus we demonstrate the usefulness of picture languages for "classical" string languages.

Theorem 5.1. *There is a fixed linear language L such that it is undecidable for regular grammars G whether or not $L(G) \subseteq L$.*

Proof. We consider the languages

$$L_1 = \{uw_1 rddlw_2 u\colon\ w_1 \in \{r, dru, urd\}^*,\ w_2 \in \{l, dlu, uld\}^*,\ \#_r(w_1) = \#_l(w_2)\}$$

and

$$L_2 = \{uw_2 rddlw_1 u\colon\ w_2 \in \{l, dlu, uld\}^*,\ w_1 \in \{r, dru, urd\}^*,\ \#_r(w_1) = \#_l(w_2)\}.$$

In both cases we use the basic ideas on which the construction in the proof of Theorem 3.1 is based. Moreover, by the *right-to-left* part in L_1 and the *left-to-right*

part in L_2 pictures corresponding to words in L_1 and L_2 have the same start- and endpoint. Thus the language

$$L = \pi^*(L_1 \cup L_2)\pi^*$$

contains no description of a tree. Furthermore, L is linear. Now let G be a regular grammar which simulates the work of a linearly bounded automaton as in the proof of Theorem 3.1. Then we obtain $L(G) = R \subseteq L$ if and only if $Pic(G)$ contains no trees. Thus the decidability of "$R \subseteq L$?" would imply the decidability of the existence of a tree in $Pic(G)$ in contradiction to Theorem 3.1. □

Note that L can also be accepted by a one-counter machine. Some applications of Theorem 5.1 are given in [12].

6. Some decidable problems

In this section we consider properties of pictures such that the existence of some picture with this property in the language or the appearance of this property in all pictures of the language are decidable.

Let L_0 be the set of all words describing pictures for which start- and endpoints coincide. Note that the associated set of Parikh vectors

$$\Psi(L_0) = \{c(1,1,0,0) + c'(0,0,1,1): c, c' \in N_0\}$$

is semilinear. Furthermore, every word w with $\Psi(w) \in \Psi(L_0)$ is an element of L_0. This is, L_0 is characterized by $\Psi(L_0)$.

To prove the following theorem we need the following facts on semilinear sets (see [9]):

- Semilinear sets are closed under intersection.
- Membership and the emptiness problem are decidable for semilinear sets.

Theorem 6.1. *For context-free picture languages B, it is decidable whether or not all pictures of B are trees.*

Proof. Let G be a context-free grammar with $Pic(G) = B$. First assume B contains a picture p which is not a tree. Let $p = pic(w)$ for some $w \in L(G)$. Then w contains a subword w' such that the first and the last letter of w' are not inverse to each other and $pic(w')$ is a closed curve (take a minimal description of a closed curve in p). Thus w' is contained in

$$L = SUB(L(G)) \setminus (r\pi^*l \cup l\pi^*r \cup u\pi^*d \cup d\pi^*u \cup \{\lambda\})$$

($SUB(K)$ denotes the set of all subwords of the language K) and $\Psi(L) \cap \Psi(L_0)$ is not empty.

On the other hand, if $\Psi(L) \cap \Psi(L_0)$ is nonempty then B contains a nontree since

the words v with $\Psi(v) \in \Psi(L_0)$ describe closed curves, i.e., $L(G)$ contains $v'vv''$ which describes a nontree.

Since G is context-free, L is also context-free. Thus $\Psi(L)$ is semilinear, and therefore the emptiness of $\Psi(L) \cap \Psi(K_0)$ is decidable. Hence the existence of a nontree is decidable. □

We now present a lemma which will be used in the proof of the following theorem.

Lemma 6.2. *For any regular language L the language $fac(L)$ defined by*

$$fac(L) = \{w\colon\ w^n \in L \text{ for some } n > 0\}$$

is regular.

Proof. Let $A = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton with $L(A) = L$. For $q, q' \in Q$, we set

$$L(q, q') = \{w\colon\ \delta(q, w) = q'\}.$$

A word of the form w^n is in L if and only if there is a sequence of states $q_0, q_1, \ldots, q_n$ with $q_n \in F$ such that, for $0 \leq i < n-1$, $w \in L(q_i, q_{i+1})$. With any such sequence we associate the intersection

$$L(q_0, q_1) \cap L(q_1, q_2) \cap \cdots \cap L(q_{n-1}, q_n),$$

which is a regular language. The union of all these intersections is $fac(L)$. Although there are infinitely many sequences, there are only finitely many possibilities for the intersection language, since intersection is commutative and idempotent. This is, $fac(L)$ can be considered as a finite union of regular languages, which finishes the proof. □

Moreover, we need the following facts on matrix languages (see [7]).

Fact 6.3. (i) *The emptiness problem is decidable for matrix languages.*

(ii) *The family of matrix languages is closed under intersection with regular languages and union.*

Furthermore, we have to recall the notion of retreats as redundant retreats and their properties.

A retreat is a word of the form $winv(w)$. For example, the word $w = rudr$ contains a retreat. It is easy to see that there is no retreat-free description of $pic(w)$. In contrast, for every simple curve, there exists a retreat-free description. (If the simple curve is not closed, such a description starts at one of the nodes of degree 1 and ends at the other.) However, a word of the form $w = xyinv(y)yz$ contains the subword $yinv(y)y$, which describes a subpicture of $pic(w)$ which is traversed three

times. Therefore this subword can be replaced by y, i.e., $pic(w)=pic(xyz)$. Such subwords are called redundant retreats.

The language $r-red(w)$ of all words that can be formed by successive deletions of redundant retreats from w is formally given by the conditions:

- $w\in r-red(w)$,
- if $xyinv(y)yz\in r-red(w)$, then $xyz\in r-red(w)$.

A word is called redundant-retreat-free if no further deletions are possible, i.e., if it does not contain a subword of the form $yinv(y)y$. Of course, every retreat-free word is redundant-retreat-free, however, *rudr* is redundant-retreat-free and not retreat-free.

We extend the operators $r-red$ and inv to languages by

$$inv(L)=\{inv(w):\ w\in L\},\qquad r-red(L)=\bigcup_{w\in L} r-red(w).$$

Obviously, we have the following

Fact 6.4. *For every language L, $pic(L)=pic(inv(L))=pic(r-red(L))$.*

From the well-known closure properties of the family of regular languages and [10, Theorem 3.2] (see [4] for a related result) we obtain

Fact 6.5. *For every regular language L, $inv(L)$ and $r-red(L)$ also are regular.*

Let us now study simple curves. We already mentioned that, for every simple curve, there exists a retreat-free description. Yet there are descriptions that are redundant-retreat-free according to our definition and not retreat-free. In general, we decompose a description w of a simple curve into maximal retreat-free subwords $w=y_1y_2\dots y_m$. For $1\le j<m$, let z_j be the suffix of y_j with

$$|z_j|=min\{|y_j|,|y_{j+1}|\},$$

and for $1<j\le m$, let x_j be the prefix of z_j with

$$|x_j|=min\{|y_{j-1}|,|y_j|\}.$$

It is easy to see that z_jx_{j+1} is a retreat, i.e., $z_j=inv(x_{j+1})$ for $1\le j<m$. Now assume that $x_j=y_j=z_j$ for some $1<j<m$, then the subword $z_{j-1}y_jx_{j+1}$ of w is a redundant retreat. This implies the following

Fact 6.6. *For redundant-retreat-free words w with the above factorization there exists an $n\in\{0,1,\dots,m\}$ such that*

$$|y_1|<|y_2|<\dots<|y_n| \quad and \quad |y_{n+1}|\le|y_{n+2}|\le\dots\le|y_m|.$$

Moreover, n can be choosen as the maximal index j with $x_j\neq y_j$, if such a j exists, and $n=1$ otherwise. Furthermore, we have $|y_n|\ge|y_j|$ for every $j\in\{1,2,\dots,m\}$ and $pic(y_n)=pic(w)$.

Additionally to the above definition let $w' = inv(y_n)inv(y_{n-1})\dots inv(y_1)$ and $w'' = inv(y_m)inv(y_{m-1})\dots inv(y_{n+1})$. Then it is easy to see that $r-red(w'ww'')$ contains a retreat-free description of $pic(w)$. Combining this with Facts 6.4 and 6.5 we get

Fact 6.7. *Let B be a picture language, and let L be a regular string language such that $B = SUPER(Pic(L))$. Then the regular language $r-red(\pi^* L \pi^*)$ describes B and contains a retreat-free description of every simple curve in B.*

Theorem 6.8. *For regular picture languages B the following problems are decidable:*

(i) *Does B contain a rectangle?*

(ii) *Are all pictures of B rectangles?*

(iii) *Does B contain a convex curve?*

Proof. (i) Let G be a regular grammar with $B = Pic(G)$. If we decide that a straight line or the empty picture is a rectangle, we may check in the first step whether $L(G)$ contains a word in $\{u,d\}^* \cup \{r,l\}^*$. Obviously, the answer is positive if and only if B contains a rectangle of height 0 or width 0. If the answer is negative, we have to search for a rectangle with height ≥ 1 and width ≥ 1.

By Fact 6.5, there is a finite automaton A such that

$$L(A) = r-red(L(G)).$$

By Fact 6.4, $L(A)$ is a description of B.

We shall construct a matrix grammar H such that

(a) every $w \in L(H)$ describes a rectangle and

(b) every minimal $w \in L(A)$ that describes a rectangle is contained in $L(H)$,

where minimal means that, for every $w' \in L(A)$ with $pic(w') = pic(w)$, $|w| \leq |w'|$. Then B contains a rectangle if and only if $L(H) \cap L(A)$ is nonempty, which is decidable by Fact 6.3.

For every such minimal word $w \in L(A)$ we shall give a characterization by a finite sequence $ch(w)$. Since there will be only a finite set of such sequences ch, we only need to give a matrix grammar H_{ch} that generates every minimal $w \in L(A)$ with $ch(w) = ch$. Then we are done by $L(H) = \bigcup_{ch} L(H_{ch})$.

As a standard description of a rectangle we choose the clockwise description $w_{h,i} = u^h r^i d^h l^i$, and we also use the counter-clockwise description $inv(w_{h,i})$.

Let A have k states. Note that every minimal word w in $L(A)$ is redundant-retreat-free (Fact 6.4). If w describes a simple curve, we can write w as a concatenation of retreat-free subwords as in Fact 6.6. It is easy to see that, by the minimality of w, every node in $pic(w)$ is traversed at most k times in the prefix $y_1 y_2 \dots y_n$ and at most k times in the suffix $y_{n+1} y_{n+2} \dots y_m$ of w. Especially, $m \leq 2k$ and every y_j traverses its startpoint no more than k times.

Furthermore, if w describes a rectangle, then every y_j is a subword of $w_{h,i}^{k+1}$ or of $inv(w_{h,i})^{k+1}$. Thus in order to characterize y_j we give its first letter $fi(y_j)$, its last letter $la(y_j)$, a flag fl_n of value cl or co denoting clockwise or counter-clockwise for

$w_{h,i}^{k+1}$ or $inv(w_{h,i})^{k+1}$, respectively, and the number $min(j)$ which gives the minimal number t such that y_j is a subword of $w_{h,i}^{k+1}$ or $inv(w_{h,i})^{k+1}$, respectively. Note that there are only finitely many possible characterizations, since $min(j) \leq k+1$. Then the whole word w can be characterized by

$$ch(w) = (fi(y_1), min(1), fi(y_2), min(2), \ldots, fi(y_n), min(n), fl_n,$$
$$la(y_n), min(n+1), la(y_{n+1}), \ldots, min(m), la(y_m)).$$

From this sequence all characterizations of the words y_j can be constructed, since $la(y_j) = inv(fi(y_{j+1}))$ and $fl_j \neq fl_{j+1}$. Since $m \leq 2k$ there is only a finite set of possible sequences $ch(w)$.

For example, the sequence

$$ch = (u, 2, u, 3, co, l, 1, d)$$

represents all words of the form

$$w = u^p r^i d^h l^i u^h r^i d^q u^q l^i d^h r^i u^h l^s r^s d^t,$$

where the relations $p \leq h$, $q \leq h$, $s \leq i$, $t \leq h$ have to hold to ensure that w describes a rectangle, and the relations $1 \leq p$, $1 \leq q$, $1 \leq s$, $1 \leq t$ have to hold to ensure $ch(w) = ch$. For the above (representative) example of ch we construct the matrix grammar $H_{ch} = (V, \pi, M, S)$ where

$$V = \{S, R, L, D, U, U_1, D_1, U_2, R_2, R', L', U', D', U_1', D_1', U_2', D_2'\}.$$

and M is the following set of matrices:

- initial matrix:

$$m_0 = (S \to U_1 RDLURU_2 (LDRU)^* R_2 D_1),$$

- horizontal matrices:

$$h_1 = (R \to rR'),$$
$$h_2 = (L \to lL'),$$
$$h_3 = (R_2 \to lR_2' r),$$
$$h_4 = (R_2 \to R_2'),$$
$$h_5 = (R' \to R, L' \to L, R' \to R, L' \to L, R' \to R, L' \to L, R' \to R, R_2' \to R_2),$$

- vertical matrices:

$$v_1 = (U_1 \to U_1'),$$
$$v_2 = (U_1 \to uU_1'),$$
$$v_3 = (D \to dD'),$$
$$v_4 = (U \to uU'),$$
$$v_5 = (U_2 \to dU_2' u),$$

$$\upsilon_6 = (U_2 \to U_2'),$$
$$\upsilon_7 = (D_1 \to dD_1'),$$
$$\upsilon_8 = (D_1 \to D_1'),$$
$$\upsilon_9 = (U_1 \to U_1, D' \to D, U' \to U, U_2' \to U_2, D' \to D, U' \to U, D' \to D, D_1' \to D_1),$$

- final matrix:

$$m_X = (U_1 \to u, R \to r, D \to d, L \to l, U \to u, R \to r, U_2 \to ud, L \to l, D \to d,$$
$$R \to r, U \to u, L \to l, D \to d, R \to r, U \to u, R_2 \to lr, D_1 \to d).$$

It is straightforward to show that all words generated by H_{ch} describe rectangles and that all redundant-retreat-free descriptions w of rectangles with $ch(w) = ch$ are in fact generated by H_{ch}. Such a construction can be done for every ch, which finishes the proof of (i).

(ii) By pumping techniques (see, e.g., [16]) it is easy to see that a regular picture language which contains only rectangles has to be finite, except, of course, rectangles of height 0 or width 0. Therefore, for a regular grammar G with $Pic(G) = B$ we have the following algorithm:

(1) Construct the regular grammar G' with $L(G') = L(G) \setminus (\{u, d\}^* \cup \{r, l\}^*)$. (The words in $\{u, d\}^* \cup \{r, l\}^*$ describe degenerated rectangles.)

(2) Decide whether or not $Pic(G')$ is finite (see [18]).

(3) If the answer is "no", then G generates a nonrectangle, else check whether or not all pictures are rectangles.

(iii) For a given regular grammar G, we have to check whether or not $Pic(G) = B$ contains a simple convex curve.

We first construct a regular language L that contains a minimal description of a simple closed convex curve if and only if B contains a simple convex curve. By definition of convexity, B contains a simple convex curve if and only if $SUPER(B)$ contains a simple closed convex curve. By Fact 6.3, the regular language $L' = r\text{-}red(\pi^* L(G) \pi^*)$ describes $SUPER(B)$ and contains a retreat-free description of every simple convex curve in $SUPER(B)$. The language $L'\pi^*$ contains a retreat-free description of a simple closed convex curve that starts and ends in the same node if and only if L' contains a retreat-free description of a simple convex curve. Note that every such description of a simple closed convex curve p has to be of the form w^i where w is a minimal description of p and $i > 0$. Hence we set $L = fac(L'\pi^*)$, and then L contains a minimal description of w of every simple closed convex curve in $Pic(L'\pi^*)$. By Lemma 6.2, L is a regular set.

In [2] it is shown that it is decidable whether or not a regular language L contains a word q which describes a simple closed convex curve and contains no subword describing a closed curve. Obviously, B contains a convex curve if and only if L contains such a word q. This completes the proof. □

7. Conclusion

We have shown that the universal subpicture problem is undecidable for almost all pictures and regular grammars. As an application we proved the undecidability of the question whether or not a regular picture language contains a picture with a given property for some geometrical and graph-theoretical properties.

From the literature it is known that the problems of "classical" formal language theory are also almost undecidable for regular picture languages. Therefore stripe and 3-way languages are introduced in [15, 19] in order to obtain better properties. A picture language B is called a *stripe language* if there are constants k, d_1 and d_2 such that, for all pictures $p \in B$ and all nodes (m,n) of p,

$$km + d_1 \leq n \leq km + d_2.$$

It is called a 3-*way language* if $B = Pic(G)$ for some grammar which generates only words over $\{r,d,u\}$. By [11, 19] there are string representation theorems for the pictures of a regular picture language which is a stripe or 3-way language. These techniques can be used as in [5, 11, 19] to obtain that all the problems mentioned above, which are undecidable for regular picture languages, are decidable for regular stripe languages and regular 3-way languages. On the other hand, the proofs in [5] are given by linear stripe languages (sometimes slight modifications are necessary), and thus all above problems are undecidable for linear stripe languages.

Moreover, undecidability of a problem for picture languages implies the undecidability of this problem for generalized picture languages. But we do not know what is the situation with respect to generalized regular picture languages and the problems:

(1) existence of a rectangle,
(2) existence of a convex curve,
(3) are all pictures rectangles,
(4) are all pictures trees.

These problems are shown to be decidable for regular picture languages ((4) even for context-free picture languages), and we note that (2) is undecidable for linear stripe languages.

Acknowledgement

We are grateful to K. Slowinski who pointed out an error in an earlier version of this paper.

References

[1] D. Beauquier, An undecidable problem about rational sets and contour words of polyominoes, Inform. Process. Lett. 37 (1991) 257–263.

[2] D. Beauquier, M. Latteux and K. Slowinski, A decidability result about convex polyominoes, Tech. Rep. IT 214, University Lille, Lille (1991).

[3] C. Berge, Theory of Graphs and Applications (Wiley, New York, 1962).

[4] J.C. Birget, Basic techniques for two-way finite automata, in: Proceedings LIPT Spring School Formal Properties of Finite Automata and Applications, Lecture Notes in Computer Science 386 (Springer, Berlin, 1989) 56–64.

[5] J. Dassow, Graph-theoretical properties and chain code picture languages, J. Inform. Process. Cybernet. 25 (1989) 423–433.

[6] J. Dassow, On the connectedness of pictures in chain code picture languages, Theoret. Comput. Sci. 81 (1991) 289–294.

[7] J. Dassow and Gh. Paun, Regulated Rewriting in Formal Language Theory (Springer, Berlin, 1989).

[8] H. Freeman, Computer processing of line-drawing images, Comput. Surveys 6 (1974) 57–97.

[9] S. Ginsburg, The Mathematical Theory of Context-Free Languages (McGraw-Hill, New York, 1966).

[10] F. Hinz, Classes of picture languages that cannot be distinguished in the chain code concept and the deletion of redundant retreats, in: Proceedings STACS'89, Lecture Notes in Computer Science 349 (Springer, Berlin, 1989) 132–143.

[11] F. Hinz, The equivalence problem for three-way picture languages, Manuscript.

[12] F. Hinz and J. Dassow, An undecidability result for regular languages and its application to regulated rewriting, Bull. EATCS 38 (1989) 168–174.

[13] F. Hinz and E. Welzl, Regular chain code picture languages with invisible lines, Report 252, IIG, Technische Graz, Graz (1988).

[14] J.E. Hopcroft and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation (Addison-Wesley, Reading, MA, 1979).

[15] C. Kim, Complexity and decidability for restricted classes of picture languages, Theoret. Comput. Sci. 73 (1990) 295–311.

[16] C. Kim, Picture iteration and picture ambiguity, J. Comput. System Sci. 40 (1990) 289–306.

[17] C. Kim and I.H. Sudborough, The membership and equivalence problem for picture languages, Theoret. Comput. Sci. 52 (1987) 177–192.

[18] H.W. Maurer, G. Rozenberg and E. Welzl, Using string languages to describe picture languages, Inform. and Control 54 (1982) 155–185.

[19] I.H. Sudborough and E. Welzl, Complexity and decidability of chain code picture languages, Theoret. Comput. Sci. 36 (1985) 175–202.