Retreat bounded picture languages

Changwook Kim

School of Computer Science, University of Oklahoma, 200 Felgar Street, Norman, OK 73019, USA

Communicated by G. Rozenberg
Received March 1990
Revised October 1993

Abstract


A well-known string encoding of 2D pictures is to use the picture alphabet \( \pi = \{u, d, r, l\} \), where \( u \) (\( d \), \( r \), and \( l \)) means “draw a unit-line by moving the pen up (down, right, and left) from the current point”. A word over \( \pi \) is \( k \)-retreat-bounded \((k \geq 0)\) if it describes a picture in such a manner that the maximum distance of left-moves, ignoring up- and down-moves, from a rightmost point of any partially drawn picture is bounded by \( k \). A set of such words forms a \( k \)-retreat-bounded language.

A \( k \)-retreat-bounded picture language is a set of pictures described by a \( k \)-retreat-bounded language. There is a \( 1 \)-retreat-bounded regular picture language (\( 1 \)-retreat-bounded vertical-stripe linear picture language) for which the membership problem is NP-complete. The membership problem can be solved in \( O(n^4) \) time for each \( k \)-retreat-bounded nonvertical-stripe context-free picture language. The inclusion and intersection-emptiness problems are undecidable for a \( 1 \)-retreat-bounded regular picture language and a \( 2 \)-retreat-bounded regular picture language. The equivalence and ambiguity problems are undecidable for \( 2 \)-retreat-bounded regular picture languages. The picture recognition algorithm presented is the first polynomial-time algorithm in the literature for a reasonably large subclass of context-free picture languages and the NP-completeness and undecidability results improve the previously known such results for regular picture languages with no structural restriction imposed.

1. Introduction

A word over the alphabet \( \pi = \{u, d, r, l\} \) describes a picture in the two-dimensional Cartesian plane if the symbol \( u \) (\( d \), \( r \), and \( l \)) is interpreted as “draw a unit-line by moving the pen up (down, right, and left) from the current point”. A word that describes a picture in this manner is called a (Freeman’s) chain code \([9, 10]\) and has been used widely for pictorial pattern description and recognition. The chain-code model is also similar to the picture-description method used in “turtle geometry” \([1]\),
where the trace left by a turtle (or robot) while moving in the plane is interpreted to be a picture. There are other picture-description models, some of which are similar to the chain-code scheme, and we refer to [26] for various formal picture-description methods and [11] for general syntactic pattern-recognition methods.

The properties of (chain code) picture languages, i.e., sets of chain-encoded pictures, have been studied intensively, following the initial work of Feder [8] and a further systematic study of Maurer et al. [24, 25]. In these studies, picture languages are classified by the Chomsky hierarchy of their description languages and the complexity and decidability of various decision problems are investigated. For example, the picture membership (or recognition) problem of deciding whether a given picture is described by a word in a language $L \subset \pi^*$ is NP-complete for regular picture languages [28]. The equivalence, inclusion, intersection-emptiness, and ambiguity problems are undecidable for regular picture languages [20, 22, 28], where, e.g., the equivalence problem is to decide whether or not two languages over $\pi$ describe exactly the same set of pictures. Also studied much are graph-theoretical decision problems of whether or not a picture language contains some or all pictures with a certain graph-theoretical or geometrical property [4–7, 15], chain-code optimization problems of finding a shortest word for a given picture or transforming a given word into a shorter one describing the same picture [13, 27], descriptional complexity of deciding the worst-case ratio of the size of a picture in a picture language and the shortest length of its description words in the corresponding description language [14, 24, 25, 28], and other formal-language-theoretic properties of picture-description languages [2, 3, 16, 18, 20].

Due to the inherent intractability of major decision problems for picture languages, further studies have focused on finding restricted subclasses of picture languages that have better complexity and decidability results and still have power for applications. Sudborough and Welzl [28] introduced the class of stripe picture languages, whose pictures fit into a stripe defined by two parallel lines in the plane, and showed that for stripe regular picture languages the membership problem can be solved in linear time and the equivalence, inclusion, and intersection-emptiness problems are decidable. Kim [21] studied the class of three-way picture languages, which are described by languages over a three-letter subset of $\pi$, and showed, among others, that the membership problem can be solved in polynomial time for three-way context-free picture languages. Kim and Sudborough [23] extended the class of three-way picture languages to the class of $k$-reversal-bounded picture languages ($k \geq 0$), whose pictures are described by making no more than $k$ left-to-right or right-to-left reversals in the plane. The membership problem can be solved in polynomial time for $k$-reversal-bounded regular picture languages (for each $k \geq 0$) and is NP-complete for 1-reversal-bounded stripe linear picture languages.

The present paper introduces another extension of three-way picture languages, called $k$-retreat-bounded picture languages ($k \geq 0$). The extension is made by allowing retreats of distance up to $k$, where retreat means “moving left” in the plane. Namely, a $k$-retreat-bounded picture language is a picture language which is described by a
(k-retreat-bounded) language \( L \subseteq \pi^* \) such that each of its words describes a picture in such a manner that the maximum distance of left-moves, ignoring up- and down-moves, from a rightmost point of any partially drawn picture is bounded by \( k \). (One must distinguish between the concept of a “local” retreat as defined in [24], i.e., a word from the set \( \{ud, du, rl, lr\} \), and the concept of a “global” retreat as used in this paper.) Thus, the family of 0-retreat-bounded picture languages is exactly the family of three-way picture languages defined by the alphabet \( \{u, d, r\} \). The retreat as defined in this paper is a natural concept in chain code picture languages since it is used to restrict the head motion of a picture-drawing device, as in other restricted classes of picture languages.

The paper is organized as follows. Section 2 contains necessary notations and notions for chain code picture languages. In Section 3, retreat-bounded picture languages are defined formally and some basic decidability/undecidability results and hierarchy theorems are proved. In Section 4, it is shown that there is a 1-retreat-bounded regular picture language (1-retreat-bounded vertical-stripe linear picture language) for which the membership problem is NP-complete. In Section 5, the membership problem is shown to be solved in \( O(n^3) \) time for each \( k \)-retreat-bounded nonvertical-stripe context-free picture language. In Section 6, several new undecidability results for regular picture languages are presented. It is shown that the inclusion and intersection-emptiness problems are undecidable for a 1-retreat-bounded regular picture language and a 2-retreat-bounded regular picture language and that the equivalence and ambiguity problems are undecidable for 2-retreat-bounded regular picture languages. The polynomial-time membership result in Section 5 identifies a reasonably large subclass of context-free picture languages for which the membership problem can be solved efficiently. The NP-completeness and undecidability results in Sections 4 and 6 are stronger than the known such results for regular picture languages with no structural restriction imposed.

2. Preliminaries

We shall assume that the reader is familiar with the basics of the theory of formal languages and automata [19]. For the notions related to picture languages, we mostly follow [24]. (The notion of stripe picture languages is from [28].) The empty word is denoted by \( \lambda \). For a set \( A \), \( |A| \) denotes the cardinality of \( A \) and \( 2^A \) denotes the power set of \( A \). The empty set is denoted by \( \emptyset \).

Let \( \mathcal{Z} \) be the set of integers. The set \( \mathcal{Z} \times \mathcal{Z} \) is denoted by \( M_0 \). Let \( v=(m, n) \) be an element of \( M_0 \). The \( x \)-component of \( v \) is \( x(v)=m \) and the \( y \)-component of \( v \) is \( y(v)=n \). The up-, down-, right-, and left-neighbors of \( v \) are respectively \( u(v)=(m, n+1) \), \( d(v)=(m, n-1) \), \( r(v)=(m+1, n) \), and \( l(v)=(m-1, n) \). \( M_1 \) denotes the set \( \{(v, b(e)) \mid v \in M_0 \text{ and } b \in \{u, d, r, l\}\} \).

An attached basic picture \( p \) is a finite subset of \( M_1 \). The point set of \( p \) is \( V(p) = \{v \in M_0 \mid \exists v' \in p \text{ for some } v' \in M_0 \} \). (Thus, an attached basic picture is a graph, whose set of vertices is \( V(p) \) and whose set of edges is \( p \). We shall consider only
connected pictures, i.e., connected graphs.) An attached drawn picture \( q \) is a triple \((p, s, e)\), where \( p \) is an attached basic picture and either \( p = \emptyset \) and \( s = e \in M_0 \) or \( p \neq \emptyset \) and \( s, e \in V(p) \); \( s \) is called the start point and \( e \) is called the end point.

Consider a translation operation that moves an attached basic or drawn picture vertically and/or horizontally, preserving its shape and size (and also the relative locations of its start and end points in the case of an attached drawn picture). It certainly defines an equivalence relation. The equivalence class containing an attached basic (drawn) picture \( p \), denoted by \([p]\) (\( \langle p \rangle \)), is called the unattached version of \( p \) and is simply referred to as basic (drawn) picture.

Let \( \pi \) be the alphabet \{\( u, d, r, l \)\}. Each word \( w \in \pi^* \) is called a picture-description word or simply \( \pi \)-word. The drawn picture described by \( w \), denoted by \( dpic(w) \), is defined inductively by:

\[
dpic(zb) = \langle p, s, e \rangle,
\]

\[
dpic(\varepsilon) = \langle \emptyset, (0, 0), (0, 0) \rangle,
\]

where \( z \in \pi^* \), \( b \in \pi \), and \( dpic(z) = \langle p, s, e \rangle \). If \( dpic(w) = \langle p, s, e \rangle \), then the basic picture described by \( w \), denoted by \( bpic(w) \), is \([p]\).

Fig. 1 shows the drawn picture described by \( udldr'ud \), where a little circle in the figure indicates the start point of the picture and a little square indicates the end point. The picture with the little circle and the little square removed from the picture in Fig. 1 is the basic picture described by the same \( \pi \)-word.

Each language \( L \) over \( \pi \) is called a picture-description language or simply \( \pi \)-language. The drawn (basic) picture language described by \( L \) is \( dpic(L) = \{dpic(w) \mid w \in L\} \) (\( bpic(L) = \{bpic(w) \mid w \in L\} \)). Each grammar (automaton) \( A \) such that \( L(A) \) is a \( \pi \)-language is called a picture-description grammar (automaton) or simply \( \pi \)-grammar (\( \pi \)-automaton). The drawn (basic) picture language described by \( L(A) \) is simply denoted by \( dpic(A) \) (\( bpic(A) \)).

A drawn (basic) picture language \( P \) is a regular (linear, context-free, context-sensitive, or recursively enumerable) picture language if there is a regular (linear, context-free, context-sensitive, or recursively enumerable) \( \pi \)-language \( L \) such that \( dpic(L) = P \) (\( bpic(L) = P \)).

For real numbers \( \mu, d_1, d_2 \), where \( d_1 < d_2 \), the \((\mu, d_1, d_2)\)-stripe, denoted by \( M^{(\mu, d_1, d_2)}_0 \), is the set \( \{(i, j) \in M_0 \mid \mu i + d_1 \leq j \leq \mu i + d_2 \} \). As a special case, it is possible that \( \mu = \infty \), in

![Fig. 1. A drawn picture described by udldr'ud.](image-url)
Retreat bounded picture languages

Fig. 2. A (0.5, -3.5, 1.5)-stripe picture.

which cases \( M^\infty_{0.5, -3.5, 1.5} = \{(i,j) \in M_0 \mid d_1 \leq i \leq d_2\} \) represents a vertical stripe of width \( d_2 - d_1 \). \( M_{1.5, -3.5, 1.5} \) denotes the set \( \{(v, v') \in M_1 \mid v, v' \in M_{0.5, -3.5, 1.5}\} \).

A drawn picture \( q = \langle p, (0, 0), e \rangle \) with \( p \subseteq M_{\mu, d_1, d_2} \) is called a drawn \((\mu, d_1, d_2)\)-stripe picture. A basic picture \( p \) is a basic \((\mu, d_1, d_2)\)-stripe picture if there is an attached basic picture \( p' \) such that \( p = \{p'\} \) and \( p' \subseteq M_{\mu, d_1, d_2} \). For example, the drawn picture \( q \) in Fig. 2 fits into the \((0.5, -3.5, 1.5)\)-stripe, and so, \( q \) is a \((0.5, -3.5, 1.5)\)-stripe picture. A drawn (basic) picture language \( P \) is a drawn (basic) \((\mu, d_1, d_2)\)-stripe picture language (or simply stripe picture language) if every picture in \( P \) is a drawn (basic) \((\mu, d_1, d_2)\)-stripe picture; \( P \) is a vertical-stripe (nonvertical-stripe) picture language if \( \mu = \infty \) (\( \mu \neq \infty \)).

3. Retreat-bounded picture languages

In this section, we shall define retreat-bounded picture languages formally and prove some of their basic properties, in particular hierarchy theorems. Similar results for reversal-bounded picture languages were proved in [23]. We note that, for reversal-bounded picture languages, the class of context-sensitive picture languages is properly included in the class of recursively enumerable picture languages, while they are identical for retreat-bounded picture languages (Theorems 3.4 and 3.6). Other theorems presented in this section have the same form as for reversal-bounded picture languages.

Let \( k \) be a nonnegative integer. A \( \pi \)-word \( w \) is a \( k \)-retreat-bounded word if each subword \( w' \) of \( w \) has the property that \( \#_l(w') - \#_r(w') \leq k \), where \( \#_\sigma(w'), \sigma \in \{l, r\} \), denotes the number of \( \sigma \)'s in \( w' \). Namely, \( w \) is \( k \)-retreat-bounded if it draws a picture in such a manner that the maximum distance of left-moves, ignoring up- and
down-moves, from a rightmost point of any partially drawn picture is bounded by $k$. A $\pi$-language $L$ is a $k$-retreat-bounded language if each word in $L$ is $k$-retreat-bounded. A $\pi$-grammar ($\pi$-automaton) $A$ is a $k$-retreat-bounded grammar (automaton) if $L(A)$ is a $k$-retreat-bounded language. A drawn (basic) picture language $P$ is a $k$-retreat-bounded picture language if $P = \text{dpic}(L)$ ($P = \text{bpic}(L)$) for a $k$-retreat-bounded language $L$.

**Theorem 3.1.** For each $k \geq 0$, it is decidable (undecidable) whether or not a context-free (context-sensitive) $\pi$-language $L$ is a $k$-retreat-bounded language.

**Proof.** The set $R$ of all $k$-retreat-bounded $\pi$-words can be easily seen to be a regular $\pi$-language. For example, $R$ can be accepted by a finite automaton with an additional counter of size $k$ which pushes a special symbol $\odot$ into its counter on the input symbol $l$, pops on $r$ (if the counter is not empty), ignores $u$ and $d$, and accepts the given input if and only if it reaches the end of the input tape while using only the given $k$ cells in its counter. $L$ is $k$-retreat-bounded if and only if $L \subseteq R$, which is decidable (undecidable) if $L$ is a context-free (context-sensitive) language. \(\square\)

Let $D_{\text{REG}} (D_{\text{LIN}}, D_{\text{CF}}, D_{\text{CS}},$ and $D_{\text{RE}})$ be the family of drawn regular (linear, context-free, context-sensitive, and recursively enumerable) picture languages and let $B_{\text{REG}}, B_{\text{LIN}}, B_{\text{CF}}, B_{\text{CS}},$ and $B_{\text{RE}}$ be the corresponding families of basic picture languages. For $k \geq 0$, $L \in \{D, B\}$, and $X \in \{\text{REG, LIN, CF, CS, RE}\}$, let $L_X(k)$ denote the family of $k$-retreat-bounded picture languages of the type $L_X$. Let $L_X() = \bigcup_{k \geq 0} L_X(k)$.

**Theorem 3.2** (Maurer et al. [24]). For each $L \in \{D, B\}$, $L_{\text{REG}} \not\subseteq L_{\text{LIN}} \not\subseteq L_{\text{CF}} \not\subseteq L_{\text{CS}} \not\subseteq L_{\text{RE}}$.

**Theorem 3.3** (Kim and Sudborough [23]). For each $L \in \{D, B\}$, $L_{\text{REG}}(0) \not\subseteq L_{\text{LIN}}(0) \not\subseteq L_{\text{CF}}(0) \not\subseteq L_{\text{CS}}(0) \not\subseteq L_{\text{RE}}(0)$.

**Theorem 3.4.** For each $k \geq 1$ and each $L \in \{D, B\}$, $L_{\text{REG}}(k) \not\subseteq L_{\text{LIN}}(k) \not\subseteq L_{\text{CF}}(k) \not\subseteq L_{\text{CS}}(k) = L_{\text{RE}}(k)$.

**Proof.** Let $w_k = r^k d^k$ and let $L_{k, 1} = \{r^i d^i w_k | i \geq 0\}$, $L_{k, 2} = \{r^i d^i r^j d^j w_k | i \geq 0, j \geq 0\}$, and $L_{k, 3} = \{r^i d^i r^j w_k | i \geq 0\}$. It can be seen (e.g., by using the pumping arguments for picture languages [20]) that the picture language described by $L_{k, 1}$ ($L_{k, 2}$, and $L_{k, 3}$) is not a regular (linear, and context-free) picture language. However, $L_{k, 1}$ ($L_{k, 2}$, and $L_{k, 3}$) is clearly a $k$-retreat-bounded linear (context-free, and context-sensitive) $\pi$-language. Therefore, $L_{\text{REG}}(k) \not\subseteq L_{\text{LIN}}(k) \not\subseteq L_{\text{CF}}(k) \not\subseteq L_{\text{CS}}(k)$. The proof for $L_{\text{CS}}(k) = L_{\text{RE}}(k)$, $k \geq 1$, is identical to the proof for $L_{\text{CS}} = L_{\text{RE}}$ in [24, Theorems 4.5 and 4.7]. \(\square\)

**Theorem 3.5.** For each $k \geq 0$, each $L \in \{D, B\}$, and each $X \in \{\text{REG, LIN, CF, CS, RE}\}$, $L_X(k) \not\subseteq L_X(k + 1)$.
Proof. The language $L_k = \{ r^{k+1}dl^{k+1} \}$ is a $(k+1)$-retreat-bounded regular $\pi$-language, for each $k \geq 0$. However, there is no $k$-retreat-bounded recursively enumerable $\pi$-language $L$ such that $\text{dpic}(L) = \text{dpic}(L_k)$ ($\text{bpic}(L) = \text{bpic}(L_k)$). □

Theorem 3.6. For each $\mathcal{L} \in \{ \mathcal{D}, \mathcal{B} \}$, $\mathcal{L}_{\text{REG}}(\pi) \subseteq \mathcal{L}_{\text{LIN}}(\pi) \subseteq \mathcal{L}_{\text{CS}}(\pi) = \mathcal{L}_{\text{RE}}(\pi)$.

Proof. Similar to the proof of Theorem 3.4, but we need to note the following. There is a three-way recursively enumerable picture language $P$ that cannot be described by any three-way context-sensitive $\pi$-language (Theorem 3.3). However, $P$ can be described by a 1-retreat-bounded context-sensitive $\pi$-language [24, Theorem 4.5]. □

Theorem 3.7. For each $\mathcal{L} \in \{ \mathcal{D}, \mathcal{B} \}$ and each $X \in \{ \text{REG}, \text{LIN}, \text{CF}, \text{CS}, \text{RE} \}$, $\mathcal{L}_X(\pi) \subseteq \mathcal{L}_X$.

Proof. It is clear that $\mathcal{L}_X(\pi) \subseteq \mathcal{L}_X$. The proper inclusion follows from the fact that the picture language described by $\pi^*$ is in $\mathcal{L}_{\text{REG}} - \mathcal{L}_{\text{RE}}(\pi)$. □

4. NP-completeness results

The (picture) membership problem is to determine whether or not $p \in \text{dpic}(L)$ ($p \in \text{bpic}(L)$) for a drawn (basic) picture $p$ given as input and a fixed $\pi$-language $L$. The membership problem is NP-complete for regular picture languages [28] and 1-reversal-bounded stripe linear picture languages [23] and can be solved in polynomial time for stripe regular picture languages [28], reversal-bounded regular picture languages [23], and three-way (i.e., 0-retreat-bounded) context-free picture languages [21]. We shall show in this section that the membership problem is NP-complete for 1-retreat-bounded regular (1-retreat-bounded vertical-stripe linear) picture languages. Our discussion on the membership problem will continue in the next section where a polynomial-time algorithm for retreat-bounded nonvertical-stripe context-free picture languages is presented.

Theorem 4.1. There is a 1-retreat-bounded regular $\pi$-language for which the drawn and basic versions of the picture membership problem are NP-complete.

Proof. We shall construct a 1-retreat-bounded regular $\pi$-language $L_0$ and reduce the Bounded Post Correspondence Problem (BPCP), which is NP-complete [12], to the picture membership problem for $L_0$. BPCP is to decide, for two lists $x=(x_1, x_2, \ldots, x_n)$ and $y=(y_1, y_2, \ldots, y_m)$, $n \geq 1$, of nonempty words over an alphabet $\Sigma$ and a positive integer $k \leq n$, whether there exists a sequence $(i_1, i_2, \ldots, i_m)$, $1 \leq m \leq k$ and $1 \leq i_j \leq n$ for all $j=1, 2, \ldots, m$, such that $x_1x_{i_1} \cdots x_{i_m} = y_1y_{i_1} \cdots y_{i_m}$. A sequence of integers satisfying this condition is called a solution for $(x, y, k)$. As the membership problem for context-free picture languages is in NP [17, 22], this will prove the theorem.
Let \( x = (x_1, x_2, \ldots, x_n) \) and \( y = (y_1, y_2, \ldots, y_n) \) be two lists of nonempty words over the alphabet \( \Sigma = \{a, b\} \) and let \( k \leq n \) be a positive integer, given as an instance of BPCP. We construct a drawn picture \( p \) such that \( (x, y, k) \in \text{BPCP} \) if and only if \( p \in \text{dpic}(L_0) \). Note that \( L_0 \) does not depend on the given instance of BPCP. The proof is given for the drawn case only. (The reader can easily check that \( (x, y, k) \in \text{BPCP} \) if and only if \( p' \in \text{bpic}(L_0) \), where \( p' \) is the basic picture obtained from \( p \) by removing the start and end points.) We shall first show the construction of \( p \) since this seems easier to explain \( L_0 \).

The picture \( p \) will be constructed by using the four types of picture components shown in Fig. 3, which we call \( A-, B-, C-, \) and \( D-\) components, as indicated below each component, or simply primitive components. The \( A- \) and \( B- \) components will be used to describe the symbols \( a, b \in \Sigma \) and the \( C- \) and \( D- \) components will be used to separate two subpictures with certain meanings. Let \( A, B, C, \) and \( D \) be any fixed \( \pi \)-words describing these primitive components.

We first describe the structure of \( p \) informally. Refer to an example of the picture \( p \) shown in Fig. 4, constructed from \( (x, y, k) \) with \( x = (a, ba) \), \( y = (aa, b) \), and \( k = 2 \). The picture \( p \) can be divided into four parts:

1. Many primitive components attached below the bottommost long horizontal line-segment containing the start and end points, which we call the \( x- \) field;
2. Many such components attached to the left of the leftmost long vertical line-segment containing the start point, which we call the \( y- \) field;
3. Many such components attached to the right of the rightmost long vertical line-segment containing the end point, which we call the verification-field; and
4. The rest of \( p \) contained in the biggest rectangle in \( p \), three sides of which are the long horizontal and vertical line-segments used to define the \( x- \), \( y- \), and verification-fields, which we call the work-field.

Call each portion of the \( x- \) or \( y- \) field surrounded by two consecutive (i.e., nearest) \( D- \) components a block. The \( x- \) field (\( y- \) field) consists of \( k \) identical blocks, laid out horizontally (vertically) starting from the start point. A block consists of \( n \) subblocks,
each of which contains $A$- and/or $B$-components representing the symbols $a$ and $b$, surrounded by two $C$- and/or $D$-components, where a $D$-component is used only for an outermost subblock. For each $i = 1, 2, \ldots, n$, the $i$th subblock of a block in the $x$-field ($y$-field), the first subblock being the leftmost (bottommost) subblock, contains the picture components corresponding to the string $x_i$ ($y_i$). Thus, a block in the $x$-field ($y$-field) contains a complete information about $x(y)$. 

Fig. 4. An example of the picture $p$. 
Upon construction of the x- and y-fields, the work- and verification-fields are constructed as follows. For each primitive component in the x-field, its left and right vertical line-segments are stretched all the way to the topmost long horizontal line-segment. So, the work-field is completely filled vertically. For each A- or B-component in the y-field, its six horizontal line-segments are stretched all the way to the rightmost long vertical line-segment, and in addition, the same kind of primitive component is attached to the right of the work-field. However, for each C- or D-component in the y-field, its three horizontal line-segments are stretched only up to the long vertical line-segment stretched up from the matching primitive component in the x-field, where the matching occurs one by one between such primitive components in the x- and y-fields, outwards from the start point.

Now, we define the formal construction of \( p \). Introduce new symbols \( \#_i \) and \( \#_j \), and define the following words:

\[
X = x_1 \#_1 x_2 \#_2 \cdots x_{n-1} \#_i x_n \#
\]

\[
Y_i = y_1 S_i^1 y_2 S_i^2 \cdots y_{n-1} S_i^{n-1} y_n S_i^n, \quad 1 \leq i \leq k.
\]

The height of the work-field is \( x = 7(y_1 y_2 \cdots y_n + n)k \). To draw the x-field and the vertical lines in the work-field, we define a homomorphism \( h_x \) as follows:

\[
h_x(a) = ru^xrd^x A, \quad h_x(b) = ru^xrd^x B,
\]

\[
h_x(\#) = ru^xrd^x C, \quad h_x(\#) = ru^xrd^x D.
\]

Let \( m_i = 2(|x_1 x_2 \cdots x_i| + i), \quad 1 \leq i \leq n \). Each A- or B-component in the y-field is stretched to the right as many as \( \beta = km_n + 1 \) lines. Each C- or D-component located just above the \( j \)-th subblock of the \( i \)-th block in the y-field is stretched to the right as many as \( \gamma_{i,j} = (i-1)m_n + m_j \) lines, where \( 1 \leq i \leq k \) and \( 1 \leq j \leq n \). To draw the y- and verification-fields and the horizontal lines in the work-field, we define a homomorphism \( h_y \) as follows:

\[
h_y(a) = u^x A(rA)^{\beta} l^{\beta}, \quad h_y(b) = u^x B(rB)^{\beta} l^{\beta},
\]

\[
h_y(S_i^j) = u^x C(rC)^{\gamma_{i,j}} l^{\gamma_{i,j}}, \quad 1 \leq i \leq k, \quad 1 \leq j \leq n - 1,
\]

\[
h_y(S_i^n) = u^x D(rD)^{\gamma_{i,j}} l^{\gamma_{i,j}}, \quad 1 \leq i \leq k.
\]

Let \( p = \text{dpic}(D \cdot h_y(Y_1 Y_2 \cdots Y_k) \cdot d^x \cdot h_x(X^k)) \). It is not difficult to see that the transformation from \((x, y, k)\) into \( p \) can be done in polynomial time.

We describe now the construction of \( L_0 \). Informally, each word \( w \) in \( L_0 \) "tries" to create an instance of \( \text{BPCP} \), say \((x, y, k)\), in the picture form and to test if \((x, y, k) \in \text{BPCP} \). The latter condition is tested by checking if there is an integer \( m \leq k \) such that the string formed by concatenating the symbols represented in the \( i \)-th subblock of the \( j \)-th block of the \( y \)-field, over all \( j = 1, 2, \ldots, m \), is identical to the string formed by concatenating the symbols represented in the \( i \)-th subblock of the \( j \)-th block of the \( x \)-field, over all \( j = 1, 2, \ldots, m \). The picture drawn by \( w \) will have the right shape corresponding to an instance of \( \text{BPCP} \) if and only if this
condition is satisfied. (In other words, there is a bijection between the set of all positive instances of BPCP and the subset of dpic($L_0$) consisting of the pictures having the right shape for instances of BPCP. Observe that this construction of $L_0$ is sufficient for our purpose). The picture-drawing task performed by $w$ consists of many subtasks of similar nature which we call the (upward and downward) vertical drawings, done in sequence by moving from left to right in the plane. Vertical drawings can be grouped into four types, which we call INITIALIZE, PROPAGATE, MATCH, and VERIFY (or simply I, P, M, and V). As noted later, INITIALIZE (VERIFY) consists of a single upward (downward) drawing and is the first (last) vertical drawing performed by $w$. On the other hand, PROPAGATE and MATCH will each consist of a downward drawing followed by an upward drawing. These vertical drawings will be combined together to form subblocks, blocks, and eventually the $x$-field together with other fields.

The first vertical drawing determines the structure of the $y$-field and "guesses" a solution to BPCP. A solution is guessed by "marking" some of the primitive components in the $y$-field and "unmarking" others. The markings and unmarkings are done by drawing only a part of the right vertical line-segments of the primitive components. Fig. 5 shows the marked and unmarked versions of the $A$-, $B$-, $C$-, and $D$-components.

Fig. 5. The marked and unmarked versions of $A$-, $B$-, $C$-, and $D$-components.
D-components (the top four and the bottom four, respectively) used in the first vertical drawing. Let \( A, \tilde{A}, \tilde{B}, \tilde{C}, \) and \( D, \tilde{A}, \tilde{B}, \tilde{C}, \) and \( \tilde{D} \) be any fixed \( \pi \)-words describing the marked (unmarked) versions of the \( A-, B-, C-, \) and \( D \)-components.

A subblock of the \( y \)-field which is (is not) a part of the solution ending with a \( K \)-component (\( K \in \{ C, D \} \)) can be described by a \( \pi \)-word in \( Y^K \) (\( \tilde{Y}^K \)) defined by the following regular expressions:

\[
Y^K = (uA + uB)^* u\tilde{K}, \quad \tilde{Y}^K = (u\tilde{A} + u\tilde{B})^* u\tilde{K}.
\]

Now, the first vertical drawing can be done by choosing an arbitrary \( \pi \)-word from the following regular expression:

\[
I = D(Y^*_C(Y^*_C Y^*_D + Y^*_D))^* (Y^*_C Y^*_D)^*.
\]

After the \( y \)-field has been constructed as described above, a number of downward and upward drawings are performed. The purpose of these drawings is to modify or preserve the shape of the rightmost edge of the partially drawn picture, depending on whether or not a primitive component in the \( x \)-field currently considered is involved in the solution to BPCP. A downward (upward) drawing is done by moving downward (upward) along two consecutive long vertical line-segments, the right one of which is stretched from the left (right) vertical line-segment of a primitive component in the \( x \)-field, and by drawing all unit-size horizontal line-segments that lie between the two consecutive long vertical line-segments. Between a pair of downward and upward drawings, the corresponding primitive component in the \( x \)-field will be drawn. Thus, as \( L_0 \) will consist only of \( 1 \)-retreat-bounded \( \pi \)-words, the vertical line-segments not drawn on the rightmost edge of a partially drawn picture must be drawn exactly in the next vertical drawing, if the picture drawn is going be a valid one.

In Figs. 6–9, we show variations of the marked and unmarked versions of the \( A-, B-, C-, \) and \( D \)-components which will be used for vertical drawings. Let each picture component in Figs. 6–9 be described by any fixed \( \pi \)-word which we shall call as indicated below the picture component. In the names of such \( \pi \)-words, the subscripts \( p, m, \) and \( v \) indicate \textsc{propagate}, \textsc{match}, and \textsc{verify}, and the superscripts \( d \) and \( u \) indicate the downward and upward drawings.

What we are going to show is that, for each marked or unmarked component created in the \( y \)-field, the subsequent vertical drawings well-propagate its shape toward right up to the correct position where a matching or verification occurs and eventually draw a valid picture corresponding to an instance of BPCP, by using the picture components in Figs. 6–9 strictly in the order shown from left to right and by using other rules combining the vertical drawings explained later, if and only if the picture drawn corresponds to a positive instance of BPCP.

While moving from left to right in the plane, if the primitive component created in the \( x \)-field is an \( A-, B \)-component which is guessed to be not a part of the solution, then the shape of the right edge of the partially drawn picture will be preserved (or propagated) in the corresponding downward and upward drawings. The shape of
a primitive component in the $y$-field can be propagated in the downward (upward) drawing by using a $\pi$-word from $E_d$ ($E_u$) defined, for each $\sigma \in \{d, u\}$, by:

$$E_\sigma = \{ \tilde{X}_p^\sigma, \tilde{X}_p^\sigma | X \in \{A, B, C, D\} \} \cup \{ X_p^\sigma | X \in \{A, B\} \} \cup \{ \sigma_0 \},$$

where $\tilde{X}_p^\sigma$ or $\tilde{X}_p^\sigma$ ($X \in \{A, B, C, D\}$) propagates the shape of a primitive component originally drawn in the $y$-field and $X_p^\sigma$ ($X \in \{A, B\}$) or $\sigma_0$ (for a C- or D-component) propagates the shape of a primitive component for which a "matching" occurred previously (matching will be explained shortly). Now, the two vertical drawings to propagate the shape can be done by choosing a $\pi$-word from the following regular expression:

$$P = r(E_d d)^+ A + B (u E_u) +.$$

Every $A$- or $B$-component in the $x$-field which is guessed to be a part of the solution will be matched with a primitive component of the same type in the $y$-field. The matching will be performed by modifying the shape of the right edge of the portion of the partially drawn picture which is located at the intersection of the vertical and horizontal respectively line-segments stretched from these two primitive components.
This can be done by using $\bar{A}_m^u$ and $\bar{A}_m^u$ in sequence or $\bar{B}_m^d$ and $\bar{B}_m^u$ in sequence. The primitive component in the $y$-field involved in this matching should be the bottom-most not-yet-matched one. To ensure this, we define, for each $\sigma \in \{u, d\}$,

$$E^\sigma = E \sigma - \{\bar{A}_m^\sigma, \bar{B}_m^\sigma\}.$$

Now, the two vertical drawings to match a $K$-component ($K \in \{A, B\}$) can be done by choosing a $\pi$-word from the following regular expression:

$$M_K = r(E(d)E(d)^*K(d)^*\{uE_u^\sigma uK_m^u(uE_u)^\}^*).$$

Every $C$- or $D$-component in the $x$-field will be matched with a primitive component of the same type in the $y$-field. However, depending on whether or not it is guessed to be a part of the solution, the matching will be done in different ways. If it is guessed to be a part of the solution, then the matching will be performed by using $\bar{C}_m^d$ and $\bar{C}_m^u$ in sequence or $\bar{D}_m^d$ and $\bar{D}_m^u$ in sequence. Otherwise, the matching will be performed by using $\bar{C}_m^d$ and $\bar{C}_m^u$ in sequence or $\bar{D}_m^d$ and $\bar{D}_m^u$ in sequence. In either case, the
Fig. 8. Variations of the marked and unmarked C-components.

primitive component in the y-field involved in this matching should be the bottom-most not-yet-matched one. To ensure this, we define, for each $\sigma \in \{u, d\}$,

$$E^*_\sigma = E_\sigma - \{\bar{C}_p^\sigma, \bar{D}_p^\sigma, \bar{D}_p^\sigma\}.$$ 

Now, the two vertical drawings to match a K-component ($K \in \{C, D\}$) can be done by choosing a $\pi$-word from the following regular expressions:

$$M_{\bar{K}} = r(E_d)K_m^d (E_d^u)^* K(u E_u)^* uK_m^u (u E_u)^*.$$

$$M_{\bar{K}} = r(E_d)K_m^d (E_d^u)^* K(u E_u)^* uK_m^u (u E_u)^*.$$

Various types of pairs of vertical drawings explained above can be grouped into subblocks of the x-field, together with the corresponding portion of the work-field. A subblock which is (is not) a part of the solution ending with a K-component ($K \in \{C, D\}$) can be drawn by a $\pi$-word in $X_{\bar{K}}$ ($X_{\bar{K}}$) defined by the following regular expressions:

$$X_{\bar{K}} = (M_{\bar{A}} + M_{\bar{B}})^+ M_{\bar{K}}, \quad X_{\bar{K}} = P^+ M_{\bar{K}}.$$
These subblocks can now be grouped into blocks of the $x$-field. There are two types of blocks, i.e., a block containing a subblock which is a part of the solution and a block not containing such a subblock. These blocks together with the corresponding portion of the work-field can be drawn by using the following regular expressions:

$$W_d = X_C^\#(X_CX_C^\#X_d^\# + X_d^\#), \quad W_d = X_C^\#X_d^\#.$$  

At the right end of the $x$-field, all $A$- and $B$-components in the $y$-field should have been transformed into the right shape. This will be checked by using the $\pi$-words $\tilde{A}_e$, $\tilde{A}_v$, $\tilde{B}_e$, and $\tilde{B}_v$. Thus, the last vertical drawing is done by choosing an appropriate $\pi$-word from the following regular expression:

$$V = (d^\top(\tilde{A}_e d + \tilde{B}_e d)^\#(d^\top(\tilde{A}_e d + \tilde{B}_e d)^\#(d^\top(\tilde{A}_e d + \tilde{B}_e d)^\#)^*)^+.$$  

Let $L_0 - I W_d^* W_d^\# V$. Fig. 10 shows an example of the first few vertical drawings performed by a $\pi$-word in $L_0$. From left to right, we show the first vertical drawing that guesses $y = (aa, b)$, $k = 2$, and the solution $(2, 1)$, the second and third vertical drawings that propagate the shape of the right edge of the previously drawn picture,
Retreat bounded picture languages

Fig. 10. An example of the vertical drawings drawn by a word in \( L_0 \).

the fourth and fifth vertical drawings that match the \( C \)-component not involved in the solution, the sixth and seventh vertical drawings that match the \( B \)-component involved in the solution, the eighth and ninth vertical drawings that match the \( A \)-component involved in the solution, and the tenth and eleventh vertical drawings that match the \( D \)-component involved in the solution. The second through fifth vertical drawings determine simultaneously \( x_1 = a \) and the sixth through eleventh vertical
drawings determine simultaneously $x_2 = ba$. Observe that, by repeating a similar sequence of vertical drawings in the second block of the $x$-field while matching $x_1 = a$ instead of $x_2 = ba$ and by completing with a downward drawing performed by an appropriate $\pi$-word from $V$, the picture shown in Fig. 4 can be successfully drawn.

Suppose that the given instance $(x, y, k)$ of BPCP has a solution, say $(i_1, i_2, \ldots, i_m)$, $1 \leq m \leq k$. The picture $p$ constructed from $(x, y, k)$ can be drawn by a $\pi$-word $w = w_1w_2w_3$ in $L_0$ as follows. The substring $w_1$ is a word from $I$ drawing the $y$-field of $p$ in such a manner that, for each $j = 1, 2, \ldots, m$, the $A$- and $B$-components in the $i_j$th subblock of the $j$th block in the $y$-field and the $C$- or $D$-component drawn immediately above them are marked and all others are unmarked. The substring $w_2$ is a word from $W_D^+, W_D^*$ drawing the $x$-field and the work-field of $p$ in such a manner that, for each $j = 1, 2, \ldots, m$, the $A$- and $B$-components in the $i_j$th subblock of the $j$th block in the $x$-field are matched with the corresponding primitive components in the $y$-field and that all $C$- and $D$-components in the $x$-field are matched with the corresponding primitive components in the $y$-field. Since $x_1, x_2, \ldots, x_m = y_1, y_2, \ldots, y_m$, all primitive components involved in this choice of solution in the $y$-field can be exactly matched with the choice of primitive components in the $x$-field. It should be obvious how the shape of the right edge of a marked or unmarked component in the $y$-field is preserved by the vertical drawings when it is not involved in a matching. The substring $w_3 \in V$ simply draws the verification-field by using the $\pi$-words $\tilde{A}_e$ and $\tilde{B}_e$ for the $A$- and $B$-components originally marked in the $y$-field and the $\pi$-words $\tilde{A}_u$ and $\tilde{B}_u$ for the $A$- and $B$-components originally unmarked in the $y$-field. Clearly, $p = \text{dpic}(w)$.

Now, suppose that $p = \text{dpic}(w)$ for a $\pi$-word $w$ in $L_0$. The word $w$ draws a sequence of vertical drawings, moving from left to right in the plane. It is easy to see that each upward (downward) drawing must start from a point on the bottommost (topmost) long horizontal line-segment of $p$ and end at a point on the topmost (bottommost) long horizontal line-segment of $p$. Furthermore, since a primitive component is uniquely determined by its unit-size horizontal line-segments which are drawn exactly once when $w$ draws $p$, all vertical drawings must correctly guess the right types of primitive components horizontally. That is, the types of primitive components created in the $y$-field must be appropriately propagated towards right up to the correct positions.

Consider the maximum-length prefix $w'$ of $w$ which is in $I$. Then, $w'$ draws the $y$-field of $p$ with marked and unmarked primitive components. Note that every picture drawn by a word in $I$ satisfies the following basic construction rules:

1. A subblock is either a marked one consisting only of marked components or an unmarked one consisting only of unmarked components.

2. A block contains either exactly one marked subblock or no marked subblock.

3. The first block contains a marked subblock.

4. All blocks containing marked subblocks are consecutive.

Let us assume that the first $m$ blocks of the $y$-field contain marked subblocks and for each $j = 1, 2, \ldots, m$, the $i_j$th subblock of the $j$th block is marked.
When the marked and unmarked primitive components created in the y-field are propagated towards right by vertical drawings, the picture components shown in the upper rows of Fig. 6–9 can be used only for the originally marked components in the y-field; otherwise, some vertical line-segments in p cannot be drawn. Similarly, the picture components shown in the lower rows of Fig. 6–9 can be used only for the originally unmarked components in the y-field. For a marked A- or B-component in the y-field, say \(A\), the shape of the right edge is propagated and modified towards right by using \(A_p^d\) and \(A_p^u\) in sequence zero or more times, \(A_m^d\) and \(A_m^u\) in sequence exactly once, \(A_p^d\) and \(A_p^u\) in sequence zero or more times, and then finally \(A\), exactly once. No other combination can fill the vertical line-segments completely. For an unmarked A- or B-component in the y-field, say \(A\), the shape of the right edge is propagated by using \(A_p^d\) and \(A_p^u\) in sequence some number of times and then finally \(A\), exactly once, and there is no other combination to fill the vertical line-segments completely. For a marked C- or D-component in the y-field, say \(C\), the only way to fill the vertical line-segments completely is to use \(C_p^d\) and \(C_p^u\) in sequence some number of times and then finally \(C_m^d\) and \(C_m^u\) in sequence exactly once, and for an unmarked C- or D-component in the y-field, say \(C\), the only way to fill the vertical line-segments completely is to use \(C_p^d\) and \(C_p^u\) in sequence some number of times and then finally \(C_m^d\) and \(C_m^u\) in sequence exactly once.

Note that a C- or D-component in the y-field cannot match twice with the components of the same type in the x-field since there are exactly the same number of such components in the x-field and the y-field of p. Now, it is sufficient to prove that:

1. The marked A- and B-components in the y-field are matched with the primitive components of the same type drawn in the x-field, in sequence outwards from the start point; and
2. The C- and D-components in the y-field are matched with the primitive components of the same type drawn in the x-field, in sequence outwards from the start point.

The proof follows from these two since the x-field of p also satisfies the basic construction rules. That is, since there are \(m\) marked C- and/or D-components in the y-field, there must be exactly \(m\) such primitive components in the x-field that would match with those in the y-field, outwards from the start point. As the marked A- and/or B-components in the subblocks defined by such C- and/or D-components are also matched, exactly outwards from the start point, it is obvious that \((i_1, i_2, \ldots, i_m)\) is a solution for \((x, y, k)\).

To prove the first claim, let \(Z_1\) and \(Z_2\) be two marked A- and/or B-components in the y-field and let \(Z_1\) be located farther from the start point than \(Z_2\). Assume to the contrary that \(Z_1\) is matched before \(Z_2\) is matched. Assume, without loss of generality, that \(Z_1\) is a marked A-component. The matching of \(Z_1\) with an A-component in the x-field, say \(Z_3\), is performed by a word in \(M_T\). Such a word assumes that there is no unmatched marked A- or B-component below the intersection of \(Z_1\) and \(Z_3\). That is, the downward drawing does not use \(A_p^d\) or \(B_p^d\) below the intersection and the upward drawing does not use \(A_p^u\) or \(B_p^u\) below the intersection. Now once can easily observe
that some vertical line-segment located at the intersection of $Z_2$ and $Z_3$ is not drawn. So, $p$ cannot be drawn by $w$; a contradiction.

The proof of the second claim is similar. If $Z_1$, $Z_2$ are two marked $C$- and/or $D$-components in the $y$-field and if we follow the same argument as in the first case, the horizontal line-segments at the intersection of $Z_2$ and $Z_3$ are not drawn (since $d^6$ and $u^6$ would be used in sequence at the intersection, assuming that the types of the primitive components in the $y$-field are correctly propagated towards right), a contradiction to the fact that $Z_2$ is matched after $Z_1$ is matched. $\square$

**Theorem 4.2.** There is a 1-retreat-bounded linear $\pi$-language describing a vertical-stripe picture language for which the drawn and basic versions of the picture membership problem are NP-complete.

**Proof.** We refer to [23], where a 1-reversal-bounded linear $\pi$-language describing a stripe picture language, for which the picture membership problem is NP-complete, was constructed. It is easy to modify the picture components used in there so that the language becomes a 1-retreat-bounded linear $\pi$-language describing a vertical-stripe picture language for which the picture membership problem is NP-complete. $\square$

5. A polynomial-time recognition algorithm

We show that the picture membership problem can be solved in $O(n^4)$ time for each $k$-retreat-bounded context-free $\pi$-language describing a nonvertical-stripe picture language. (Note that, by Theorem 4.2, the picture membership problem is NP-complete for a 1-retreat-bounded linear $\pi$-language describing a vertical-stripe picture language.) The recognition algorithm presented in this section is based on the dynamic-programming algorithm for three-way context-free picture languages given in [21]. In fact, we shall simply add a few new concepts to the algorithm in [21] and show that this can handle retreat-bounded nonvertical-stripe context-free picture languages efficiently.

**Theorem 5.1.** Let $k$ be a nonnegative integer. For every $k$-retreat-bounded context-free $\pi$-language $L$ describing a nonvertical-stripe picture language, the drawn and basic versions of the picture membership problem for $L$ can be solved in $O(n^4)$ time.

**Proof.** Let $\mu$, $d_1$, $d_2$ be real numbers such that $\mu \neq \infty$ and $d_1 < d_2$. Let $M = (Q, \pi, \Gamma, \delta, i, Z_0, f)$ be a $k$-retreat-bounded PDA such that $\text{dpic}(M)$ is a $(\mu, d_1, d_2)$-stripe picture language; $Q$ is the set of states, $\pi$ is the input alphabet, $\Gamma$ is the stack alphabet, $i \in Q$ is the initial state, $Z_0 \in \Gamma$ is the bottom marker of the stack, $f \in Q$ is the final state, and $\delta: Q \times (\pi \cup \{\lambda\}) \times \Gamma \rightarrow 2^Q \times \Gamma^*$ is the transition function. (It is decidable whether or not $M$ is a $k$-retreat-bounded PDA (Theorem 3.1) and whether or not $\text{dpic}(M)$ is a $(\mu, d_1, d_2)$-stripe picture language [28].) We shall assume without loss of
Let \( q = (p, (0, 0), e) \) be an attached drawn picture given as input. Assume that \( \langle q \rangle \) is a \((\mu, d_1, d_2)\)-stripe picture; otherwise, \( \langle q \rangle \notin \text{dpic}(M) \). (Note that the set of all \((\mu, d_1, d_2)\)-stripe pictures is a regular picture language and its membership problem can be solved in linear time [28].) For \( v, v' \in V(p) \), let \( U(v, v') = M_1^{(v, d_1, d_2)} \cap M_1^{(v', d_1, d_2)} \) and define:

\[
W(v, v') = U(v, v') - U(v, v') - U(v, v').
\]

An arbitrary attached drawn picture \( q' = (p', s', e') \) is called a \( q \)-valid picture if \( p' \equiv p \) and \( p' \cap W(s', e') = p \cap W(s', e') \). (Observe that, if \( \langle q \rangle = \text{dpic}(x) \) for some \( x \in L(M) \) and \( x = yzw \), where \( \text{dpic}(y) = \langle p, (0, 0), s' \rangle \) and \( \text{dpic}(z) = \langle p', s', e' \rangle \), then \( (p', s', e') \) should be a \( q \)-valid picture since \( \langle q \rangle \) is a \((\mu, d_1, d_2)\)-stripe picture and \( M \) is a \( k \)-retreat-bounded PDA.) For \( a, b \in Q \) and \( A \in \Gamma \), \( (q', a, A, b) \) is a \( q \)-valid tuple if \( q' \) is a \( q \)-valid picture and \( M \) can go from state \( a \) with \( A \) alone in its stack to state \( b \), emptying the stack and consuming a \( \pi \)-word \( z \) with \( \text{dpic}(z) = \langle q' \rangle \) from the input tape.

Let \( X \) be the set of all \( q \)-valid tuples. Clearly, \( X \) is a finite set. It is easy to see that \( \langle q \rangle \in \text{dpic}(M) \) if and only if \( (q, i, Z_0, f) \in X \). Therefore, it is sufficient to construct \( X \) in order to test if \( \langle q \rangle \in \text{dpic}(M) \). To construct \( X \), we define a mapping \( \psi : X \times 2^X \times 2^X \) such that, for all \( \tau \in X \) (let \( \tau = ((p', s', e'), a, A, b) \) and \( E \in 2^X \), \( \psi(\tau, E) \) consists of all \( q \)-valid tuples \( \bar{\tau} \) defined by:

1. \( \bar{\tau} = ((v, s') \cup p', v, v'), a', B, b') \) if \( (a, A, B) \in \delta(a', \sigma, B) \), \( \text{dpic}(\sigma) = \langle \{v, s'\}, v, s' \rangle \), and \( (p', e', v'), b, B, b') \in E \); and
2. \( \bar{\tau} = ((v, v') \cup p', v, e'), a', A, b) \) if \( (b', B, A) \in \delta(a', \sigma, A) \), \( \text{dpic}(\sigma) = \langle \{v, v'\}, v, v' \rangle \), and \( (p', v', s'), b', B, a) \in E \).

Now, we state our recognition algorithm. It simply constructs \( X \) by a standard dynamic programming as reflected in the definition of \( \psi \). Run the following to test if \( \langle q \rangle \in \text{dpic}(M) \):

1. Check that \( \langle q \rangle \) is a \((\mu, d_1, d_2)\)-stripe picture;
2. Add all \( q \)-valid tuples corresponding to pop-transitions of \( M \) into an initially empty set \( X \) and into an initially empty list \( T \);
3. While \( T \) is not empty repeat the following: remove an element \( \tau \) from \( T \), compute \( Y = \psi(\tau, X) \), and add each \( \bar{\tau} \in Y \) into \( X \) and \( T \);
4. Output “yes” if \( (q, i, Z_0, f) \in X \) and output “no” otherwise.

This process indeed constructs the set \( X \) and solves the drawn version of the picture membership problem for \( M \). The proof of this claim is basically identical to the proof of Lemma 3.6 in [21]; we shall briefly sketch it here. (The above algorithm is structurally identical to the one for three-way context-free picture languages in [21] even though it is in a simpler form. We only adjusted the notion of \( q \)-valid tuples in order to handle retreat-bounded nonvertical-stripe context-free picture languages, i.e., to bound \( |X| \) by a polynomial function in \( |p| \).) Let us call a \( q \)-valid tuple \( (q', a, A, b) \) an \((m, q)\)-valid tuple if \( M \) can go from state \( a \) with \( A \) alone in its stack to state \( b \), emptying
the stack and consuming a π-word \( w \) with \( \text{dpic}(w) = \langle q' \rangle \), after exactly \( m \) moves. Certainly, each \( q \)-valid tuple is an \((m, q)\)-valid tuple for some \( m \geq 1 \). Thus, it is sufficient to show that each \((m, q)\)-valid tuple can be added to \( X \) by the above algorithm. This can be proved by an induction on \( m \). The induction basis \((m = 1)\) is clearly true since all \((1, q)\)-valid tuples are added to \( X \) in Step (2). Assume that the claim is true for all \( m \leq m_0 \), for some \( m_0 \geq 1 \), and consider any \((m_0 + 1, q)\)-valid tuple, say \( \tau = \langle q', a, A, b \rangle \).

The computation of \( M \) which realizes the \((m_0 + 1, q)\)-validness of \( \tau \) consists of (i) pushing a symbol, \( B \), into its stack, (ii) popping this symbol \( B \) after some number of moves, and (iii) popping the symbol \( A \) exposed on top of the stack after some number of moves. Clearly, the subcomputations of \( M \) corresponding to (ii) and (iii) define \( q \)-valid tuples, say \( \tau_1 \) and \( \tau_2 \). Let \( \tau_1 \) be \((m_1, q)\)-valid and \( \tau_2 \) be \((m_2, q)\)-valid. Then \( m_1, m_2 \leq m_0 \). By the induction hypothesis, both \( \tau_1 \) and \( \tau_2 \) are eventually added to \( X \) (and to \( T \); see Steps (2) and (3)). When the last one of them is removed from \( T \) in Step (3), the other one is in \( X \) since it was already added to \( T \) (and so, to \( X \), too) and then removed. Now, it is easy to see from the definition of \( \psi \) that \( \tau \) is added to \( X \) at this point.

Now, we consider the running time of this process. Let \( n = |p| \). Step (1) takes \( O(n) \) time. Each \( q \)-valid picture considered in Step (2) is a one line picture. So, Step (2) takes \( O(n) \) time. For \( v, v' \in V(p) \), the number of distinct \( q \)-valid pictures of the form \( \langle p', v, v' \rangle \) is \( O(1) \) since both \( U(v, v) \) and \( U(v', v') \) are of constant size. Therefore, \( |X| = O(n^2) \). In Step (3), each \( q \)-valid tuple which is added to \( X \) is added to \( T \) exactly once. Therefore, Step (3) is iterated \( O(n^2) \) times. Note that, in the computation of \( \psi(\tau, X) \), the number of \( q \)-valid tuples in \( X \) that need to be considered is \( O(n) \) and the picture union operation takes \( O(n) \) time. Therefore, \( \psi(\tau, X) \) can be computed in \( O(n^2) \) time. It follows that Step (3) takes \( O(n^4) \) time. Step (4) takes \( O(n^3) \) time. Altogether, the algorithm takes \( O(n^4) \) time.

If the input is an attached basic picture \( p \), we run a slightly modified version of the above algorithm. Assume without loss of generality that \( p \neq \emptyset \) and \( (0, 0) \in V(p) \). We first move \( p \) vertically so that it fits into the \((\mu, d_1, d_2)\)-stripe (if Step (1) of the algorithm is true) and then run Steps (2) and (3) to check finally in Step (4) whether or not \( X \) contains a tuple \( \langle (p, s, e), i, Z_0, f \rangle \) for some \( s, e \in V(p) \). This also takes \( O(n^4) \) time.

6. Undecidability results

We consider some standard decision problems for picture languages other than the membership problem, such as equivalence, inclusion, intersection emptiness, and ambiguity. For example, the equivalence problem is to decide whether or not \( \text{dpic}(L_1) = \text{dpic}(L_2) \) (or \( \text{bpic}(L_1) = \text{bpic}(L_2) \)) for two π-languages \( L_1 \) and \( L_2 \) and the ambiguity problem is to decide whether or not there exist two distinct words in a π-language \( L \) describing the same drawn (or basic) picture. These problems are undecidable for regular picture languages \([20, 22, 28]\) and decidable for stripe regular
picture languages [28]. Other known undecidability results include the equivalence problem for a three-way stripe regular picture language and a three-way stripe linear picture language [21], the intersection emptiness and ambiguity problems for three-way stripe linear picture languages [20, 21], and the intersection emptiness problem for a three-way stripe regular picture language and a 1-reversal-bounded stripe linear picture language [23].

In this section, we shall prove some new undecidability results for (retreat-bounded) "regular" picture languages. The proofs are based on the ideas presented already in [20] and [22], and so, we shall only sketch the principal modifications. In order to parallel the proofs with those in [20] and [22] and to better understand the reductions, we shall assume that the retreat bounds are imposed vertically rather than horizontally, i.e., replace the roles of $r$ and $l$ by $d$ and $u$, respectively, in considering the retreat size.

For the proofs of the undecidability results we shall state later, we use reductions from the blank-tape halting problem for oblivious one-tape Turing machines. All reductions are based on a single transformation from a Turing machine $M$ into a 1-retreat-bounded regular $\pi$-language $R_M$, and so, we shall describe it first. In the sequel, an oblivious one-tape Turing machine (or simply a Turing machine) is a deterministic one-tape Turing machine that moves its tape head either left or right at every transition and changes the direction of its head motion exactly when it scans a blank symbol. We assume without loss of generality that such a machine moves its tape head left initially, never prints a blank symbol on the tape, never enters its initial state again once it makes a transition, and if it halts then it does so right after it reverses the direction of its head motion from left to right. The problem of whether or not a Turing machine with three tape symbols $a$, $b$, and $c$ (the blank symbol) halts on the blank tape is undecidable [19].

Let $M$ be an arbitrary Turing machine with tape symbols $a$, $b$ and $c$. These symbols will be represented in the pictures described by $R_M$ by using the picture components shown in Fig. 11. Let $A$, $B$, and $C$ be any fixed $\pi$-words describing these picture components.

A picture in $dpic(R_M)$ will describe the actions of $M$ by representing successive passes of the tape head over the written portion of the tape through successive rows of the picture containing the picture primitives (described by) $A$, $B$, and $C$. In a pass, rewriting of the symbols is represented by drawing consecutively the picture components for reading and writing the symbols in the direction $M$ moves on its tape. If $M$ reads a blank symbol, then it reverses the direction of its head motion. In this case,

![Fig. 11. Picture components for the symbols a, b, and c.](image-url)
Fig. 12. An example of a sequence of pictures drawn by a word in $R_M$.

The picture simulation of $M$ will continue in the next row, after drawing an auxiliary picture component $C$ in order to mark the next reversal point.

Fig. 12 shows an example of a sequence of pictures drawn by a $\pi$-word in $R_M$ corresponding to the first seven transitions of $M$. These picture-drawing tasks indicate the following rewriting actions by $M$: (a) read $c$, write $b$, and move left, (b) read $c$, write $a$, and move right (reverse the head motion), (c) read $b$, write $a$, and move right, (d) read
Retreat hounded picture languages

c, write a, and move left (reverse the head motion), (e) read b, write a, and move left, (f) read a, write b, and move left, and (g) read c, write a, and move right (reverse the head motion).

Let the transition function \( \delta \) of \( M \) be defined by: \( \delta(p, d) = (p', d', D) \) if \( M \) in state \( p \) scanning the symbol \( d \) from the tape replaces \( d \) by \( d' \), changes its state to \( p' \), and moves left or right depending on whether \( D = L \) or \( D = R \). Let \( h(a) = A \) and \( h(b) = B \). Define a right-linear grammar \( G_M = (Q, \pi, P, s) \), where \( Q \) is the set of states of \( M \), \( s \in Q \) is the initial state of \( M \), and \( P \) consists of the following productions (\( p, q \in Q \) and \( d, d' \in \{a, b\} \)):

\[
\begin{align*}
& s \to Cd^{12}C \delta^8 h(d)l^3p & \text{if } \delta(s, c) = (p, d, L), \\
& p \to Cd^8C \delta^8 h(d)q & \text{if } \delta(p, c) = (q, d, R), \\
& p \to r^4h(d)h(d')q & \text{if } \delta(p, d) = (q, d', R), \\
& p \to r^4Cd^8C \delta^8 h(d)l^4q & \text{if } \delta(p, c) = (q, d, L), \\
& p \to h(d)l^4h(d')l^4q & \text{if } \delta(p, d) = (q, d', L), \\
& p \to \vdash & \text{if } p \text{ is an accepting state of } M. 
\end{align*}
\]

Let \( R_M = L(G_M) \). Then \( R_M \) is a 1-retreat-bounded regular \( \pi \)-language. Observe that \( G_M \) correctly keeps track of the state transitions of \( M \). It simulates the computation of \( M \) correctly if its guessing of the tape symbols is also correct at every step. It follows that \( M \) halts on the blank tape if and only if there is a picture in \( \text{dpic}(R_M) \) such that every picture component for reading a symbol in the picture is identical to the picture component located immediately above it (which is the symbol written last on the tape cell). The reader can see that, in Fig. 12(e), \( G_M \) makes an incorrect guessing of the tape symbol.

Theorem 6.1. It is undecidable whether or not (1) \( \text{dpic}(L_1) \cap \text{dpic}(L_2) = \emptyset \), and (2) \( \text{bpic}(L_1) \cap \text{bpic}(L_2) = \emptyset \), for a 1-retreat-bounded regular \( \pi \)-language \( L_1 \) and a 2-retreat-bounded regular \( \pi \)-language \( L_2 \).

Proof. Given a Turing machine \( M \), a 1-retreat-bounded regular \( \pi \)-language \( R_M \) can be constructed from \( M \) as discussed earlier in this section. Let \( L_1 = R_M \). Let \( \tilde{A} = AdAu \), \( \tilde{B} = BdBu \), and \( \tilde{C} = CudCu \). Let \( L_2 = X(YZ)^* W \), where

\[
\begin{align*}
X &= \tilde{C}d, \\
Y &= r^8(A + B)(r^8(A + B))^* r^8\tilde{C}d, \\
Z &= l^4(A + B)(l^8(A + B))^* l^8\tilde{C}d, \\
W &= r^8(A + B)(r^8(A + B))^* r^8C(l^8)^* dl^4Cr^8(A + B).
\end{align*}
\]

\( L_2 \) is clearly a 2-retreat-bounded regular \( \pi \)-language. The reader is advised to observe that the picture shown in Fig. 12(b) can be described by a \( \pi \)-word in \( L_2 \), but no other picture shown in Fig. 12 (including Fig. 12(g)) can be described by a \( \pi \)-word in \( L_2 \). Note that \( M \) halts on the blank tape if and only if there is a picture in \( \text{dpic}(L_1) \) such that every picture component drawn to simulate the reading of a tape symbol of
$M$ is identical to the picture component located immediately above it. $L_2$ describes only the pictures satisfying this property. As $L_1$ correctly simulates the sequence of states of $M$ involved in the computation of $M$, the additional condition imposed by $L_2$ guarantees that $M$ halts on the blank tape if and only if $\text{dpic}(L_1) \cap \text{dpic}(L_2) \neq \emptyset$ ($\text{bpic}(L_1) \cap \text{bpic}(L_2) \neq \emptyset$).

**Theorem 6.2.** The drawn and basic versions of the ambiguity problem for 2-retreat-bounded regular picture languages are undecidable.

**Proof.** The 1-retreat-bounded regular $\pi$-language $L_1$ and the 2-retreat-bounded regular $\pi$-language $L_2$ defined in the proof of Theorem 6.1 are unambiguous. Let $L = L_1 \cup L_2$. Then, $\text{dpic}(L_1) \cap \text{dpic}(L_2) \neq \emptyset$ ($\text{bpic}(L_1) \cap \text{bpic}(L_2) = \emptyset$, respectively) if and only if $L$ is picture-ambiguous.

**Theorem 6.3.** It is undecidable whether or not (1) $\text{dpic}(L_1) \subseteq \text{dpic}(L_2)$, and (2) $\text{bpic}(L_1) \subseteq \text{bpic}(L_2)$, for a 1-retreat-bounded regular $\pi$-language $L_1$ and a 2-retreat-bounded regular $\pi$-language $L_2$.

**Proof.** As in the proof of Theorem 6.1, let $L_1 = R_M$, where $R_M$ is a 1-retreat-bounded regular $\pi$-language constructed from a Turing machine $M$. We construct a 2-retreat-bounded regular $\pi$-language $L_2$ such that $M$ does not halt on the blank tape if and only if $\text{dpic}(L_1) \subseteq \text{dpic}(L_2)$ if and only if $\text{bpic}(L_1) \subseteq \text{bpic}(L_2)$. The idea is similar to the construction of $L_2$ in the proof of Theorem 6.1. Here, $L_2$ will only describe the pictures containing at least one picture component corresponding to the reading of a tape symbol which is different from the picture component located immediately above it.

Let $A = AuBd + AuCd$, $B = BuAd + BuCd$, and $C = CuAd + CuBd$. Let $h(a) = A$, $h(b) = B$, $h(a) = A$, and $h(b) = B$. Let $G_M = (Q \cup \tilde{Q}, \pi, P', s)$ be the right-linear grammar such that $Q$ is the set of states of $M$, $\tilde{Q} = \{\tilde{q} | q \in Q\}$, $s \in Q$ is the initial state of $M$, and $P'$ consists of the following productions $(p, q \in Q$ and $d, d' \in \{a, b\})$:

$$s \rightarrow C_{dr}^{1,2}C_{dr}^{1}h(d)qA^{p}$$ if $\delta(s, c) = (p, d, L)$,

$$p \rightarrow C_{dr}C_{dr}^{1}h(d)qA^{p}$$ if $\delta(p, c) = (q, d, R)$,

$$\tilde{p} \rightarrow C_{dr}C_{dr}C_{dr}C_{dr}h(d)qA^{\tilde{p}}$$ if $\delta(p, c) = (q, d, R)$,

$$p \rightarrow r_{h}^{1}h(d)qA^{p}$$ if $\delta(p, c) = (q, d, R)$,

$$\tilde{p} \rightarrow r_{h}^{1}h(d)qA^{\tilde{p}}$$ if $\delta(p, c) = (q, d, R)$,

$$p \rightarrow r_{h}^{1}h(d)qA^{p}$$ if $\delta(p, c) = (q, d, R)$,

$$\tilde{p} \rightarrow r_{h}^{1}h(d)qA^{\tilde{p}}$$ if $\delta(p, c) = (q, d, R)$,

$$p \rightarrow r_{h}^{1}h(d)qA^{p}$$ if $\delta(p, c) = (q, d, R)$,

$$\tilde{p} \rightarrow r_{h}^{1}h(d)qA^{\tilde{p}}$$ if $\delta(p, c) = (q, d, R)$,

$$p \rightarrow h(d)qA^{p}$$ if $p$ is an accepting state of $M$. 
Clearly, $L_2 = L(G'_M)$ is a 2-retreat-bounded regular $\pi$-language. It is not difficult to see that $M$ does not halt on the blank tape if and only if dpic($L_1$) $\subseteq$ dpic($L_2$) if and only if bpic($L_1$) $\subseteq$ bpic($L_2$).

**Theorem 6.4.** The drawn and basic versions of the equivalence problem for 2-retreat-bounded regular picture languages are undecidable.

**Proof.** Let $L_1$ and $L_2$ be the $\pi$-languages defined in the proof of Theorem 6.3 and let $L = L_1 \cup L_2$. Then, dpic($L_1$) $\subseteq$ dpic($L_2$) if and only if dpic($L$) = dpic($L_2$). The basic-version case is similar. □

**References**