**Prof. Dr. Jürgen Dassow**
**Otto-von-Guericke-Universität Magdeburg**
**Fakultät für Informatik**

# G R A M M A T I C A L P I C T U R E G E N E R A T I O N

**Manuscript**

Magdeburg, April 2011 – July 2011

# Contents

**Theorem 2.23** *The equivalence problem*

> *Given two phrase structure grammars $G_1 = (N_1, \pi, P_1, S_1)$ and $G_2 = (N_2, \pi, P_2, S_2)$ decide whether or not $bccp(G_1) = bccp(G_2)$ holds?*

*is undecidable for regular grammars.* □

## 2.3.2 Decidability of Properties Related to Subpictures

The membership problem (for word languages) has two variants with respect to subwords: given a grammar $G$ and a word $w$, decide whether or not $w$ is a subword of at least one word (or of all words) in $L(G)$. Both variants are decidable for context-free grammars. This can be seen as follows.

We construct a context-free grammar $G'$ with $L(G') = sub(L(G))$ (see Theorem 1.16) and decide $w \in L(G')$. Since membership is decidable for context-free grammars, the first variant is a decidable problem for context-free grammars.

The set $K = \pi^*\{w\}\pi^*$ of all words with subword $w$ is regular (see Theorem 1.15). By Theorem 1.14, the complement $\overline{K}$ consisting of all words which do not have subword $w$ is regular, too. We consider $L(G) \cap \overline{K}$ which is empty if and only if all words of $L(G)$ have the subword $w$. Since $L(G) \cap \overline{K}$ is context-free (see Theorem 1.12) and the emptiness is decidable for context-free languages, the second variant is decidable for context-free grammars, too.

We now study these two variants of the membership problem for picture grammars.

**Definition 2.24** *We say that the basic chain code picture $p$ is a* <u>*subpicture*</u> *of the basic chain code picture $q$ if there is a chain code picture $p'$ such that $\overline{p'} \equiv p$ and $p' \subseteq q$.*
*We say that the basic chain code picture $p$ is a* <u>*subpicture*</u> *of the basic chain code picture language $L$, if $p$ is a subpicture of some $q \in L$.*

**Theorem 2.25** *i) For an arbitrary basic chain code picture $p$ and an arbitrary context-free grammar $G = (N, \pi, P, S)$, it is decidable whether or not $p$ is a subpicture of $bccp(G)$.*
*i) For an arbitrary chain code picture $p$ and an arbitrary monotone grammar $G = (N, \pi, P, S)$, it is undecidable whether or not $p$ is a subpicture of $bccp(G)$.* □

We omit the proof of Theorem 2.25 which can be given by modifications of the proofs of Theorem 2.14 (consider sets of descriptions of subpictures instead of pictures).

**Definition 2.26** *We say that the basic chain code picture $p$ is a* <u>*universal subpicture*</u> *of the basic chain code picture language $L$, if $p$ is a subpicture of any $q \in L$.*

**Theorem 2.27** *For an arbitrary basic chain code picture $p$ and an arbitrary regular grammar $G = (N, \pi, P, S)$, it is undecidable whether or not $p$ is a universal subpicture of $bccp(G)$.*

*Proof.* We shall present a reduction of the universal subpicture problem to the (undecidable) emptiness problem for monotone grammars. By Theorem 1.22, the languages generated by monotone grammars can be accepted by linearly bounded automata. Thus the emptiness problem for linearly bounded automata (decide whether or not the language accepted by a given linearly bounded automaton is empty) is undecidable, too.

We shall present a linearly bounded automaton in the following normal form: There are two markers b at the tape which are written in the cells before and after the word. The automaton scans in the beginning the left endmarker. Then it scans and rewrites the complete input from left to right until it reaches the right endmarker. Then it performs a stationary step and scans and rewrites the word at the tape from right to left until it reaches the left endmarker. It performs a stationary step, again. This procedure is iterated until the automaton reaches an accepting state which is only possible if the automaton scans the right endmarker. It is easy to see that any linearly bounded automaton $\mathcal{M}$ can be transformed in a linearly bounded automaton $\mathcal{M}'$ in this normal form such that $T(\mathcal{M}') = T(\mathcal{M})$.

A run of a linear bounded automaton in normal form can be written in a rectangle where each row corresponds to a move from the left marker to the right marker or to a move from the right marker to the left marker. In the first row the input (with the markers) is written. Therefore the width of the rectangle is given by the length of the input word increased by 2. The height of the rectangle depends on the number of moves along the word on the tape. An example for such a description is given in Figure 2.9.

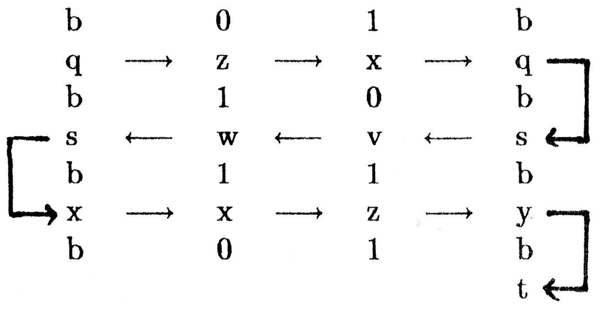


Figure 2.9: Illustration of a run of a linearly bounded with input symbols 0 and 1, marker b and states q,z,x,s,v,w,z,t, where q is the initial state and t is the accepting state

Now we consider the homomorphism $h$ which encodes the input symbols and the marker b by a word of $\{0,1\}^n$ for some $n$. We choose $h$ in such a way that $h(x) = x_1x_2\dots x_n$ and $h(y) = y_1y_2\dots y_n$ contain positions $i$ and $j$ such that $x_i = 0$, $y_i = 1$, $x_j = 1$ and $y_j = 0$. Moreover we consider a linearly bounded automaton $\mathcal{M}''$ which accepts the language $uh(L(\mathcal{M}))u$ where $u$ is the image of the marker. It is easy to construct $\mathcal{M}'' = ((\{0,1\}, \{0,1\}, *, Z, z_0, F, F, \delta)$ from $\mathcal{M}'$. $u$ is used as a "software endmarker" in $\mathcal{M}''$. Obviously, $uh(L(\mathcal{M}))u$ is empty if and only $L(\mathcal{M})$ is empty. Figure 2.10 illustrates this transformations for the run given in Figure 2.9.

Now we associated with $\mathcal{M}''$ the regular grammar grammar $G = (Z, \pi, P, z_0)$ where $P$ consists of all rules of the forms

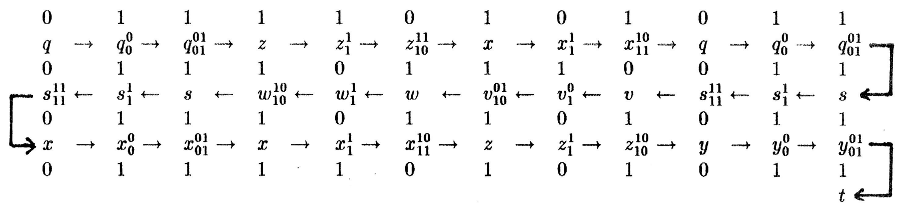


Figure 2.10: Illustration of a transformed run where we use $h(\mathrm{b}) = 011$, $h(0) = 110$ and $h(1) = 101$

$$\begin{array}{ll}
q \to \lambda & \text{for } q \in F, \\
q \to \overrightarrow{read}(a)\overrightarrow{write}(b)\ q' & \text{for } (q', b, R) \in \delta(q, a) \\
q \to \overrightarrow{read}(a)\overrightarrow{write}(b) right-to-left\ q' & \text{for } (q', b, N) \in \delta(q, a) \\
q \to \overleftarrow{read}(a)\overleftarrow{write}(b)\ q' & \text{for } (q', b, L) \in \delta(q, a) \\
q \to \overleftarrow{read}(a)\overleftarrow{write}(b) right-to-left\ q' & \text{for } (q', b, N) \in \delta(q, a)
\end{array}$$

where

$$\overrightarrow{read}(0) = r,\ \overrightarrow{read}(1) = urd,\ \overleftarrow{read}(0) = l,\ \overleftarrow{read}(1) = uld, \tag{2.13}$$

$$\overrightarrow{write}(0) = dru,\ \overrightarrow{write}(1) = r,\ \overleftarrow{write}(0) = dlu,\ \overleftarrow{write}(1) = l, \tag{2.14}$$

$$right-to-left = rddl,\ left-to-right = lddr.$$

The pictures corresponding to these words are given in Figure 2.11. Since the nonterminals

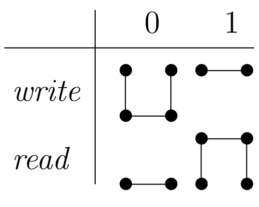


Figure 2.11: The pictures to the words of (2.13) and (2.14)

correspond to the states we draw a picture according to a run of machine where we draw for each direction of the move first the read letter and then the written letter. For the run given in Figure 2.10 we get the picture given in Figure 2.12.

If we move backwards in the next scan of the word we have to read that letter which was written the scan before. The possible pictures which can occur in a "column" are given in Figure 2.13. Obviously, if a run is correctly simulated by the grammar, then the picture $p_{01}$ does not occur as a subpicture. Hence we get that $\mathcal{M}''$ is empty if and only $p_{01}$ is a universal subpicture of $bccp(G)$. Therefore the decidability of the universal subpicture problems implies the decidability of the finiteness problem for monotone grammars. Since the latter problem is undecidable, we have shown the statement of this theorem. □

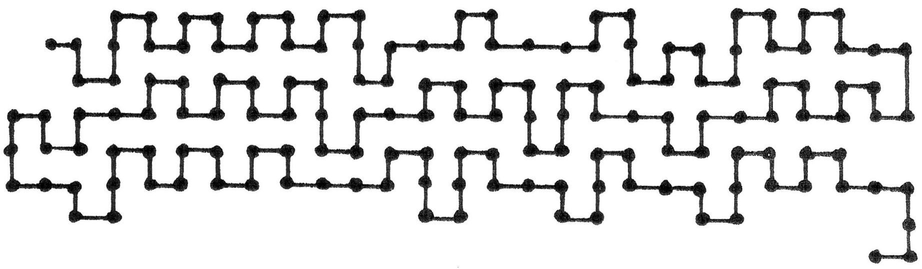


Figure 2.12: The picture corresponding to the run of Figure 2.10

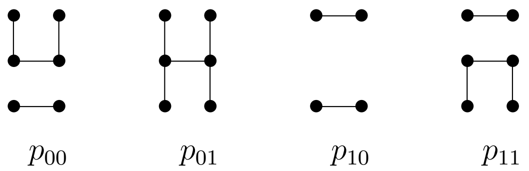


Figure 2.13: The pictures that can occur in a column of a rectangle describing a run

### 2.3.3 Decidability of "Geometric" Properties

We now use the results of the preceding section to get some facts on the decidability of some properties which are of geometric or graph-theoretical origin. We start with the definition of the properties we are interested in.

**Definition 2.28** *A chain code picture p is a simple curve, if all its nodes have a degree at most 2.*

*A chain code picture p is a closed simple curve, if all its nodes have degree 2.*

*A chain code picture p is a tree, if it does not contain a closed simple curve as a subpicture.*

*A chain code picture p is called regular, if all nodes of p have the same degree.*

*A chain code picture p is called Eulerian, if*
*— all nodes of p have an even degree or*
*— there are two nodes n and $n'$ in p such that all nodes of p different from n and $n'$ have even degree.*

*A chain code picture p is called Hamiltonian, if it contains a subpicture p which is a simple curve and contains all nodes of p.*

*A chain code picture p is called convex if there is a chain code 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.*

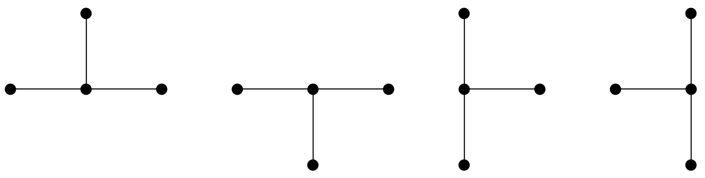
With respect to the definition of convexity we had to change the usual geometric definition to ensure that it fits to chain code pictures. The usual definition requires that a straight line connecting two arbitrary points of the curve does not intersect the curve in a point different from the two given points. Since lines of a chain code picture are parallel to the axes of the system, we restrict the lines connecting the points to such lines, too.

We also note that the definition of an Eulerian curve is usually given in another way which is dual to that of a Hamiltonian curve. A curve is called Eulerian if it can be drawn in such a way that each unit line is only drawn once. It is well known from graph theory (and easy to prove) that our definition is equivalent to the usual one.

Furthermore, we mention that regularity is almost the same as closed simplicity. More precisely, the following assertion holds: *A curve is regular if and only if it is a simple closed curve or it is a curve consisting of a single unit line.* This can be seen as follows: If that the degree of any point is 1, then the curve has to be a single unit line. If the degree of any point is 2, then it is a simple closed curve by definition. Now assume that the curve contains a point of degree $k \geq 3$. Then we consider the most right-upper point $(m, n)$ of the curve. Clearly, $(m, n)$ has a degree $\leq 2$ since $(m+1, n)$ and $(m, n+1)$ are not points of the curve (otherwise $(m, n)$ would not be the most right-upper point). Therefore we have two points with different degree which proves that the curve is not regular.

**Theorem 2.29** *Given a regular grammar $G = (N, \pi, P, S)$, it is undecidable whether or not bccp$(G)$ contains*
*a) a simple curve,*
*b) a closed simple curve,*
*c) a Eulerian picture,*
*d) a tree*
*e) a Hamiltonian picture,*
*f) a regular picture.*

*Proof.* a) In the proof of Theorem 2.27 we have constructed a grammar $G$ such that it is undecidable whether or not $p_{01}$ (see Figure 2.13) is a universal subpicture of *bccp*$(G)$. Obviously, a curve is a simple curve if and only if it contains none of the pictures



as a subpicture. Thus by the proof of Theorem 2.27 *bccp*$(G)$ contains a simple curve if and only if $p_{01}$ is not a universal subpicture of *bccp*$(G)$. Since the latter property is undecidable, the existence of a simple curve is also undecidable.

b) We add to the pictures generated by the grammar $G$ considered in the proof of Theorem 2.27 a tail which connects the end point with with the start point. This can be done by using

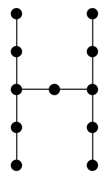
$$q \to U,\ U \to lU,\ U \to uU',\ U' \to uU',\ U' \to rrr$$

for $q \in F$ instead of $q \to \lambda$. Now it is easy to see that the modified grammar generates a closed simple curve if and only if the original grammar does not have the universal subpicture $p_{01}$. As above this gives the undecidability of the existence of a closed simple curve.

c) By definition of $G$ given in the proof of Theorem 2.27, the generated picture contains a start and end point which both have the degree 1. Furthermore, by construction, *bccp*$(G)$ contains no points of degree 4. Thus a Eulerian picture of *bccp*$(G)$ contains besides the start and end point only nodes of degree 2. This means that any Eulerian curve in *bccp*$(G)$ is a simple curve. Therefore *bccp*$(G)$ contains a Eulerian curve if and only if it contains a simple curve. Now the statement follows from a).

d) follows from b).

e) We consider the grammar $G'$ obtained from the grammar $G$ given in the proof of Theorem 2.27 by replacing any occurrence of a letter $b \in \pi$ by $bb$. Then it is undecidable whether or not the picture $p'_{01}$



is a universal subpicture. Moreover, it is easy to see that a picture of *bccp*$(G')$ is Hamiltonian if and only if it does not contain the picture $p'_{01}$ (the strechted versions of the other pictures given in Figure 2.13) do not destroy Hamiltonicity). Thus the existence of a Hamiltonian picture is equivalent to the non-existence of $p'_{01}$ as a universal subpicture. Obviously, the latter property is undecidable, too, and therefore the existence of a Hamiltonian curve is also undecidable.

f) By the fact that each picture generated by the grammar $G$ of the proof of Theorem 2.27 contains at least two unit lines and the assertion given before the Theorem, the statement follows from b). $\square$

We now turn to the case that a property is required for all pictures.

**Lemma 2.30** *Let $G$ be a regular grammar such that all elements of bccp$(G)$ are closed simple curves. Then bccp$(G)$ is finite.*

*Proof.* Let $Z$ be the set of words $b_1b_2b_3b_4$ such that *bccp*$(b_1b_2b_3b_4)$ is a unit square.

Since $G$ is regular, $L(G)$ can be represented according to Theorem 1.15. By $r(L)$ we denote the number of operation which is necessary to get $L$. We now prove by induction on the number $r$ of operations union, product and Kleene closure $*$ the statement.

It is easy to see that all sets – with exception of the sets $\{z\}$, $z \in Z$ – obtained by 3 or less operation do not only consist of simple closed curves. For the sets $\{z\}$, $z \in Z$, the statement holds by definition and thus the induction basis is shown.

Let $r \geq 4$ and $r = r(L)$. If $L = L_1 \cup L_2$, then $r(L_1) < r$ and $r(L_2) < r$ and $L_1$ and $L_2$ contain only closed simple curves. By induction hypothesis, *bccp*$(L_1)$ and *bccp*$(L_2)$ are finite. Consequently, *bccp*$(L) =$ *bccp*$(L_1) \cup$ *bccp*$(L_2)$ is also finite.

Let $L = (L')^*$. We consider some $w \in L'$. Since $w \in L' \subseteq L$, *bccp*$(w)$ is a closed simple curve. If the start point and endpoint of *dccp*$(w)$ are different, then it is easy to see that *bccp*$(ww)$ is not simple closed curve since it contains a point of degree at least 3. However, $ww \in (L')^2 \subseteq L$ and therefore *bccp*$(ww)$ has to be a closed simple curve. But if the start and endpoint coincide, then *bccp*$(w) =$ *bccp*$(w^2) =$ *bccp*$(w^3) = \dots$. Moreover, if $v \in L'$ and $v \neq w$, then *bccp*$(wv)$ is not a simple closed curve since it also contains a point of degree at least 3. Thus $L'$ consists of a single word, and $L$ has only one non-empty picture. Hence $L$ is finite.

Now let $L = L_1L_2$. If $L_1 = L'_1 \cup L'_2$, then $L = L'_1L_2 \cup L'_2L_2$ and the statement follows as in the case of union. Let $L_1 = (K')^*$. If $w_1 \in K'$ and the start point and the end point of *dccp*$(w_1)$ are different, then *bccp*$(w_1w_2v_2)$ for $v_2 \in L_2$ is not a simple curve. We continue as in the case of Kleene closure above and get that $K'$ contains only one non-empty word. By the same considerations for $L_2 = L_1'' \cup L_2''$ and $L_2 = (K'')^*$, we get that *bccp*$(L)$ is finite or *bccp*$(L) =$ *bccp*$(u_1u_2)$ where $u_1 \in K'$ and $u_2 \in K''$. Clearly, *bccp*$(L)$ is finite in all these cases. It remains the case that $L = L_1'L_2'L_1''L_2''$ because $L_1$ and $L_2$ are products. Then we continue as above and get that *bccp*$(L)$ is a product of $r$ single letter languages, which also implies the finiteness of *bccp*$(L)$. □

**Theorem 2.31** *For an arbitrary regular grammar $G = (N, \pi, P, S)$, it is decidable whether or not*

*a) all pictures of bccp$(G)$ are closed simple curves,*
*b) all pictures of bccp$(G)$ are rectangles,*
*c) bccp$(G)$ contains a rectangle,*
*d) bccp$(G)$ contains a convex picture.*

*Proof.* a) Let $G$ be given. We first decide whether or not *bccp*$(G)$ is finite. If the answer is no, then *bccp*$(G)$ contains a picture, which is not a simple closed curve by Lemma 2.30. If the answer is yes, then we get a number $r$ such that all pictures of *bccp*$(G)$ are contained in a circle with the radius $r$ and the centre $(0, 0)$. We now check any picture which is contained in this circle and is not a closed simple curve whether or not it is generated by $G$. If the answer in all cases is negative, *bccp*$(G)$ contains only closed simple curves; otherwise not.

b) The proof is the same as in a) with the only difference that we test all non-rectangles in the circle.

c) and d) We omit the proofs and refer to [2]. □

## 2.3.4 Stripe Languages

We have seen in the preceding sections that most of the interesting problems are undecidable or at least it is very hard to decide them. Thus one is interested in special cases where the problems can be decided (easily). We now present such a case.

**Definition 2.32** *A picture languages $L$ is called a stripe picture language, if there are real numbers $k$, $d_1$ and $d_2$ such that, for any picture $p \in L$ and any point $(m, n) \in V(p)$,*

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

*We then also say that $L$ is a $(k, d_1, d_2)$-stripe language.*

By definition, a stripe is given by two parallel lines, and all points of all pictures of the picture language can be placed between the two lines. An example is given in Figure 2.14.

**Lemma 2.33** *Let $G = (N, \pi, P, S)$ be a reduced*[1] *context-free grammar such that dccp$(G)$ is a $(k, d_1, d_2)$-stripe picture. For any non-empty word $x \in \pi$ such that $A \Longrightarrow^* xAy$ or $A \Longrightarrow^* yAx$ for some $A \in N$, $sh(x) = (0, 0)$ or $sh(x) = (m, km)$ for some $m$.*

[1]A context-free grammar is called reduced iff, for any nonterminal $A$, there are derivations $S \Longrightarrow^* xAy$ and $A \Longrightarrow^* w$ for some $u, v \in (N \cup T)^*$ and $w \in T^*$, i.e., none of the nonterminals can block the derivation and none of the nonterminals is superflous.

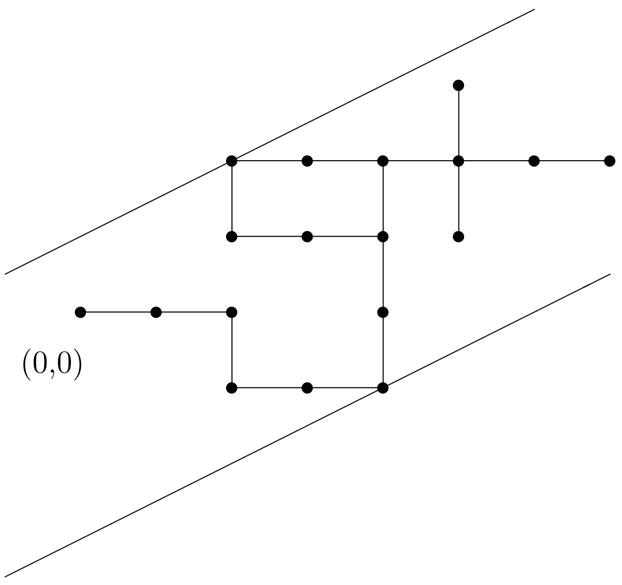


Figure 2.14: Example of a stripe language (with $k = 1/2, d_1 = 1$ and $d_2 = -4$)

*Proof.* Let us assume that $A \Longrightarrow^* xAy$ with $sh(x) = (m, n)$. By our assumption, for any $i \geq 0$, we have the derivations

$$D_i : S \Longrightarrow^* zAz' \Longrightarrow^* zxAyz' \Longrightarrow zx^2Ay^2z' \Longrightarrow^* \ldots \Longrightarrow^* zx^iAy^iz' \Longrightarrow^* zx^iwy^iz' = w_i \in T^*$$

in $G$. If $sh(z) = (m', n')$, then $sh(zx^i) = (m' + im, n' + in)$. Since $dccp(G)$ is a $(k, d_1, d_2)$-stripe language, we have

$$d_1 \leq n' + in - k(m' + im) \leq d_2$$

or equivalently,

$$d_1 \leq n' - km' + i(n - km) \leq d_2 \tag{2.15}$$

for any $i \geq 0$. Therefore $n - km = 0$ holds. Thus $n = m = 0$ or $n = km$, which proves our statement. $\square$

**Corollary 2.34** *Any $(k, d_1, d_2)$-stripe picture language with a non-rational $k$ is finite.*

*Proof.* Assume that $k$ is non-rational. By Lemma 2.33 we only have derivations $A \Longrightarrow^* xAy$ with $sh(x) = sh(y) = (0, 0)$. This means that the grammar is normal, which implies the finiteness of the language (see Corollary 2.20). $\square$

The essential key for a better situation for stripe languages with respect to decision problems is the following lemma which reduces the problem for picture languages to that of word languages.

**Lemma 2.35** *Let $k$ be a rational number and $d_1$ and $d_2$ real numbers with $d_1 < d_2$. Then there is an alphabet $V$ and an encoding $\mu$ which maps any picture in the stripe to a word over $V$ such that the following properties hold:*

*i) For two $(k, d_1, d_2)$-stripe pictures $q$ and $q'$, $\mu(q) = \mu(q')$ if and only if $q = q'$.*

*ii) For a $(k, d_1, d_2)$-stripe picture $q$, $\mu(q)$ can be computed in linear time (in the size of $q$).*

*iii) If $L$ is a regular $(k, d_1, d_2)$-stripe picture language, then $\mu(L) = \{\mu(q) \mid q \in L\}$ is a regular word language over $V$, which can effectively be constructed.* □

We do not prove Lemma 2.35. We only give one mapping $\mu$ which satisfies the requirement of Lemma 2.35. We divide a given stripe picture into slices of a given width $d \geq 0$ where $d$ is a natural number. Obviously, a slice is bounded by the lines $y = kx + d_1$, $y = kx + d_2$ $x = id + j$ and $x = (i+1)d + j$ for some $i$ and $j$, $0 \leq j \leq d - 1$. The point and lines on the left bounding line belong to the slice (those of the right bound belong to the right neighbour slice. Hence a slice is a finite part of the plain. Thus only a finite number of pictures fits into a slice. All these pictures form a finite set which will be the alphabet $V$. The mapping $\mu$ maps any slice onto the picture contained in the slice. Since a picture $p$ can be described as a sequence of slices, $\mu(p)$ is a sequence of elements of $V$, i.e., a word over $V$.

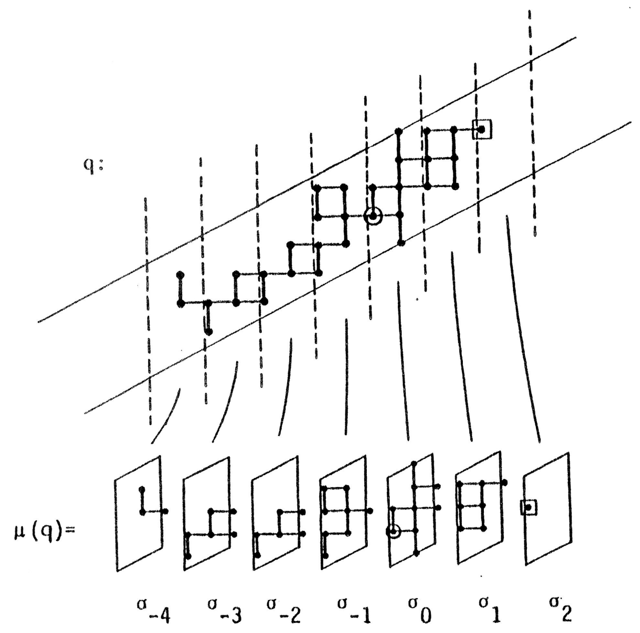


Figure 2.15: Slicing of a picture and the associated mapping *mu*

We are now in the position that some problems, which are very hard for arbitrary regular picture languages, have a low complexity for regular stripe picture languages or that some problems, which are undecidable for arbitrary regular picture languages, are decidable for regular stripe picture languages.

**Theorem 2.36** *For a regular grammar $G$ such that $bccp(G)$ is a stripe picture language and a picture $p$, it is decidable in linear time whether or not $p \in bccp(G)$.*

*Proof.* Given a regular grammar $G$ such that $bccp(G)$ is a stripe language, we construct the regular grammar $G'$ generating $\mu(bccp(G))$. For a given picture $p$ of size $t$ we construct in $O(t)$ the word $\mu(p)$. Obviously, by the definition of $\mu$, $|\mu(p)| = O(t)$. Therefore we can decide $\mu(p) \in \mu(bccp(G))$ in time $O(t)$. Since $p \in bccp(G)$ if and only if $\mu(p) \in \mu(bccp(G))$, the statement is proved. $\square$

**Theorem 2.37** *For two regular grammars $G_1$ and $G_2$ such that $bccp(G_1)$ and $bccp(G_2)$ are stripe picture languages, it is decidable whether or not $bccp(G_1) = bccp(G_2)$.*

*Proof.* We construct the regular grammar $G_1'$ and $G_2'$ generating $\mu(bccp(G_1))$ and $\mu(bccp(G_2))$, respectively. Obviously, $bccp(G_1) = bccp(G_2)$ if and only if $L(G_1') = L(G_2')$. Since there is a deciding procedure for the latter equality (see Theorem1.24) we can decide whether or not $bccp(G_1) = bccp(G_2)$. $\square$

**Theorem 2.38** *For a regular grammar $G$ such that $bccp(G)$ is a stripe picture language, it is decidable whether or not $bccp(G)$ contains*
*a) a simple curve,*
*b) a closed simple curve,*
*c) a regular curve,*
*d) an Eulerian curve.*

*Proof.* a) Let $G$ be a regular grammar. Let $h$ be the homomorphisms with $h(b) = b^2$ for $b \in \pi$. Then we construct the grammar $G'$ with $L(G') = h(L(G))$. We now choose an even $d$ and slice the picture in such a way that the bounding vertical lines are of the form $x = id+1$. By this choice it is obvious that the points on the bounding vertical lines have degree 2 since the are middle point of subpictures corresponding to $rr$ or $ll$ (and there are no lines of the picture which are a part of a bounding vertical line). Now it is obvious that a picture of $bccp(G)$ is a simple closed curve if and only if all – with exception of perhaps exactly two points of degree 1 – points of the picture have the degree 2. Now let

- $V_1$ be the set of all slices such that its points which are not on a bounding line have degree 2 with exception of two points with degree 1,
- $V_1'$ be the set of all slices such that its points which are not on a bounding line have degree 2 with exception of one point with degree 1,
- $V_2$ be the set of all slices such that its points which are not on a bounding line have degree 2.

Now it is obvious that $p$ is a simple curve if and only if

$$\mu(p) \in K = (V_2)^* V_1 (V_2)^* \cup (V_2)^* V_1' (V_2)^* V_1' (V_2)^* \cup (V_2)^* .$$

Thus in order to check whether $bccp(G)$ contains a simple curve we have only to check whether $\mu(L(G)) \cap K$ is non-empty. Since $K$ is regular (see Theorem 1.15), the set of regular word languages is closed under intersection (see Theorem 1.12) and the decidability of the emptiness problem for regular word languages (see Theorem 1.24), the existence of a simple curve in $bccp(G)$ can be decided.

b) We can give the same proof; we have only to choose $K = (V_2)^*$.

c) By the remarks after Definition 2.28, a picture is a regular curve if and only if it is a unit line or a closed simple curve. Thus we first check whether *bccp*$(G)$ contains one of the four unit lines (which can be done by Theorem 2.14 i)). In the affirmative case *bccp*$(G)$ contains a regular curve. In the negative case we check whether *bccp*$(G)$ contains a simple closed curve by b).

d) We can give a proof analogous to that of a) taking

- $V_1$ be the set of all slices such that its points which are not on a bounding line have even degree with exception of two points with odd degree,
- $V_1'$ be the set of all slices such that its points which are not on a bounding line have even degree with exception of one point with odd degree,
- $V_2$ be the set of all slices such that its points which are not on a bounding line have even degree 2.

□

All our positive decision results given above have the supposition that the regular grammar generates a stripe language. Thus the results are only useful if we can decide whether or a given grammar generates a stripe language. We shall prove that this is decidable for context-free grammars. For this purpose we need the following lemma.

**Lemma 2.39** *Let $G = (N, \pi, P, S)$ be a context-free grammar in the normal form of Theorem 1.6 iii) such that $L(G) = pref(L(G))$. If there is a real number $k$ such that, for all words $x$ in the set*

$$T = \{x \mid x \in \pi^*,\ A \Longrightarrow^* xAy \text{ or } A \Longrightarrow^* yAx \text{ for some } y \in \pi^* \text{ and } A \in N\} \qquad (2.16)$$

*$sh(x) = (m, mk)$ for some $m$, then there are two real numbers $d_1$ and $d_2$ such that $dccp(G)$ is a $(k, d_1, d_2)$-stripe language.*

*Proof.* Since $L(G) = pref(L(G))$ we have only to show, that there are real numbers $d_1$ and $d_2$ such that, for any $w \in L(G)$ with $sh(w) = (m, n)$,

$$km + d_1 \leq n \leq km + d_2. \qquad (2.17)$$

Let $w$ be a word of $L(G)$ which is longer than the constant $c$ of the pumping lemma (see Theorem 1.7). Then there is a decomposition $w = z_1xz_2yz_3$ with $xy \neq \lambda$, $z_1z_2z_3 \in L(G)$ and there is a derivation

$$S \Longrightarrow^* z_1Az_3 \Longrightarrow^* z_1xAyz_3 \Longrightarrow^* z_1xz_2yz_3$$

for some $A \in N$ (the last property holds by the proof of the pumping lemma). Let $sh(w) = (m, n)$ and $sh(z_1z_2z_3) = (m', n')$. Then we get $sh(xy) = (m - m', n - n')$. By assumption $sh(x) = (a, ka)$ and $sh(y) = (b, kb)$ for some $a$ and $b$. Then $(m - m', n - n') = sh(xy) = sh(x) + sh(y) = (a + b, k(a + b))$. Hence $n - n' = k(m - m') = km - km'$, from which $n - km = n' - km'$ follows. Consequently, $km + d_1 \leq n \leq km + d_2$ if and only if $km' + d_1 \leq n' \leq km' + d_2$. Therefore the relation (2.17) holds for $w$ if and only it holds for the shorter word $z_1z_2z_3$, too. Therefore we have to find $d_1$ and $d_2$ such that (2.17) holds for all words of length less than $c$.

If $z \in L(G)$, $|z| = r$ and $sh(z) = (s,t)$, then $-r \leq s \leq r$ and $-r \leq t \leq r$ since any letter of $z$ contributes at most 1 to $s$ or $t$. We now choose $d_2 = c + kc$. Let $z$ be a word of length $r \leq c$ and with $sh(z) = (m,n)$. Then we have $n - km \leq r + kr \leq c + kc = d_2$. Therefore $n \leq km + d_2$. Analogously, we can show that $d_1 = -d_2$ satisfies $km + d_1 \leq n$.

Thus (2.17) holds for any word $z \in L(G)$ of length $\leq c$ and therefore for all words $w \in L(G)$. □

**Theorem 2.40** *For a context-free grammar $G$, it is decidable whether or not $bccp(G)$ is a stripe language.*

*Proof.* Let $G = (N, \pi, P, S)$ be given. Obviously, $L(G)$ is a stripe language if and only if $pref(L(G))$ is a stripe languages because the intermediate points of the drawing have to be in the stripe, too. Such without loss of generality we can assume that $G$ is in the normal form of Theorem 1.6 iii) and $L(G) = pref(L(G))$. By Lemmas 2.33 and 2.39 (note that $(0,0) = (0, k \cdot 0)$), $L(G)$ is a stripe language if and only if, for all words $x$ in the set $T$ of (2.16), $sh(x) = (m, km)$ for some $k$ which can easily be derived from $T$. Let $k = \frac{p}{q}$ where the greatest common divisor of $p$ and $q$ is 1.

First we note that $T$ is context-free. We consider the context-free grammars

$$G_A = (N, \pi, P \cup \{A \rightarrow \#_A\}, A)$$

with new symbols $\#_A$. It is easy to see

$$T = \bigcup_{A \in N} h(pref(L(G_A) \cap \pi^*\{\#_A\})$$

where $h$ is the homomorphism defined by $h(b) = b$ for $b \in \pi$ and $h(\#_A) = \lambda$ for $A \in N$. Hence $\Psi(T)$ is a semi-linear set.

On the other hand, let $T_1$ be the set of all words $w$ such that $sh(w) = (0,0)$ and $T_2$ the set of all words $w$ such that $sh(w) = (m, mk)$ for some $m$ (or equivalently, $\#_r(w) - \#_l(w) = k(\#_u(w) - \#_d(w))$ or

$$\#_l(w) = \#_r(w) - k(\#_u(w) - \#_d(w)) \tag{2.18}$$

. Then, for $\pi = \{u, d, r, l\}$,

$$\begin{aligned} \Psi(T_1) &= \{\alpha(1,1,0,0) + \beta(0,0,1,1) \mid \alpha, \beta \in \mathbb{N}_0\}, \\ \Psi(T_2) &= \{\alpha(q,0,0,-p) + \beta(0,q,0,p) + \gamma(0,0,1,1) \mid \alpha, \beta, \gamma \in \mathbb{N}_0\}, \end{aligned}$$

(note that $\psi(T_2) = \{\alpha q, \beta q, \gamma, \gamma - \alpha p + \beta p)$ from which $\#_l(w) = \gamma - \alpha p + \beta p = \gamma - (\alpha q - \beta q)k = \#_r(w) - k(\#_u(w) - \#_d(w))$, i.e., (2.18) follows). Thus $T_1 \cup T_2$ is semi-linear, too. Moreover, $G$ generates a stripe languages if and only if $T \subseteq T_1 \cup T_2$ if and only if $\Psi(T) \subseteq \Psi(T_1) \cup \Psi(T_2)$ which is decidable by Theorem 1.25. □

We mention that there are some other approaches to simplify the decision procedures. As an example we give the 3-way pictures introduced by Ch. Kim in [16]. Here we restrict to words over $\{u, d, r\}$, i.e., we cannot move to the left during the drawing process. It is easy to see that curves describing a function which occur very often and in a lot of fields can be presented by a 3-way picture.

## 2.4 Some Generalizations

In this section we shortly mention some possible generalization of the chain code pictures which we have introduced and studied in the preceding sections. All these generalizations are motivated by better drawings of some pictures.

The first type of generalization consist in a larger number of directions. We have to note that the original definition of chain code pictures given by H. Freeman in [7] used eight directions 0,1,2,. . . ,7 given in Figure 2.16. It is very obvious that the description

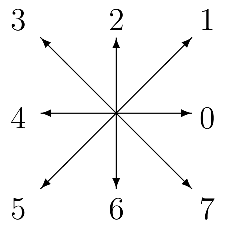


Figure 2.16: Directions used by H. Freeman

of the line going from $(0, 0)$ to $(2, 2)$ can only be described over $\pi$ by a stair, i.e., by one of the words *ruru* or *urur* or *ruur* or *urru*, which does not give the curve we are interested in, whereas the description 11 in the Freeman code gives exactly the line under consideration. Also the part of the parabola $y = x^2$ between the points $(-2, 4)$ and $(2, 4)$ is better described by 67671212 than by *ddrddrruuruu*.

However, with respect to the properties of the language families we get almost the same results if we go from $\pi$ to the Freeman alphabet. With respect to undecidabilities this is clear by the following remarks: The additional letters of the Freeman alphabet correspond to lines which cannot be drawn by words over $\pi$; hence these letters cannot be used in the generation of the pictures generated over $\pi$; thus we can repeat literally the proofs for $\pi$ if we use the same pictures. With respect to decidabilities it is mostly easy to transfer the proof to the Freeman alphabet because the basic properties (as the regularity of the set of all words describing a picture etc.) are also valid.

In order to get coloured pictures we can consider a finite set $C$ of colours and the basic alphabet $\pi \times C$, i.e., the letters are of the form $(b, c)$ with a direction $b \in \pi$ and a colour $c \in C$. The meaning of $(b, c)$ is to draw a unit line in direction $b$ with colour $c$. Here we have again the situation the most of the results presented for $\pi$ hold for $\pi \times C$, too.

Our last extension is made in order to get disconnected figures whereas we know that all pictures $bccp(w)$ with $w \in \pi^*$ are connected. We add two new letters $\uparrow$ and $\downarrow$ to the alphabet such that we get the alphabet

$$\pi_{\updownarrow} = \{u, d, r, l, \uparrow, \downarrow\}\,.$$

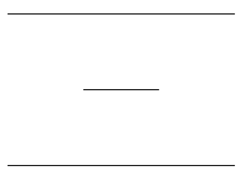
The intuitive meaning of these additional letters is "lift the pen" ($\uparrow$) and "lower the pen" ($\downarrow$). Thus we can have two states: "pen-up" and "pen-down". The new feature is that we can move the pen in the state "pen-up" which is not accompanied by a drawing whereas in the state "pen-down" we have the usual drawing process. Obviously, if we perform two or more times in succession the operation $\uparrow$ we do not change the state "pen-up", and an analogous remark holds for $\downarrow$.

Therefore we associate an extended drawn picture $edccp(w)$ with a word $w$ over $\pi_{\updownarrow}$ in the following inductive way.

- if $w = \lambda$, then $edccp(w) = ((0,0), \emptyset, (0,0), \downarrow)$, and
- if $w = w'b$, $w' \in \pi_{\updownarrow}^*$, $b \in \pi$ and $edccp(w') = ((0,0), p, z, \downarrow)$, then $edccp(w) = ((0,0), p \cup \{z, b(z))\}, b(z), \downarrow)$,
- if $w = w'b$, $w' \in \pi_{\updownarrow}^*$, $b \in \pi$ and $edccp(w') = ((0,0), p, z, \uparrow)$, then $edccp(w) = ((0,0), p, b(z), \uparrow)$,
- if $w = w'b$, $w' \in \pi_{\updownarrow}^*$, $b = \uparrow$, $edccp(w') = ((0,0), p, z, s)$ and $s \in \{\uparrow, \downarrow\}$, then $edccp(w) = ((0,0), p, z, \uparrow)$,
- if $w = w'b$, $w' \in \pi_{\updownarrow}^*$, $b = \downarrow$, $edccp(w') = ((0,0), p, z, s)$ and $s \in \{\uparrow, \downarrow\}$, then $edccp(w) = ((0,0), p, z, \downarrow)$.

The additional fourth component gives the state "pen-up" ($\uparrow$) or "pen-down" ($\downarrow$) and we only draw, i.e., add a line to the picture, if we are in the state "pen-down".

The picture associated with $rrr \uparrow ddd \downarrow lll \uparrow ur \downarrow r$ is



where the origin is the left-upper point of the picture.

The basic picture $ebccp(w)$ associated with a word $w \in \pi_{\updownarrow}^*$ is defined analogously to the definition if $w \in \pi^*$.

For a grammar $G = (N, \pi_{\updownarrow}, P, S)$ we define

$$edccp(G) = \{edccp(w) \mid w \in L(G)\} \text{ and } ebccp(G) = \{ebccp(w) \mid w \in L(G)\},$$

and for a family $X$ of grammars, we set

$$\mathcal{CCP}_{\updownarrow}(X) = \{ebccp(G) \mid G \in X,\ L(G) \subseteq \pi_{\updownarrow}^*\}.$$

Without proof we mention the following relations.

**Theorem 2.41** *i)* $\mathcal{CCP}_{\updownarrow}(REG) \subset \mathcal{CCP}_{\updownarrow}(CF) \subset \mathcal{CCP}_{\updownarrow}(CS) = \mathcal{CCP}_{\updownarrow}(RE)$.
*ii)* $\mathcal{CCP}(X) \subset \mathcal{CCP}_{\updownarrow}(X)$ *for* $X \in \{REG, CF, CS, RE\}$. □

We also mention that with respect to decision problems the situation is more worse for extended chain code picture languages, i.e., we have at least the same statement as in case of usual chain code picture languages or the decidability status changes from decidable to undecidable if we go from (usual) chain code picture languages to extended chain code picture languages. For instance, the membership problem for regular extended chain code picture languages is **NP**-complete, too, however, the membership problem for context-free extended chain code picture languages is already undecidable.

For a detailed information on extended chain code pictures we refer to [13].

## 2.5 Lindenmayer Chain Code Picture Languages and Turtle Grammars

In this section we shall start with chain code picture languages which are generated by some types of Lindenmayer systems. Then we shall give another type of picture generating devices which are – in a certain sense – equivalent to chain code pictures, however, the new type allows changes of the direction (in the drawing process) by angles which are not multiples of $90^0$. This approach is used to draw plants in different phases of their developments. Some further applications to space-filling curves and kolam pictures from India will be given, too. Here we shall not develop a theory, we are more interested to show some applications by examples.

### 2.5.1 Definitions and some Theoretical Considerations

In the preceding section we studied the properties of families of chain code picture languages which are generated by sequential grammars of the Chomsky hierarchy. Obviously, the concepts considered there can be transformed to most devices which generate words over $\pi$ because such words can be interpreted as chain code picture.

With respect to Lindenmayer systems we obtain the following concept. For an ET0L system $G = (v, \pi, P_1, P_2, \ldots, P_n, w)$ we define

$$bccp(G) = \{bccp(w) \mid w \in L(G)\} \text{ and } dccp(G) = \{dccp(w) \mid w \in L(G)\}.$$

**Example 2.42** We consider the D0L system

$$K_1 = (\pi, \pi, \{u \to urul^2url, d \to dldr^2dld, r \to rdru^2rdr, l \to luld^2lul\}, urdl).$$

Since $K_1$ is a deterministic system with only one table, the systems generates in $n$ steps exactly one word $w_n$. The languages $L(K_1)$ consists of all words $w_n$, $n \geq 0$. Note that $w_0 = urdl$. Moreover,

$$w = w_0 \Longrightarrow w_1 = urul^2urlrdru^2rdrdldr^2dldluld^2lul.$$

Since we replace any letter by a word of length 8 and $|w| = 4$, for $n \geq 0$, $|w_n| = 4 \cdot 8^n$. The pictures $bccp(w_n)$ with $n \in \{0, 1, 2, 3\}$ are given in Figure 2.17.

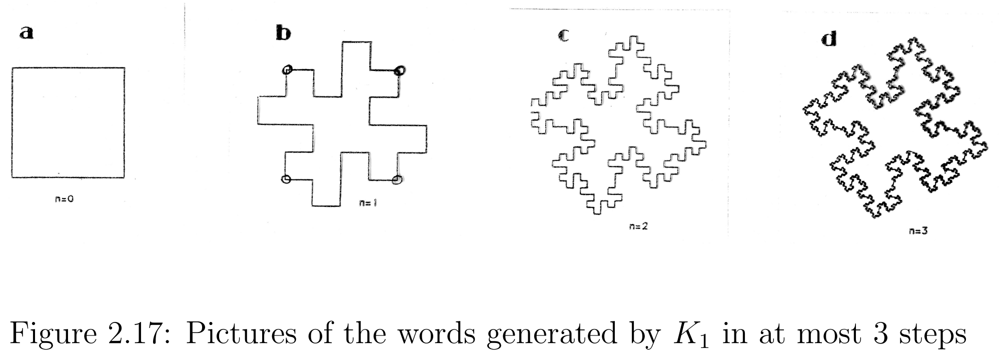


Figure 2.17: Pictures of the words generated by $K_1$ in at most 3 steps

By $\mathcal{CCP}(E0L)$ and $\mathcal{CCP}(ET0L)$ we denote the families of all chain code picture languages which can be generated by E0L and ET0L systems, respectively.

It is very easy to transform some of the inclusions known for word languages to picture languages. As an example we present the following statement.

**Theorem 2.43** $\mathcal{CCP}(CF) \subset \mathcal{CCP}(E0L)$

*Proof.* Let $G$ be a context-free grammar. By Theorem 1.11, $\mathcal{L}(CF) \subset \mathcal{L}(E0L)$ and therefore there is an E0L system $G'$ such that $L(G') = L(G)$. Thus $bccp(G') = bccp(G)$. Hence $\mathcal{CCP}(CF) \subseteq \mathcal{CCP}(E0L)$.

For the language $L = \{rru^{2^n}rr \mid n \geq 0$ we have shown in the proof of Theorem 2.13 that $bccp(L) \notin \mathcal{CCP}(CF)$. On the other hand $L = L(G)$ for the E0L system

$$(\pi, \pi, \{u \to u^2, d \to d, r \to r, l \to l\}, rrurr)$$

which proves $bccp(L) \in \mathcal{CCP}(E0L)$. Hence the inclusion $\mathcal{CCP}(CF) \subseteq \mathcal{CCP}(E0L)$ is strict. □

The change of the direction in the drawing process of a chain code picture is a multiple of $90^0$ (e.g. from $r$ to $u$ it is $90^0$, from $r$ to $d$ it is $-90^0$ and from $r$ to $l$ it is $180^0$). From the point of modelling the development of plant – which is the original idea behind Lindenmayer systems – it is natural to allow other angles, too, because the drawing of a branching in a plant requires angles different from multiples of $90^0$ (usually the branchings do not go perpendicularly to the original direction). Therefore one has introduced turtle grammars.

**Definition 2.44** *A turtle grammar is an* $(n+4)$*-tuple* $G = (V, T, P_1, P_2, \ldots, P_n, w, \alpha_0, \alpha)$, *where*
— $V$ *is an alphabet not containing* $+$ *and* $-$, *and* $T \subseteq V$ *is a subset containing the letter* $F$,
— *for* $1 \leq i \leq n$, $P_i$ *is a finite set of productions of the form* $A \to v$ *with* $A \in V$ *and* $v \in (V \cup \{+, -\})^*$,
— $w \in (V \cup \{+, -\})^*$,
— $\alpha_0$ *and* $\alpha$ *are two angles.*

For $1 \leq i \leq n$, we define $dom(P_i)$ as the set of all letters $A \in V$ such that $P_i$ contains a rule with left hand side $A$ and set

$$P_i' = P_i \cup \{a \to a \mid a \in (V \setminus dom(P_i)) \cup \{+, -\}\}.$$

Then, by definition, $H_G = (V \cup \{+, -\}, T \cup \{+, -\}, P_1', P_2', \ldots, P_n', w)$ is an ET0L system. We call $H_G$ the ET0L system of $G$.

Obviously, all words $z$ with $w \Longrightarrow^* z$ satisfy $z \in (V \cup \{+, -\})^*$. Therefore we now give an interpretation of a word over $(V \cup \{+, -\})$ as an picture.

The letter $F$ is interpreted as a drawing of a line of unit length in the current direction. The remaining letters of $V$ are ignored in the drawing. $+$ and $-$ mean a turn (rotation) by the angle $\alpha$ and $-\alpha$, respectively. (This process models the move of a turtle who first changes the direction by moving the head in the new direction and then a move of the body along the new direction.) The angle $\alpha_0$ gives the start direction. Formally we get the following definition.

**Definition 2.45** *Let $\alpha_0$ and $\alpha$ be two angles and $V$ an alphabet containing the letter $F$ and not containing the letters $+$ and $-$. For a word $w \in (V \cup \{+,-\})^*$, we define inductively a configuration $c(w) = (M,(x,y),\beta)$ with a set $M$ of lines of unit length, a point $(x,y)$ in the plain and an angle as follows*

- $c(\lambda) = (\emptyset,(0,0),\alpha_0)$,
- *if* $c(w) = (M,(x,y),\beta)$, *then*
  - $c(wx) = c(w)$ *for* $x \in V$ *and* $x \neq F$,
  - $c(w+) = (M,(x,y),\beta+\alpha)$ *and* $c(w-) = (M,(x,y),\beta-\alpha)$,
  - $c(wF) = (M \cup \{b\},(x',y'),\beta)$, *where $(x',y')$ is the point such that the distance between $(x',y')$ and $(x,y)$ is 1, $b$ is the line connecting $(x,y)$ and $(x',y')$ and the angle between $b$ and the $x$-axes is $\beta$.*

*The picture $tur(w)$ is defined as the first component of $c(w)$.*

**Definition 2.46** *For a turtle grammar $G$ with associated ET0L system $H_G$ we define the picture language $tur(G)$ generated by $G$ as*

$$tur(G) = \{tur(w) \mid w \in L(H_G)\}.$$

**Example 2.47** We consider the angles $\alpha_0 = 0^0$, $\alpha = 90^0$ and the word $w = +F-F+FF$. We obtain the following sequence of configurations

$$\begin{aligned}
c(\lambda) &= (\emptyset,\ (0,0),\ 0^0),\\
c(+) &= (\emptyset,\ (0,0),\ 90^0),\\
c(+F) &= (\{((0,0),(0,1))\},\ (0,1),\ 90^0),\\
c(+F-) &= (\{((0,0),(0,1))\},\ (0,1),\ 0^0),\\
c(+F-F) &= (\{((0,0),(0,1)),\ ((0,1),(1,1))\},\ (0,1),\ 0^0),\\
c(+F-F+) &= (\{((0,0),(0,1)),\ ((0,1),(1,1))\},\ (0,1),\ 90^0),\\
c(+F-F+F) &= (\{((0,0),(0,1)),\ ((0,1),(1,1)),\ ((1,1),(1,2))\},\ (0,1),\ 90^0),\\
c(+F-F+FF) &= (\{((0,0),(0,1)),\ ((0,1),(1,1)),\ ((1,1),(1,2)),((1,2),(1,3))\},\ (0,1),\ 90^0)
\end{aligned}$$

and therefore the picture

$$tur(w) = \{((0,0),(0,1)),\ ((0,1),(1,1)),\ ((1,1),(1,2)),((1,2),(1,3))\}$$

consists of the four lines. It is easy to see that $tur(w) = dccp(uruu)$.

**Example 2.48** *The picture $dccp(urdl)$ can be described in the turtle mechanism as $tur(F+F+F+F)$ if we use $\alpha_0 = 90^0$ and $\alpha = 90^0$. We now consider the turtle grammar*

$$K_1' = (\{F\},\{F\},\ \{F \to F-F+F+FF-F-F+F\},\ F+F+F+F,\ 90^0,\ 90^0).$$

*Since $F \to F-F+F+FF-F-F+F$ corresponds to $r \to rdru^2rdr$, $u \to urul^2url$, $d \to dldr^2dld$ and $l \to luld^2lul$ if the direction is given by $0^0$, $90^0$, $-90^0$ and $180^0$, respectively, it is easy to see that $tur(K_1') = dccp(K_1)$.*

**Example 2.49** *Let the turtle grammar*

$$K_2 = (\{F\},\ \{F\},\ \{F \to F+F--F+F\},\ F,\ 0^0,\ 60^0)$$

*be given. For the word $F+F--F+F$ we get the sequence of configurations*

$$\begin{aligned}
c(\lambda) &= (\emptyset, (0,0), 0^0),\\
c(F) &= (\{((0,0),(1,0))\},\ (1,0),,\ 0^0),\\
c(F+) &= (\{((0,0),(1,0))\},\ (1,0),,\ 60^0),\\
c(F+F) &= (\{((0,0),(1,0)),((1,0),(\frac{3}{2},\frac{\sqrt{3}}{2}))\},\ (\frac{3}{2},\frac{\sqrt{3}}{2}),\ 60^0),\\
c(F+F-) &= (\{((0,0),(1,0)),((1,0),(\frac{3}{2},\frac{\sqrt{3}}{2}))\},\ (\frac{3}{2},\frac{\sqrt{3}}{2}),\ 0^0),\\
c(F+F--) &= (\{((0,0),(1,0)),((1,0),(\frac{3}{2},\frac{\sqrt{3}}{2}))\},\ (\frac{3}{2},\frac{\sqrt{3}}{2}),\ -60^0),\\
c(F+F--F) &= (\{((0,0),(1,0)),((1,0),(\frac{3}{2},\frac{\sqrt{3}}{2})),((\frac{3}{2},\frac{\sqrt{3}}{2}),(2,0))\},\ (2,0),\ -60^0),\\
c(F+F--F+) &= (\{((0,0),(1,0)),((1,0),(\frac{3}{2},\frac{\sqrt{3}}{2})),((\frac{3}{2},\frac{\sqrt{3}}{2}),(2,0))\},\ (2,0),\ 0^0),\\
c(F+F--F+F) &= (\{((0,0),(1,0)),((1,0),(\frac{3}{2},\frac{\sqrt{3}}{2})),((\frac{3}{2},\frac{\sqrt{3}}{2}),(2,0)),((2,0),(3,0))\},(2,0),0^0).
\end{aligned}$$

*The corresponding picture is given in the right picture of the first line of Figure 2.18. Since the associated Lindenmayer system $H_{K_2}$ is a D0L system, it generates a unique sequence of words. The pictures of the first six pictures generated by $K_2$ are given in Figure 2.18.*

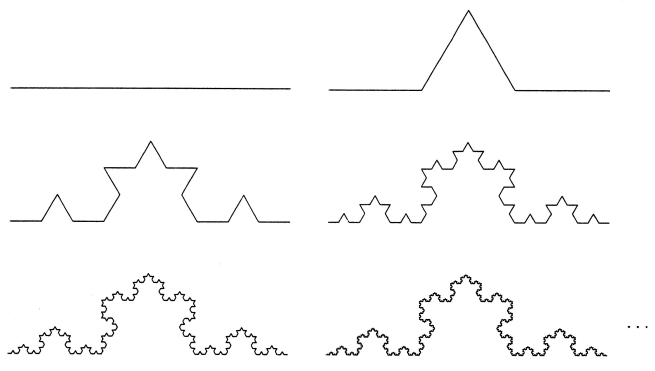


Figure 2.18: Pictures generated by $K_2$ in at most 5 steps

## 2.5.2 Applications for Simulations of Plant Developments

In this section we give descriptions for the development of some plants. Because branching occurs very often for plants, it is necessary to add a feature which can cover this aspect. Therefore we change the concept of a turtle grammar slightly. The start of a branch is denoted by the additional letter [ and its end by the letter ]. Moreover we use two stacks which remember the branching point and the direction which was used before starting the branch. We omit the formal definition of the operation and the drawing process with the additional use of *enc*. Furthermore, for $n \geq 1$, we write $enc_{+^n}$ and $enc_{-^n}$ to denote the a change of the direction by $n \cdot \alpha$ and $-n \cdot \alpha$, respectively (as usual $+ = +^1$ and $- = -^1$).

**Example 2.50** We consider the extended turtle grammar

$$(\{F\}, \{F \to FF[+FF][-F+[F]]\},\ F,\ 90^0,\ 20^0)).$$

Then the start word $F$ describes a upwards oriented unit line since the basic direction is $90^0$. Since the associated Lindenmayer system is a deterministic 0L system, we get a unique sequence of derived words and corresponding a unique sequence of pictures. The pictures obtained after 1, 2, 3, and 4 steps are given in Figure 2.19.

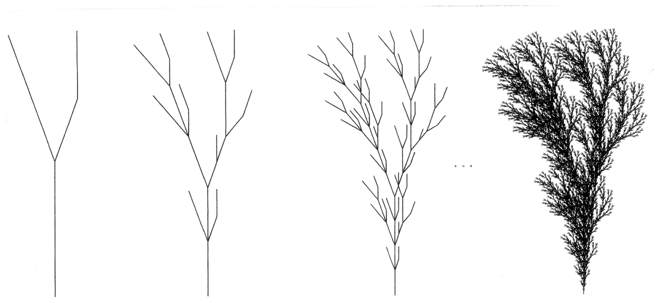


Figure 2.19: Pictures generated by the extended turtle grammar of Example 2.50

In the Figure 2.20 we present some further examples of descriptions of plants. In these figures the basic direction $\alpha_0$ is not mentioned; we use in all cases $\alpha_0 = 90^0$ since it is natural that plants grow upwards.

Hitherto we have only considered deterministic Lindenmayer systems with one table. We now discuss partly a system with two tables. Let

$$Z = (\{F\},\ \{F\},\ P_1,\ P_2,\ F,\ 90^0,\ 22.5^0)$$

with

$$\begin{aligned}
P_1 &= \{F \to F\ enc_{+^2}[F]F\ enc_{-^2}[F+[F+[F]]]\ enc_{+}[F+[F+[F]]]\},\\
P_2 &= \{F \to F\ enc_{+^2}[F]F\ enc_{-^2}[F+[F]]\ enc_{+}[F+[F]]\\
&\qquad F \to F\ enc_{+^2}[F]F\ enc_{-^2}[F]\ enc_{+}[F+[F]]\\
&\qquad F \to F\ enc_{+^2}[F]F\ enc_{-^2}[F+[F]]\ enc_{+}[F]\}
\end{aligned}$$

The pictorial interpretation of the rule of $P_1$ is

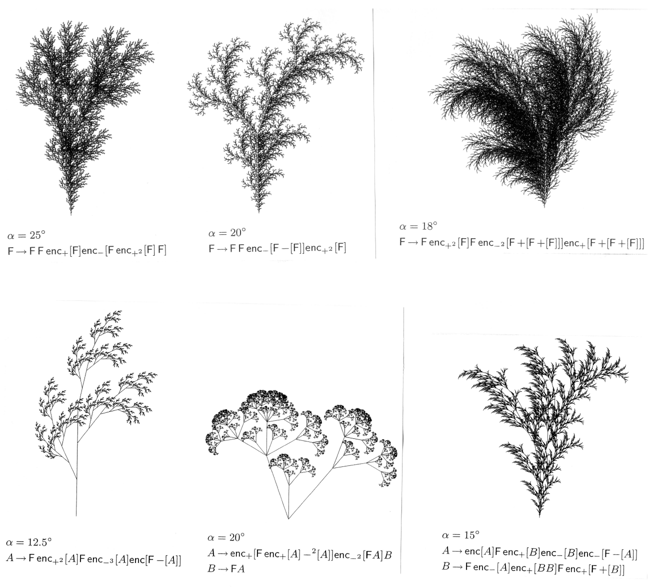
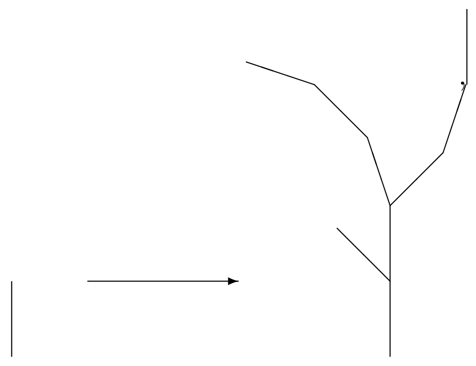

$\alpha = 25°$
$\mathsf{F} \rightarrow \mathsf{F\,F\,enc_{+}[F]enc_{-}[F\,enc_{+^2}[F]\,F]}$
$\alpha = 20°$
$\mathsf{F} \rightarrow \mathsf{F\,F\,enc_{-}[F-[F]]enc_{+^2}[F]}$
$\alpha = 18°$
$\mathsf{F} \rightarrow \mathsf{F\,enc_{+^2}[F]F\,enc_{-^2}[F+[F+[F]]]enc_{+}[F+[F+[F]]]}$
$\alpha = 12.5°$
$A \rightarrow \mathsf{F\,enc_{+^2}}[A]\mathsf{F\,enc_{-^3}}[A]\mathsf{enc[F-}[A]]$
$\alpha = 20°$
$A \rightarrow \mathsf{enc_{+}[F\,enc_{+}}[A]-^2[A]]\mathsf{enc_{-^2}[F}A]B$
$B \rightarrow \mathsf{F}A$
$\alpha = 15°$
$A \rightarrow \mathsf{enc}[A]\mathsf{F\,enc_{+}}[B]\mathsf{enc_{-}}[B]\mathsf{enc_{-}[F-}[A]]$
$B \rightarrow \mathsf{F\,enc_{-}}[A]\mathsf{enc_{+}}[BB]\mathsf{F\,enc_{+}[F+}[B]]$

Figure 2.20: Pictures generated by extended turtle grammars describing plant developments



If the conditions of the environment are bad, then the growth is smaller. This is modelled by the rules of $P_2$ where the first and second rule correspond to the situations that the upper right or left branch is shortened by one line, the third rule describes the simultaneous cancellation of one segment of the upper right branch and one segment of the upper left branch. In dependence of the environmental situations we have to apply a sequence $P_{i_1}P_{i_2}P_{i_3}\dots$ of tables. In Figure 2.21 we have given the results for three different sequences (where the sequence in the Figure only gives the indexes. For each sequence we present three different pictures because we can use different rules of $P_2$ for different occurrences of $F$ in a word. We see that the differences of the generated pictures of the plants come mostly from the difference in the sequences of tables; if the same sequence is used, but different rules are applied the three plants do not differ too much.

We refer to [20], [3]and [19] for a more detailed discussion of the application of turtle and chain code picture Lindenmayer systems for the description of the development of plants. We mention that this approach in combination with a precise drawing of a leave,

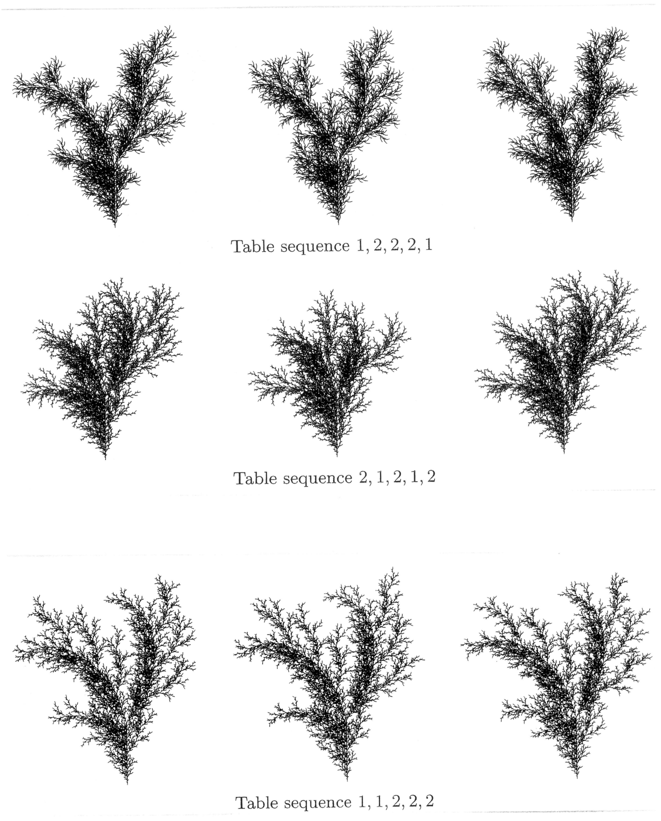


Table sequence $1, 2, 2, 2, 1$
Table sequence $2, 1, 2, 1, 2$
Table sequence $1, 1, 2, 2, 2$

Figure 2.21: Pictures/plants generated by extended turtle grammar $Z$

a stalk etc. is sometimes used in computer graphics for the drawing of plants.

### 2.5.3 Space-Filling Curves

We start with an easy mapping which maps any point of the unit interval to a point of the unit square and conversely. For any point $X$ of the unit interval, there is a representation as a decimal with an infinite number of digits after the decimal point, i.e., $X = 0.x_1x_2x_3\ldots$ with $x_i \in \{0, 1, 2, \ldots, 9\}$ for $i \geq 1$ (the number 1 is represented as $0.999\ldots$). We define

$$f(X) = (0.x_1x_3x_5\ldots, 0.x_2x_4x_6\ldots)\,.$$

Obviously, any point $X$ of the unit interval is mapped to a point of the unit square. Obviously, $f^{-1}$ also exists and we have

$$f^{-1}(0.a_1a_2\ldots, 0.b_1b_2\ldots) = 0.a_1b_1a_2b_2\ldots\ .$$

However, these functions have some very bad properties. Both functions are not bijective. This can be seen from the different points $X = 0.0090909\ldots$ and $Y = 0.1000\ldots$ of the unit interval which lead to the same point

$$f(X) = (0.0999\ldots, 0.000\ldots) = (0.1000\ldots, 0.000\ldots) = f(Y)$$

of the unit square. Moreover, the mappings are not continuous.

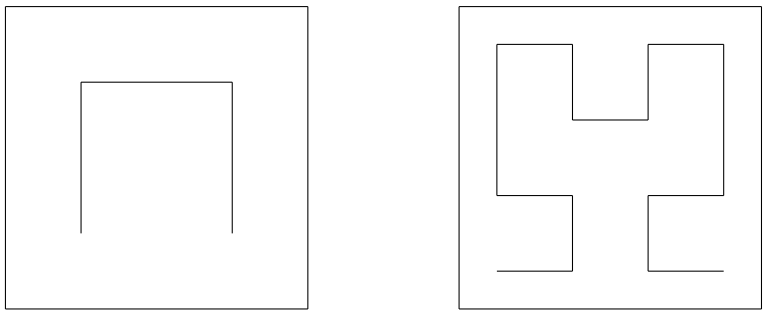


Figure 2.22: The basic steps of the construction of the Hilbert curve

From the mathematical point of view one is interested in bijective and continuous mappings. In 1878 Georg Cantor gave a bijective mapping from the unit interval onto the unit square. However, in 1879 Eugen Netto proved the non-existence of a mapping which is bijective as well as continuous. In 1890 Guiseppe Peano and David Hilbert gave examples of continuous and surjective (but not bijective) mappings. The graph of such a function has to be a simple curve which fills the unit square completely. Therefore such curves are called *space-filling* curves.

We give here as an example the construction of Hilbert. We start with the curve given in the left part of Figure 2.22 . Then we take this curve as a basic element, take four copies scaled to the half (in width and height), rotate them in such a way that a continuous connection is possible (in the two upper quadrants we do no rotation, in the lower quadrants we rotate by 90 to the right and to the left, respectively; see the right part of Figure 2.22 b). This process is iterated. The fifth iteration is shown in Figure 2.23.

It is easy to see that the distance of an arbitrary point of the unit square to the curve obtained in the $n$-th iteration is at most

$$\frac{1}{2^n}\cdot\frac{1}{4}\sqrt{2}.$$

Therefore the distance tends to 0 if $n$ tends to infinity.

The sequence of pictures defined by the Hilbert curve construction is given by the turtle grammar

$$(\{X,Y,F\},\ \{X\to +YF{-}XFX{-}FY{+},\ Y\to -XF{+}YFY{+}FX{-},\ F\to F\},\ X,\ 0^0,\ 90^0)\,.$$

The sequence of generated words starts with

$$\begin{aligned} w_0 &= X,\\ w_1 &= +YF-XFX-FY+,\\ w_2 &= +-XF+YFY+FX-\ F-\ +YF-XFX-FY+\ F\ +YF-XFX-FY+\\ &\qquad -F\ -XF+YFY+FX-+. \end{aligned}$$

The pictures corresponding to $w_1$ and $w_2$ are shown in Figure 2.22.

The Peano curve and some other space-filling curves can also be described by turtle grammars.

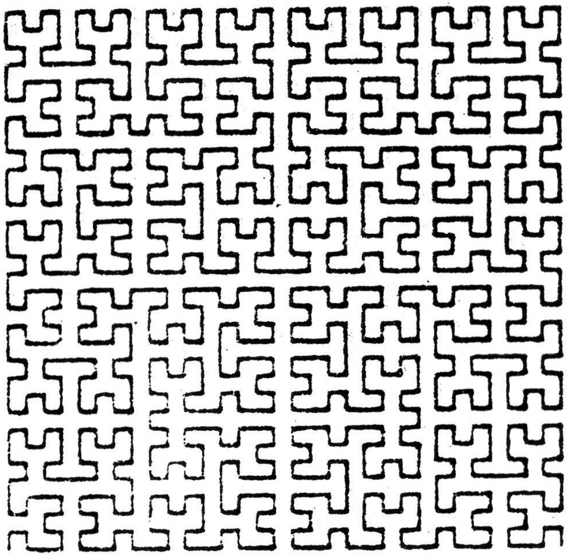


Figure 2.23: The fifth iteration step of the construction of the Hilbert curve

### 2.5.4 Kolam Pictures

As our final application we consider kolams. These are special figures which are mostly drawn by the women in India into the sand before the house. These pictures should prevent bad ghosts etc. and bring positive features as fertility and so on. Some typical kolams – which are very often symmetric – are shown in Figure 2.24. Mostly the women start by drawing some points and then the kolam itself which consists of curves going around the points. The kolams differ from region to region. One of the most known kolams is Krishna's footlace. The basic figure is shown in the left part of Figure 2.25; the right part of Figure 2.25 shows a slightly more complicated version.

Some Indian computer scientists especially the married couple Rani and Gift Siromoney and their co-workers have described kolams using the idea of chain codes where they use curves instead of unit lines as basic elements. We illustrate this idea by Krishna's footlace. We consider the alphabet $\{A, B, B', B''\}$, which symbols are interpreted as follows: $A$ is a straight line, and $B$ is a "circle" (see left part of Figure 2.25). The picture of $B'$ is the same as the that corresponding to $B$. $B''$ is associated with the half of the arcs connecting the basic Krishna's footlace in the right part of Figure 2.25. Now we consider the DOL system

$$(\{A, B, B', B''\}, \{A \to A, B \to B''AB'ABAB'AB'', B' \to B', B'' \to B''\}, ABABABAB)$$

which generates a sequence of words beginning with

$$ABABABAB,$$
$$AB''AB'ABAB'AB''AB''AB'ABAB'AB''AB''AB'ABAB'AB''AB''AB'ABAB'AB''$$

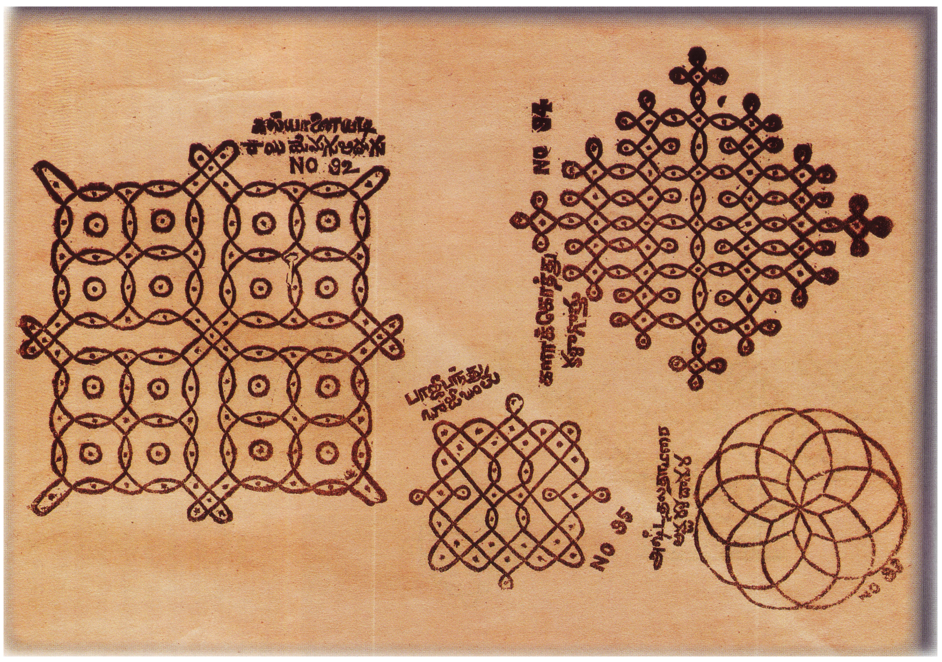


Figure 2.24: Some typical kolam pictures

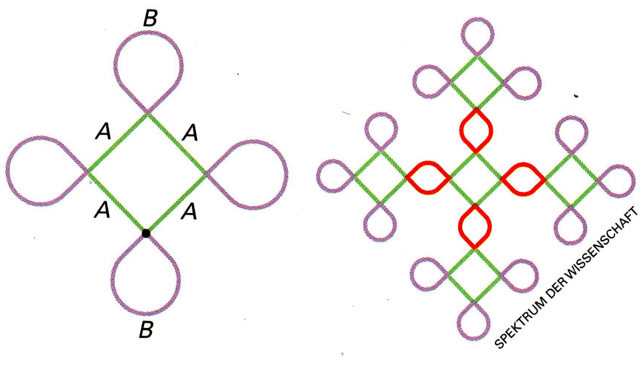


Figure 2.25: Krishna's footlace

which correspond to the pictures shown in Figure 2.25. In the sequel we replace the most upper, most right, most down and most left "circle" $B$ by a picture corresponding to $B''AB'ABAB'AB''$ which is given by one of the pictures between two of the original lines corresponding to $A$.

# Chapter 3

# Siromoney Matrix Grammars

In this chapter we study a different grammatical approach for generation of pictures. In the first step we generate a matrix, i.e., a two-dimensional scheme of symbols, and in a second step we replace the symbols by (chain code) pictures.

In the literature one can find different ideas to generate two-dimensional schemes of letters. We first mention here array grammars. Here instead of words in all cases we replace letters by two-dimensional objects and start with a two-dimensional object. Usually, one assumes that one has a grid, and the unit squares of the grid can be filled with letters. Obviously, one cannot replace a nonterminal symbol which is surrounded by symbols by an object consisting of some letters because we destroy connections between the surrounding symbols. Therefore, if one wants to get a context-free variant (i.e., the application of a rule does not take into consideration the surrounding symbols), the rules can be applied only to nonterminals which are at the border of the object. Moreover, we take into consideration the free places around a symbol which can be filled by letters. As an example we consider an array grammar with nonterminals $A, A', B, B'$, terminals $a, b_1, b_2, b_3, b_4$, rules

$$\begin{array}{|c|}\hline A\\ \hline\end{array} \to \begin{array}{|c|}\hline a\\ \hline\end{array}, \quad \begin{array}{|c|c|}\hline A & \ \\ \hline\end{array} \to \begin{array}{|c|c|}\hline a & A\\ \hline\end{array}, \quad \begin{array}{|c|c|}\hline A' & \ \\ \hline\end{array} \to \begin{array}{|c|c|}\hline b_4 & A\\ \hline\end{array},$$

$$\begin{array}{|c|}\hline B\\ \hline\end{array} \to \begin{array}{|c|}\hline a\\ \hline\end{array}, \quad \begin{array}{|c|c|}\hline \ & B\\ \hline\end{array} \to \begin{array}{|c|c|}\hline B & a\\ \hline\end{array}, \quad \begin{array}{|c|c|}\hline \ & B'\\ \hline\end{array} \to \begin{array}{|c|c|}\hline B & b_2\\ \hline\end{array},$$

$$\begin{array}{|c|}\hline A\\ \hline \ \\ \hline\end{array} \to \begin{array}{|c|}\hline b_1\\ \hline B'\\ \hline\end{array}, \quad \begin{array}{|c|}\hline B\\ \hline \ \\ \hline\end{array} \to \begin{array}{|c|}\hline b_3\\ \hline A'\\ \hline\end{array},$$

and take $A$ as the start symbol. A typical derivation is given in Figure 3.1.

It is easy to see that our array grammar generates all two dimensional objects which consist of a certain number of rows, say $n$ rows, such that, for $n = 1$, the row consists of a certain number of $a$'s, and for $n > 1$, the following conditions hold:
– the first row starts with an $a$ and ends with $b_1$,
– any $i$-th row, where $i$ is an odd number and $i < n$, starts with $b_4$ and ends with $b_1$,
– any $i$-th row, where $i$ is an even number and $i < n$, starts with $b_3$ and ends with $b_2$,
– the $n$-th row starts with $a$ and ends with $b_2$ or starts with $b_4$ and ends with $a$, if $n$ is even or odd, respectively,
– besides the start and end letter, the rows contain only $a$'s,
– for $i \geq 0$ and $2i < n$, the last letter of the $(2i + 1)$-st row is above the last letter of the

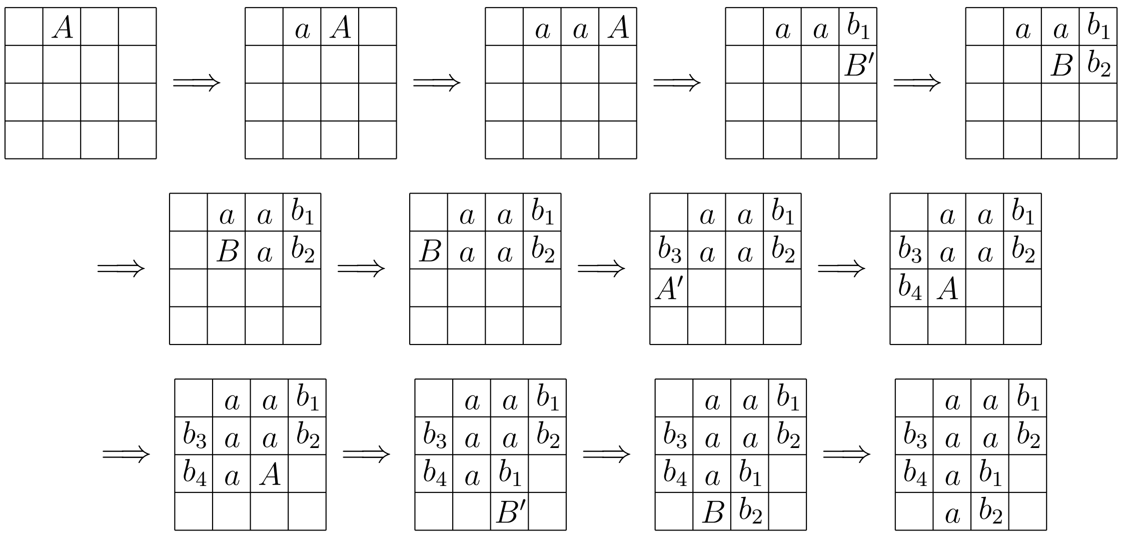
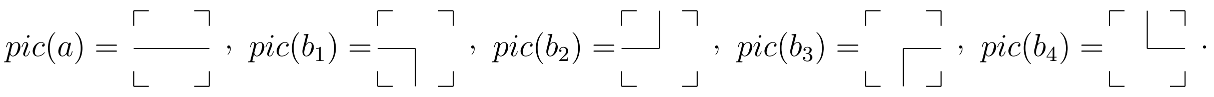


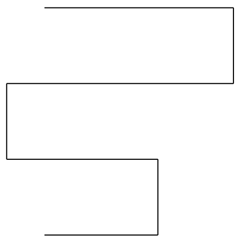
Figure 3.1: Derivation of a array grammar (we give all necessary squares of the grid although if not all are filled from the beginning)

$(2i)$-th row,
– for $i \geq 1$ and $2i+1 < n$, the first letter of the $(2i)$-th row is above the first letter of the $(2i+1)$-st row.

We now interpret the letters $a, b_1, b_2, b_2, b_3$ and $b_4$ as pictures within a square of with sides of length 2 by the mapping *pic* as follows (the small angles give the corners of the square):



The two-dimensional object generated by the derivation given in Figure 3.1 is then transformed into



The set of all pictures consist of an arbitrary number of vertical lines of arbitrary lengths which are connected analogously to the given picture.

We do not discuss array grammars in detail (we refer for further information to [26], [27], [28], [36], [29], [37] and [23]); in the following sections we consider Siromoney matrix grammars where the basic idea is very similar. Again, we first generate two-dimensional objects of letters and interpret the letters by pictures; however, the derivation of the two-dimensional objects of letters is different from that used in array grammars.

## 3.1 Definitions and Examples

In Siromoney matrix grammars we first derive a word $A_1A_2\dots A_n$ by a usual phrase structure grammar, and then we derive from any letter $A_i$ a word $w_i$ by a regular grammar in normal form and write these words as columns; therefore we get an object consisting of $n$ columns of letters. Before we give the formal definition of Siromoney matrix grammars we introduce some concepts which describe the generated two-dimensional objects of letters.

**Definition 3.1** *A matrix $(a_{i,j})_{k,l}$ over a set $V$ is a rectangular scheme with $k$ rows and $l$ columns, and the element $a_{i,j} \in V$ is in the meet of the $i$-th row and the $j$-th column, $1 \le i \le k$ and $1 \le j \le l$.*

*A quasi-matrix $(a_{i,j})_{k_1,k_2,\dots,k_l}$ over $V$ is a scheme with $l$ columns of length $k_1, k_2, \dots, k_l$ and the element $a_{i,j} \in V$ is the $i$-th element of the $j$-th column (where we count from above to below), $1 \le i \le k_j$ and $1 \le j \le l$.*

By definition, a quasi-matrix consists of some columns of different length, whereas the lengths of all columns of a matrix are equal. Thus each matrix $(a_{i,j})_{k,l}$ is a quasi-matrix $(a_{i,j})_{l,l,\dots,l}$, and each quasi-matrix $(a_{i,j})_k$ (with exactly one column) is a matrix $(a_{i,j})_{k,1}$.

If a matrix $M$ has $m$ rows and $n$ columns, then we say that $M$ is a $(m,n)$-matrix.

We now give the notion of a shift of an extended chain code picture which is already implicitly given in connection with the equivalence relation for (drawn and basic) chain code pictures (see Section 2.1, page 27). For an extended chain code picture $p$ and two integers $m$ and $n$, let $sh_{m,n}(p)$ be the picture such that

$$((u,v), b((u,v))) \in p \text{ iff } ((u+m, v+n), b((u+m, v+n)) \in sh_{m,n}(p).$$

We say that $sh_{m,n}(p)$ is obtained by a shift of $p$ by $(m,n)$.

We now give an interpretation of letters of a (quasi-)matrix by extended chain code pictures.

**Definition 3.2** *Let $T$ be an alphabet. For two natural numbers $s \ge 1$ and $t \ge 1$, let $CCP_{s,t}$ be the set of all extended basic chain code pictures $p$ such that, for $(m,n) \in V(p)$, $0 \le m \le s$ and $0 \le n \le t$. Let $pic_{s,t} : T \to CCP_{s,t}$ be a mapping.*

*For a quasi-matrix $M = (a_{i,j})_{k_1,k_2,\dots,k_l}$ or a matrix $M = (a_{i,j})_{k,l}$, we define the picture $Pic(M)$ as the set*

$$Pic(M) = \bigcup_{\substack{1 \le j \le l \\ 1 \le i \le k_j}} sh_{(i-1)s,-jt}(pic_{s,t}(a_{i,j}))$$

*and*

$$Pic(M) = \bigcup_{\substack{1 \le j \le l \\ 1 \le i \le k}} sh_{(i-1)s,-jt}(pic_{s,t}(a_{i,j})),$$

*respectively.*

**Definition 3.3** *i) A Siromoney matrix grammar is a construct*

$$G = (N_1, N_2, I, T, P_1, P_2, S_1, s, t, pic_{s,t})$$

*where*
– $G_1 = (N_1, I, P_1, S_1)$ *is a phrase structure grammar,*
– $I \subseteq N_2$,
– *for any* $A \in I$, $G_A = (N_2, T, P_2, A)$ *is a regular grammar in normal form,*
– $s, t \in \mathbb{N}$,
– $pic_{s,t} : T \to CCP_{s,t}$.

*ii)* $M(G)$ *is the set of all matrices* $(a_{ij})_{k,l}$, $1 \le i \le k$, $1 \le j \le l$, $k \ge 1$, $l \ge 1$ *such that there is a word* $A_1A_2 \dots A_l \in L(G_1)$ *and* $a_{1j}a_{2j} \dots a_{kj} \in L(G_{A_j})$ *for* $1 \le j \le l$.

$QM(G)$ *is the set of all quasi-matrices* $(a_{ij})_{k_1,k_2,\dots,k_l}$, $l \ge 1$, $k_u \ge 1$ *for* $1 \le u \le l$ *such that there is a word* $A_1A_2 \dots A_l \in L(G_1)$ *and* $a_{1j}a_{2j} \dots a_{k_jj} \in L(G_{A_j})$ *for* $1 \le j \le l$.

*iii) We set*

$$PM(G) = \{Pic(M) \mid M \in M(G)\} \quad \textit{and} \quad PQM(G) = \{Pic(M) \mid M \in QM(G)\}.$$

This type of grammars was originally introduced by Rani Siromoney in [30]; however, there only matrices were considered. The interpretation as pictures (not necessarily chain code pictures) was firstly considered in [31] in order to describe some designs related to kolams (see Section 2.5.4). The interpretation as chain code pictures in the way given above was introduced by Ralf Stiebe in [32] and [33]. This interpretation leads to a generation of (extended) chain code pictures by Siromoney matrix grammars, and thus it is possible to compare this type of generation with that via words over $\pi$ as it was done in Chapter 2.

The grammar $G_1$ is called the horizontal grammar because it derives the intermediate (horizontal) word $A_1A_2 \dots A_l \in I^+$. The set $I$ is called the set of intermediates since its letters are used to build the horizontal words, which are intermediate steps of the (quasi-) matrix generation. The grammars $G_i$ with $i \in I$ are called vertical because they derive the (vertical) columns of the matrix.

We now give some examples.

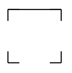
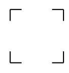
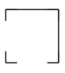
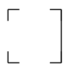
**Example 3.4** We consider the Siromoney matrix grammar

$$G = (\{S, S'\}, \{A, A', B, B'\}, \{A, B\}, P_1, P_2, S, 1, 1, pic)$$

where

$$\begin{aligned} P_1 &= \{S \to AS, S \to BS', S' \to AS', S' \to A\}, \\ P_2 &= \{A \to aA', A' \to a'A', A' \to a', B \to bB', B' \to b'B', B' \to b'\} \end{aligned}$$

and

$pic(a) =$ , $pic(a') =$ , $pic(b) =$ , $pic(b') =$ .

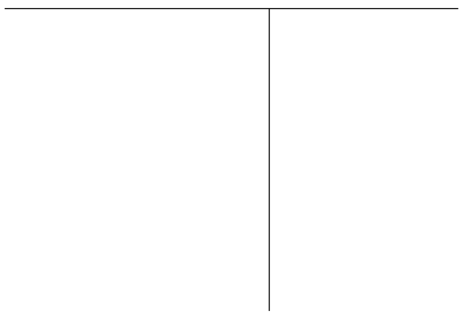
It is easy to see that

$$L(G_1) = \{A^nBA^m \mid n \ge 0, m \ge 0\}, \tag{3.1}$$
$$L(G_A) = \{a(a')^r \mid r \ge 1\},$$
$$L(G_B) = \{b(b')^s \mid s \ge 1\}. \tag{3.2}$$

Therefore all matrices generated by $G$ have the form

$$\begin{array}{cccccccc}
a & a & \dots & a & b & a & a & \dots & a \\
a' & a' & \dots & a' & b' & a' & a' & \dots & a' \\
a' & a' & \dots & a' & b' & a' & a' & \dots & a' \\
 & \vdots & & & \vdots & & & \vdots & \\
a' & a' & \dots & a' & b' & a' & a' & \dots & a' \\
a' & a' & \dots & a' & b' & a' & a' & \dots & a'
\end{array}$$

where $n$ columns and $m$ columns are before and after the $b$-column, respectively. Moreover, the generated pictures have the form



where the length of the horizontal line before the vertical line is $n+1$, the length of the horizontal line after the vertical line is $m$ and the length of the vertical line is $s$, where the values for $n$, $m$ and $s$ can be taken from (3.1) and (3.2).

If we consider quasi-matrices generated by $G$, then we have almost the same structure, only the length of the columns can be different. Since the pictures corresponding to $a'$ are empty, it is obvious that $PM(G) = PQM(G)$.

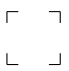
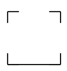
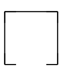
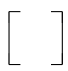
**Example 3.5** Let the Siromoney matrix

$$G = (\{S\}, \{A, B, A', B'\}, \{A, B\}, \{a, a', b, b'\}, P_1, P_2, S, 1, 1, pic)$$

with

$$\begin{aligned}
P_1 &= \{S \to ABA\}, \\
P_2 &= \{A \to aA, A \to a', B \to bB', B \to b, B' \to b'B, B' \to b'\}
\end{aligned}$$

and

$pic(a) =$  , $pic(a') =$  , $pic(b) =$  , $pic(b') =$  .

be given. Then we get

$$\begin{aligned}
L(G_1) &= \{ABA\}, \\
L(G_A) &= \{a^n a' \mid n \geq 0\}, \\
L(G_B) &= \{b(b')^n \mid n \geq 0\},
\end{aligned}$$

and therefore $M(G)$ and $PM(G)$ consist of all matrices and all pictures of the form given in the left and right part of Figure 3.2, respectively.

$$\begin{array}{ccc} a & b & a \\ a & b' & a \\ a & b' & a \\ \vdots & & \vdots \\ a & b' & a \\ a' & b' & a' \end{array}$$

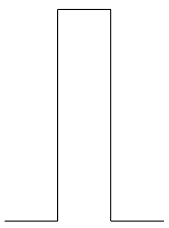


Figure 3.2: Typical form of the elements of $M(G)$ and $PM(G)$

We note that the picture language $PM(G)$ coincides with the language $\{bccp(ru^n rud^n r) \mid n \geq 1\}$ which is shown to be in $\mathcal{CCP}(CF) \setminus \mathcal{CCP}(REG)$ in part i) of the proof of Theorem 2.13. With respect to Siromoney grammars we have used only regular grammars.

If we consider the quasi-matrices and pictures of $PQM(G)$, then we get three columns of different length and pictures where the right and left horizontal lines can be positioned at places different from the lower ends of the vertical lines.

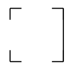
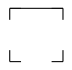
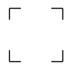
**Example 3.6** We consider the Siromoney grammar

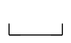
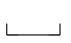
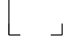
$$G = (\{S,T\}, \{A,B,C,B'\}, \{A,B,C\}, \{a,b,b',b'',b''',c\}, P_1, P_2, S, 1, 1, pic)$$

where

$$\begin{aligned} P_1 &= \{S \to AT, T \to BT, T \to BC\}, \\ P_2 &= \{A \to aA, A \to a, B \to b''', B \to bB', B' \to b'B', B' \to b'', C \to cC, C \to c\} \end{aligned}$$

and $pic$ is given by

$pic(a) =$  , $pic(b) =$  , $pic(b') =$  ,

$pic(b'') =$  , $pic(b''') =$  , $pic(c) =$  .

Then we obtain

$$\begin{aligned} L(G_1) &= \{AB^nC \mid n \geq 1\}, \\ L(G_A) &= \{a^m \mid m \geq 1\}, \\ L(G_B) &= \{b'''\} \cup \{b(b')^k b'' \mid k \geq 0\}, \\ L(G_C) &= \{c^m \mid m \geq 1\}. \end{aligned}$$

Then we get that $M(G)$ consists of all matrices of the form

$$\begin{array}{cccccc} a & b & b & \dots & b & c \\ a & b' & b' & \dots & b' & c \\ a & b' & b' & \dots & b' & c \\ & \vdots & & & \vdots & \\ a & b' & b' & \dots & b' & c \\ a & b'' & b'' & \dots & b'' & c \end{array} \qquad \text{or} \qquad a \quad b''' \quad b''' \quad \dots \quad b''' \quad c\ ,$$

and therefore $PM(G)$ is the set of all rectangles.

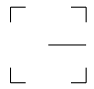
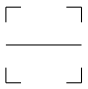
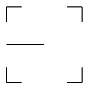
**Example 3.7** Let the Siromoney matrix grammar

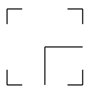
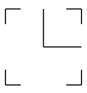
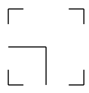
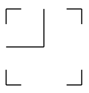
$$G = (\{S,T\},\{A,B,C,A',A'',C'\},\{A,B,C\},\{a,b,c,a_1,a_2,c_1,c_2\},S,P_1,P_2,2,2,pic)$$

with

$$\begin{aligned}
P_1 &= \{S \to AT, T \to BT, T \to C\},\\
P_2 &= \{A \to a, A \to aA', A' \to a_1A'', A'' \to a_2A', A' \to a, A'' \to a_2,\\
&\qquad B \to bB, B \to b, C \to c, C \to c_1C', C' \to c_2C, C' \to c_2\}
\end{aligned}$$

and

$pic(a) =$  , $pic(b) =$  , $pic(c) =$  ,

$pic(a_1) =$  , $pic(a_2) =$  , $pic(a_1) =$  , $pic(a_1) =$  .

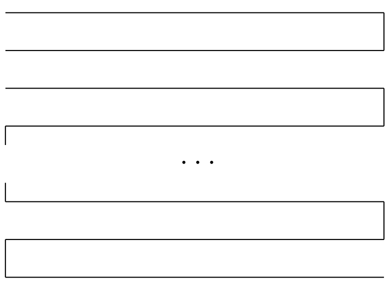
We obtain

$$\begin{aligned}
L(G_1) &= \{AB^nC \mid n \geq 0\},\\
L(G_A) &= \{a(a_1a_2)^n \mid n \geq 0\} \cup \{a(a_1a_2)^n a \mid n \geq 0\},\\
L(G_B) &= \{b^m \mid m \geq 1\},\\
L(G_C) &= \{(c_1c_2)^k c \mid k \geq 0\} \cup \{(c_1c_2)^k \cup k \geq 1\}.
\end{aligned}$$

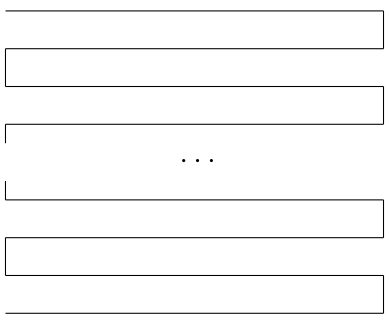
The matrices generated by $G$ are all matrices of one of the forms

$$\begin{array}{cccccc}
a & b & b & \dots & b & c_1\\
a_1 & b & b & \dots & b & c_2\\
a_2 & b & b & \dots & b & c_1\\
a_1 & b & b & \dots & b & c_2\\
 & \vdots & & & \vdots & \\
a_2 & b & b & \dots & b & c_1\\
a_1 & b & b & \dots & b & c_2\\
a_2 & b & b & \dots & b & c
\end{array}
\quad \text{or} \quad
\begin{array}{cccccc}
a & b & b & \dots & b & c_1\\
a_1 & b & b & \dots & b & c_2\\
a_2 & b & b & \dots & b & c_1\\
a_1 & b & b & \dots & b & c_2\\
 & \vdots & & & \vdots & \\
a_2 & b & b & \dots & b & c_1\\
a_1 & b & b & \dots & b & c_2\\
a_2 & b & b & \dots & b & c_1\\
a & b & b & \dots & b & c_2
\end{array}$$

where the number of rows is odd and even in the first and second case, respectively. These matrices result in pictures of the forms



or



## 3.2 Hierarchies of Siromoney Matrix Languages

We want to give an analogy to the Chomsky hierarchy known for word languages generated by the different types of phrase structure grammars. Therefore we first introduce the corresponding notions.

A Siromoney matrix grammar $G = (N_1, N_2, I, T, P_1, P_2, S, s, t, pic_{s,t})$ is called an X Siromoney matrix grammar if $G_1 = (N_1, I, P_1, S)$ is an X grammar.

$\mathcal{M}(X)$, $\mathcal{QM}(X)$, $\mathcal{PM}(X)$ and $\mathcal{PQM}(X)$ denote the families of all languages $M(G)$, $QM(G)$, $PM(G)$ and $PQM(G)$, respectively, where $G$ is an X Siromoney matrix grammar.

**Theorem 3.8** *i)* $\mathcal{M}(REG) \subset \mathcal{M}(CF) \subset \mathcal{M}(CS) \subset \mathcal{M}(RE)$,
*ii)* $\mathcal{QM}(REG) \subset \mathcal{QM}(CF) \subset \mathcal{QM}(CS) \subset \mathcal{QM}(RE)$,
*iii)* $\mathcal{PM}(REG) \subset \mathcal{PM}(CF) \subset \mathcal{PM}(CS) \subset \mathcal{PM}(RE)$,
*iv)* $\mathcal{PQM}(REG) \subset \mathcal{PQM}(CF) \subset \mathcal{PQM}(CS) \subset \mathcal{PQM}(RE)$.

*Proof.* We only give the proof for $\mathcal{M}(CF) \subset \mathcal{M}(CS)$ and $\mathcal{PM}(CF) \subset \mathcal{PM}(CS)$. The corresponding proofs for the other two strict inclusions can analogously be given using $\{a^nba^n \mid n \geq 1\} \in \mathcal{L}(CF) \setminus \mathcal{L}(REG)$ and a recursively enumerable language $L$ over a single letter alphabet which is not in $\mathcal{L}(CS)$. The proofs for the case of quasi-matrices are then literally the same proofs.

$\mathcal{M}(CF) \subset \mathcal{M}(CS)$. The inclusion holds by definition. We now prove the strictness of the inclusion.

By Example 1.4 and part ii) of the proof of Theorem 1.11, there exist a grammar $G_1 = (N_1, \{a', b'\}, P_1, S)$ with

$$L(G_1) = \{b'b'(a')^{2^n}b'b' \mid n \geq 0\} \in \mathcal{L}(CS) \setminus \mathcal{L}(CF)$$

(use $a'$ and $b'$ instead of $a$ and $b$ in all places of the construction). We consider the monotone Siromoney grammar

$$G = (N_1, \{a', b'\}, \{a', b'\}, \{a, b\}, P_1, \{a' \to a, b' \to b\}, S, 1, 1, pic).$$

Then all generated (quasi-)matrices consist of exactly one row $bba^{2^n}bb$. By definition $M(G) \in \mathcal{M}(CS)$. Let us assume that there is a context-free Siromoney matrix grammar

$$H = (N_1', N_2', I', \{a, b\}, P_1', P_2', S', s, t, pic_{s,t})$$

such that $M(H) = M(G)$. It is easy to see that all rules in $P_2$ are of the $i \to a$ or $i \to b$ where $i \in I'$. Let

$$I_a = \{i \mid i \in I',\ i \to a \in P_2'\} \text{ and } I_b = \{i \mid i \in I',\ i \to b \in P_2'\}.$$

Let $H_1 = (N_1', I', P_1', S')$. It is clear that

$$L(H_1) \subseteq \bigcup_{n \geq 0} I_b^2 I_a^{2^n} I_b^2$$

and that. for any $n \geq 0$, there is a word $w_n = b_1 b_2 a_1 a_2 \dots a_{2^n} b_3 b_4 \in L(H_1)$ such that $b_1, b_2, b_3, b_4 \in I_b$ and $a_j \in I_a$ for $1 \leq j \leq 2^n$. Let

$$H_1' = (N_1' \cup I', \{a, b\}, P_1' \cup \{i \to a \mid i \in I_a\} \cup \{i \to b \mid i \in I_b\}, S').$$

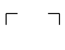
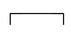
Then $H_1$ is context-free. By the above properties of $L(H_1)$, it is easy to see that

$$K = L(H_1') = \{bba^{2^n}bb \mid n \geq 0\}$$

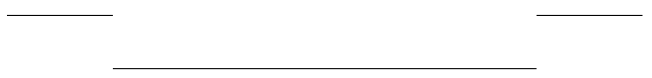
and thus $K \in \mathcal{L}(CF)$. However, on the other hand, $K \notin \mathcal{L}(CF)$ by part ii) of the proof of Theorem 1.11. This contradiction proves that $M(G) \notin \mathcal{M}(CF)$.

$\mathcal{PM}(CF) \subset \mathcal{PM}(CS)$. Again, the inclusion holds by definition. We prove the strictness.

We complete the definition of $G$ by

$pic(a) =$  and $pic(b) =$  .

Then all pictures of $PM(G)$ have the form



where the two upper lines have the length 2 and the lower line has length $2^n$ for some $n \geq 0$ ($n = 3$ in the picture).

Now assume that there is a context-free Siromoney grammar

$$H = (N_1', N_2', I', T', P_1', P_2', S, s, t, pic_{s,t})$$

with $PM(H) = PM(G)$. By the structure of the pictures in $PM(G)$ – without loss of generality – we can assume that $s = 1$ and therefore $A \to a \in T'$ for all rules in $P_2'$.

We set $m = \lceil \frac{2}{t} \rceil$ and $m' = tm - 2$. Then the two parts of a picture in $PM(H)$ consisting of the upper line of length 2 and the lower line of length $m'$ are covered by the first and last $m$ letters, respectively, and $m'$ .

Let $p$ be the constant of the pumping lemma for the context-free grammar $H_1 = (N_1', I', P_1', S)$. We choose $n$ such that

$$2^n - 2m' < 2^n - 2m' + 2tp < 2^{n+1} - 2m' \text{ and } tp < 2^n - 2m'. \tag{3.3}$$

Then the picture with a lower line of length $2^n$ is obtained from a word $x = x_1 x_2 \dots x_q$ where $q = \frac{2^n+4}{t}$. Moreover, the beginning and end of the picture of length $tm = m' + 2$ is covered by $x_1 x_2 \dots x_m$ and $x_{q-m+1} x_{q-m+2} \dots x_q$, and the middle part consisting of a lower line of length $2^n - 2m'$ is covered by $x_{m+1} x_{m+2} \dots x_{q-m}$. By our choices and the pumping lemma, $x = x_1 x_2 x_3 x_4 x_5$ and $x' = x_1 x_2^2 x_3 x_4^2 x_5 \in L(H_1)$. If $x_2$ or $x_4$ contribute to the upper line, then we get a picture from $x'$ where the upper line consist of at least 5 unit lines, which contradicts the structure of the picture in $PM(H)$. If $x_2$ and $x_4$ only contribute to the lower line, then the lower line is extended at most by $tp$ unit lines. But by (3.3), the length of the lower line is not a power of 2, which contradicts the structure of the picture in $PM(H)$, again.

Since we get a contradiction in all cases, our assumption has to fail. □

We now give a results which compares the families of picture languages generated by chain code picture grammars and Siromoney matrix grammars.

**Theorem 3.9** *i) $\mathcal{PQM}(CF)$ is a subset of $\mathcal{CCP}_\updownarrow(CF)$.*
*ii) $\mathcal{PM}(REG)$ is not contained in $\mathcal{CCP}_\updownarrow(CF)$.*
*iii) $\mathcal{CCP}(REG)$ is not contained in $\mathcal{PM}(CF)$.*

*Proof.* i) Let $L \in \mathcal{PQM}(CF)$. Then there is a context-free Siromoney matrix grammar $G = (N_1, N_2, I, T, P_1, P_2, S, s, t, pic)$ such that $L = PQM(G)$. For any $a \in T$, let $w_a \in \pi_\updownarrow$ be an extended chain code word such that $dccp(w_a) = ((0,0), pic(a), (0,-t))$, i.e., we start the drawing in $(0,0)$ and end in $(0,-t)$ which are the right upper and right lower corner of the rectangle in which $pic(a)$ is placed; it is obvious that such a word $w_a$ exists since we can extend another drawing by adding in the beginning and end some moves in the state "pen-up". Now we construct the extended context-free chain code grammar

$$H = (N_1 \cup N_2, \pi_\updownarrow, P_1 \cup P_2', S)$$

with

$$P_2' = \{A \to w_a B \uparrow u^t \downarrow \mid A \to aB \in P_2\} \cup \{A \to w_a \uparrow r^s u^t \downarrow \mid A \to a \in P_2\}.$$

It is easy to see that $A \Longrightarrow^* a_1 a_2 \dots a_n$ in $G_A$ if and only if

$$A \Longrightarrow w_{a_1} w_{a_2} \dots w_{a_n} \uparrow r^s u^t \downarrow (\uparrow u^t \downarrow)^{n-1} = z.$$

Moreover, the picture which corresponds to the column $(a_1 a_2 \dots a_n)^T$ is identical to $dccp(w_{a_1} w_{a_2} \dots w_{a_n})$ and the drawing ends in $(0, -nt)$. If we complete the drawing according to $z$ then we move in the state "pen-up" $s$ steps to the right and $nt$ steps upwards, i.e., $dccp(z)$ is also the picture corresponding to $(a_1 a_2 \dots a_n)^T$ and the drawing ends in $(s, 0)$, i.e., it ends in that point where we have to start the drawing of the next column.

Thus we get $bccp(H) = PQM(G)$. Therefore $L \in \mathcal{CCP}(CF)$.

ii) We consider the set $K$ of all rectangles. By Example 3.6, $K \in \mathcal{PM}(REG)$. On the other hand, one can show by pumping arguments that we cannot ensure the equal length of the two vertical lines if we use context-free chain code picture grammars (more precisely, if we can generate words $z$ such that $bccp(z)$ is a rectangle of sufficiently large height and width, then we can also generate a word which does not describe a rectangle). For a detailed proof we refer to [25].

iii) We consider the word language

$$L = \{d^k r^l d^m \mid k \geq 0,\ l \geq 0,\ m \geq 0\},$$

which is generated by the regular grammar

$$G = (\{A, B, C\}, \pi, \{A \to dA, A \to dB, B \to rB, B \to rC, C \to dC, C \to d\}, A),$$

and the picture language $P_L = bccp(L)$. By construction $P_L \in \mathcal{CCP}(REG)$.

We now prove that $P_L \notin \mathcal{PM}(CF)$. Assume the contrary, i.e., assume that there is a context-free Siromoney matrix grammar $H = (N_1, N_2, I, T, P_1, P_2, S, s, t, pic)$ with $PM(H) = P_L$. Let $n = \#(N_2)$. Then $bccp(d^{(n+1)t} r^{3s} d^{n^2(n+1)t})$ is contained in $P_L$. Now there is a column such that its picture is $bccp(r^s)$. Let this column be generated from the intermediate $A$. The column is a vertical word $w_1 a w_2$, where $|w_1| \geq n+1$, $|w_2| \geq n^2(n+1)$, $1 \leq |a| \leq 2$, $Pic(w_1)$ and $Pic(w_2)$ are empty and $Pic(a) = bccp(r^s)$.

According to the pumping lemma for regular languages (see Theorem 1.7 and note that the constant can be chosen as $n+1$), there is a decomposition $w_1 = v_1v_2v_3$ such that $v_1v_2^iv_3w_2 \in L(G_A)$ for $i \geq 0$. Moreover, let $|v_2| = p$. Then $1 \leq p \leq n$. Furthermore, $w_2$ can be written as $w_2 = u_1u_2 \dots u_{n^2}w_2'$ where $|u_j| = n+1$ for $1 \leq j \leq n^2$. Again, each $u_j$ has a decomposition $u_j = u_j'u_j''u_j'''$ such that with $z_j = u_j'u_j'''$ all words in

$$K = \{v_1v_2^iv_3 \mid i \geq 0\}\{a\}\{u_1, z_1\}\{u_2, z_2\} \dots \{u_{n^2}, z_{n^2}\}\{w_2'\}$$

belong to $L(G_A)$. Since $|u_j''| \leq n$ for $1 \leq j \leq n^2$, there are at least $n$ words $u_{i_1}'', u_{i_2}'', \dots, u_{i_n}''$, which have the same length $q$ for some $q$ with $1 \leq q \leq n$. We now take $p$ of these words and $z_j$ instead of $u_i$ and choose $i = q$ to choose a word from $K$. This word has the form $z = v_1v_2^qv_3auw_2'$ where $u$ is a word of length $n^2(n+1) - qp$ because we cancelled $p$ words of length $q$. Thus $|z| = |w_1aw_2|$ and $z \in L(G_A)$. Therefore we can take $z$ for the column instead of $w_1aw_2$. However, now the horizontal part $r^p$ is at another (strongly larger) height, i.e., we get an disconnected picture whereas $P_L$ contains only connected picture. Thus we have an contradiction. □

As a consequence of the results presented in Theorem 3.9 we get the diagram

$$\begin{tikzcd}
\mathcal{CCP}(CF) & \mathcal{PM}(CF) \\
\mathcal{CCP}(REG) \arrow[u] & \mathcal{PM}(REG) \arrow[u]
\end{tikzcd}$$

where an arrow denotes strict inclusion and two families are not connected if they are incomparable. Informally, the diagram says that picture language families generated by chain code pictures grammars or Siromoney matrix grammars seem to be completely different.

## 3.3 Decision Problems for Siromoney Matrix Grammars

In the preceding section we have seen that the families of picture languages generated by chain code pictures grammars or Siromoney matrix grammars are incomparable. Therefore we cannot transfer the results with respect to decision problems from chain code picture grammars to Siromoney matrix grammars. We have renewed to investigate the decision problems for Siromoney matrix grammars.

### 3.3.1 Classical Problems

We start with the classical problems which we only discuss for the matrix case; the modifications for quasi-matrices are left to the reader.

*Matrix version of the membership problem:*
given a matrix $M$ and a Siromoney matrix grammar $G$, decide whether or not $M \in M(G)$,

*Matrix version of the emptiness problem:*
given a Siromoney matrix grammar $G$, decide whether or not $M(G)$ is empty,

*Matrix version of the finiteness Problem:*
given a Siromoney matrix grammar $G$, decide whether or not $M(G)$ is finite,

*Picture version of the membership problem:*
given a picture $p$ and a Siromoney matrix grammar $G$, decide whether or not $p \in PM(G)$,

*Picture version of the emptiness problem:*
given a Siromoney matrix grammar $G$, decide whether or not $PM(G)$ is empty,

*Picture version of the finiteness Problem:*
given a Siromoney matrix grammar $G$, decide whether or not $PM(G)$ is finite.

**Theorem 3.10** *i) The matrix version of the membership problem is decidable for monotone Siromoney matrix grammars.*

*ii) The matrix version of the membership problem is undecidable for arbitrary Siromoney matrix grammars.*

*Proof.* i) Let the monotone Siromoney matrix grammar $G = (N_1, N_2, I, T, P_1, P_2, S, s, t, pic)$ and the $(m, n)$-matrix $M$ be given. For each column $(a_{1i}, a_{2i}, \ldots, a_{mi})^T$, $1 \leq i \leq n$, and any $A \in I$ we decide whether $w_i = a_{1i}a_{2i} \ldots a_{mi}$ is an element $L(G_A)$. If the answer is positive, then $A$ is a candidate for the $i$-th letter of the intermediate word. Let $M_i$ be the set all letters $A \in I$ for which the answer is positive. If $M_i$ is empty for some $i$, then $M$ cannot be generated. If $M_i \neq \emptyset$ for $1 \leq i \leq n$, then the intermediate word has to be in $K = M_1M_2 \ldots M_n$. Since $M_1 \subseteq I \subseteq N_2$, $K$ is a finite set. We now test all words of $K$ if they are in $L(G_1)$ for $G_1 = (N_1, I, P_1, S)$. If there is one word with a positive answer, then using this word as intermediate word, we can generate all columns and thus $M$. If no word of $K$ is in $L(G_1)$, then $M$ cannot be generated.

ii) For an arbitrary phrase structure grammar $H = (N, T, P, S)$. Let $T' = \{a' \mid a \in T\}$, and let the homomorphism $h : (N \cup T)^* \to (N \cup T')^*$ be given by $h(A) = A$ for $A \in N$ and $h(a) = a'$ for $a \in T$. We construct the grammar $H' = (N, T', P', S)$ with $P' = \{A \to h(w) \mid A \to w \in P\}$. It is easy to see that $L(H') = h(L(H))$.

Now we define the Siromoney matrix grammar

$$G = (N, T', T', T, P', \{a' \to a \mid a \in T), S, s, t, pic)$$

(where $s$,$t$ and $pic$ can chosen arbitrarily since we are only interested in matrices). It is obvious that $L(G)$ consists only of matrices which have exactly one row and that $(a_1, a_2, \ldots, a_n)$ is a matrix in $M(G)$ if and only if $a_1a_2 \ldots a_n$ is a word in $L(H)$. Thus the decidability of the matrix version of the membership problem for arbitrary Siromoney matrix grammars implies the decidability of the membership problem for arbitrary phrase structure grammars. Since the latter problem is undecidable, the former one is undecidable, too. □

For context-free Siromoney grammars, the complexity of the deciding algorithm given in the proof of Theorem 3.10 has polynomial complexity. This can be seen as follows. We have to solve the membership problems for $n$ regular word grammars and words of length $m$ (where $n$ is the number of columns and $m$ is the number of rows)) and at most $n^k$ membership problems for a context-free word grammar and words of length $n$ (where

$k$ is the number of vertical nonterminals). Because these membership problems can be decided in linear or cubic time, the algorithm is polynomial in the size of the picture and the size of the grammar. Thus we have the following statement.

**Corollary 3.11** *The matrix version of the membership problem for context-free Siromoney matrix grammars is decidable in polynomial time.* □

**Theorem 3.12** *i) The picture version of the membership problem is decidable for monotone Siromoney matrix grammars.*

*ii) The picture version of the membership problem is undecidable for arbitrary Siromoney matrix grammars.*

*Proof.* i) Let a Siromoney matrix grammar $G = (N_1, N_2, I, T, P_1, P_2, S, s, t, pic)$ and a picture $p$ be given. There is only a finite number of matrices $M$ such that $Pic(M) = p$. Let $M_p$ be the set of all these matrices. Then $p \in PM(G)$ if and only if there is an $M \in M_p$ such that $M \in M(G)$. Thus we can decide the picture version of the membership problem by finitely many applications of the matrix version of the membership problem. This implies the first statement of the theorem.

ii) We take the Siromoney grammar $G$, given in part ii) of the proof of Theorem 3.10 and complete its definition by defining $s$, $t$ and $pic$. Let $T = \{a_1, a_2, \dots a_k\}$. Then we set $s = 1$ and $t = k - 1$, and $pic(a_i) = bccp(\uparrow u^{i-1} \downarrow r)$, i.e., $pic(a_i)$ consists of a vertical unit line in height $i - 1$. Thus a generated picture consists of a sequence of unit lines. From this we can uniquely determine the word over $T$. If we can decided the picture version of the membership problem for $G$, then we can decided the matrix version of the membership problem for $G$ which is undecidable. □

However, with respect to the complexity we have a large increase if we go from the matrix version to the picture version. More precisely, we have the following theorem.

**Theorem 3.13** *The picture version of the membership problem for regular Siromoney matrix grammars is* **NP***-complete.*

*Proof.* The problem is in **NP**. This can be seen as follows. For a picture $p$, we can guess a matrix $M$ with $Pic(M) = p$ and check in polynomial time whether or not $M \in M(G)$. The answer is positive for at least one $M$ if and only if $p \in PM(G)$ holds.

We now give a reduction to the satisfiability problem. Let

$$K = D_1 \wedge D_2 \wedge \cdots \wedge D_m$$

be a formula of the propositional calculus in conjunctive normal form over $n$ variables, i.e., each $D_i$, $1 \leq i \leq m$, is an expression $D_i = x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_{n(i)}}$. For $1 \leq i \leq m$ and $1 \leq j \leq n$, we now define

$$a_{i,j} = \begin{cases} 1 & x_j \text{ appears in } D_i \\ -1 & \neg x_j \text{ appears in } D_i \\ 0 & \text{neither } x_j \text{ nor } \neg x_j \text{ appear in } D_i \end{cases}$$

Moreover, we set

$$M_K = \begin{array}{cccccc} \# & \# & \# & \dots & \# & \# \\ \# & a_{1,1} & a_{1,2} & \dots & a_{1,n} & \# \\ \# & a_{2,1} & a_{2,2} & \dots & a_{2,n} & \# \\ & \vdots & & & \vdots & \\ \# & a_{m,1} & a_{m,2} & \dots & a_{m,n} & \# \\ \# & \# & \# & \dots & \# & \# \end{array}$$

As an example, for the expression

$$(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_3 \vee \neg x_4) \vee (x_2 \vee x_3 \vee x_4),$$

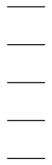
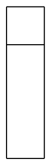
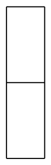
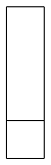
we get the matrix

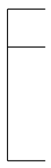
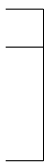
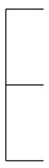
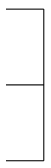
$$\begin{array}{cccccc} \# & \# & \# & \# & \# & \# \\ \# & 1 & \text{-1} & 1 & 0 & \# \\ \# & \text{-1} & 0 & 1 & \text{-1} & \# \\ \# & 0 & 1 & 1 & 1 & \# \\ \# & \# & \# & \# & \# & \# \end{array}$$

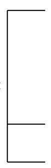
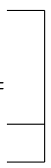
Now we construct the regular Siromoney matrix grammar

$$G = (N_1, N_2, \{A, B, C\}, \{\#, 1, 1_r, 1_l, 0, 0_r, 0_l, -1, -1_r, -1_l\}, P_1, P_2, S, 4, 1, pic)$$

where

$pic(\#) =$  , $pic(1) =$  , $pic(0) =$  , $pic(-1) =$  ,

$pic(1_l) =$  , $pic(1_r) =$  , $pic(0_l) =$  , $pic(0_r) =$  ,

$pic(-1_l) =$  , $pic(-1_r) =$  ,

and the nonterminal sets and production sets are chosen such that

$$\begin{aligned} L(G_1) &= \{C\}\{A, B\}^*\{C\}, \\ L(G_A) &= \{\#\}\{0_r, 0_l, 1, -1_r, -1_l\}^*\{1\}\{0_r, 0_l, 1, -1_r, -1_l\}^*\{\#\}, \\ L(G_A) &= \{\#\}\{0_r, 0_l, , 1_r, 1_l, -1\}^*\{-1\}\{0_r, 0_l, , 1_r, 1_l, -1\}^*\{\#\}, \\ L(G_C) &= \{\#\}^*. \end{aligned}$$

Note that any matrix in $M(G)$ starts and ends with a column consisting of #'s only (originating from the $C$'s) and its first and last row consist of #'s only. Moreover, any column originating from $A$ or $B$ contains at least one 1 and no -1 or contains at least one $-1$ and no 1, respectively. Let

$$U = \begin{array}{cccccc} \# & \# & \# & \cdots & \# & \# \\ \# & a'_{1,1} & a'_{1,2} & \cdots & a'_{1,n} & \# \\ \# & a'_{2,1} & a'_{2,2} & \cdots & a'_{2,n} & \# \\ & \vdots & & & \vdots & \\ \# & a'_{m,1} & a'_{m,2} & \cdots & a'_{m,n} & \# \\ \# & \# & \# & \cdots & \# & \# \end{array}$$

be a matrix of $M(G)$. We associated with it an expression $K'$ in conjunctive normal form and an assignment $\alpha$ as follows: $K(U)$ consists of $m$ disjunctions $D'_1, D'_2, \ldots, D'_m$ over $n$ variables $x_1, x_2, \ldots, x_n$; for $1 \leq i \leq m$,
– $D_i$ contains $x_j$ if and only if $a_{i,j} \in \{1, 1_r, 1_l\}$,
– $D_i$ contains $\neg x_j$ if and only if $a_{i,j} \in \{-1, -1_r, -1_l\}$,
– $D_i$ contains neither $x_j$ nor $\neg x_j$ if and only if $a_{i,j} \in \{0_r, 0_l\}$,
and
– $\alpha(x_j) = true$ if and only if the $(j+1)$-column contains a 1,
– $\alpha(x_j) = false$ if and only if the $(j+1)$-column contains a 1.
This means that we choose $A$ or $B$ as the $(j+1)$-th letter of the intermediate word, if we assign *true* or *false* to the $j$-th variable, respectively. We write $U = M_{K',\alpha}$. It is clear that $M(G)$ is the set of all matrices $M_{K',\alpha}$ where $K'$ is an expression in conjunctive normal form and $\alpha$ is an assignment.

We now discuss whether or not $Pic(M_K)$ is contained in $PM(G)$. By definition of *pic*, $Pic(M_K)$ contains exactly $n+1$ vertical lines in any height $k$ with $-5 \geq k \geq 4m+4$. If $Pic(M_K) \in PM(G)$, then $Pic(M_K) = Pic(M_{K',\alpha})$ for some $K'$. Since the horizontal lines of $Pic(M_K)$ and $Pic(M_{K',\alpha})$ coincide, we get $K = K'$. Obviously, $Pic(M_{K,\alpha})$ contains less than $n+1$ vertical lines in height $k$ for $-4k+1 \geq k \geq -4k+m$, if the $(k+1)$-th row of $M_{K,\alpha}$ does contain neither 1 nor $-1$ (because any remaining indexed letter contributes at most one vertical line, and since the row contains $n$ letters we get at most $n$ vertical lines). This implies $Pic(M_K) \neq Pic(M_{K,\alpha})$. Therefore any row of $M_{K,\alpha}$ has to contain at least one 1 or one $-1$. Assume it contains a 1, say $a'_{i,j} = 1$. By construction, $D_i$ contains the variable $x_j$ and $\alpha(x_j) = true$. Thus we obtain that $D_j$ also gets the value *true*. Moreover, using letters with index $l$ to the left and letters with index $r$ to the right, we get $n+1$ vertical lines. Analogously, the existence of a $-1$ leads to the value *true* for $D_i$ and $n+1$ vertical lines, too. Thus $Pic(M_K)$ is in $PM(G)$ if and only if, $D_i$ gets *true* for $1 \leq i \leq m$, i.e., $K$ is satisfiable. Since the satisfiability problem for expression (in conjunctive normal form) is **NP**-complete, the picture version of the membership problem for regular Siromoney grammars is **NP**-complete, too. □

**Theorem 3.14** *The matrix and picture versions of the emptiness problem are decidable for context-free Siromoney matrix grammars, and they are undecidable for monotone Siromoney matrix grammars.*

*Proof.* It is obvious that, for a given Siromoney grammar $G$, $PM(G)$ is empty if and only if $M(G)$ is empty. Therefore it is sufficient to show that the matrix version of the emptiness problem for context-free Siromoney matrix grammars is decidable.

Let $G = (N_1, N_2, I, T, P_1, P_2, S, s, t, pic)$ be a context-free Siromoney grammar. We define the homomorphism $h : T^* \to \{a\}^*$ where $a$ is a new symbol. Moreover, for any intermediate symbol $A$, we set $L_A = h(L(G_A))$. Then $a^n \in L_A$ if and only if $L(G_A)$ contains a word of length $n$. For a set $R \subseteq I$, we set $L_R = \bigcap_{A \in R} L_A$. Then $a^n \in L_R$ if and only if we can generate from any word of $R^*$ a matrix with $n$ columns. Thus $M(G)$ is empty if, for any $R \subset I$, either $L_R$ is empty or $L_R$ is non-empty and $L(G_1) \cap R^*$ is empty. By the closure properties of $\mathcal{L}(REG)$ and $\mathcal{L}(CF)$ (see Theorem 1.12), for any $R \subseteq I$, $L_R$ is regular and $L(G_1) \cap R^*$ is context-free. Therefore we can decide the above emptiness problems for any $R \subseteq I$. Since there exist only finitely many sets $R$ the matrix version of the emptiness problem for context-free Siromoney matrix grammars is decidable. □

**Theorem 3.15** *The matrix and picture versions of the finiteness problem are decidable for context-free Siromoney matrix grammars, and they are undecidable for monotone Siromoney matrix grammars.*

*Proof.* Again, for a given Siromoney grammar $G$, $PM(G)$ is finite if and only if $M(G)$ is finite. Therefore it is sufficient to show that the matrix version of the finiteness problem for context-free Siromoney matrix grammars is decidable.

For a set $R \subset I$, we construct the sets $L_R$ and $L(G_1) \cap R^*$ as in the proof of Theorem 3.14. It is easy to see that $M(G)$ is finite if and only if there is no set $R \subseteq I$ such that
– $L_R$ is non-empty and $L(G_1) \cap R^*$ is infinite, or
– $L_R$ is infinite and $L(G_1) \cap R^*$ is non-empty.

Obviously, these conditions are decidable. □

### 3.3.2 Decision Problems related to Submatrices and Subpictures

As in the case of chain code pictures we now discuss problems related to subpictures (and submatrices, too) and we shall use these results later to get some results on the decidability of geometrical properties. We start with the problem of existence of a given submatrix/subpictures in a matrix/picture languages generated by a Siromoney matrix grammar.

*Submatrix Problem:*
Given: Siromoney matrix grammar $G$, matrix $M$
Question: Is there a matrix $M' \in M(G)$ such that $M$ is a submatrix of $M'$ ?

*Subpicture Problem:*
Given: Siromoney matrix grammar $G$, chain code picture $p$
Question: Is there a matrix $M' \in M(G)$ such that $p$ is a subpicture of $Pic(M')$?

**Theorem 3.16** *For context-free Siromoney matrix grammars and arbitrary matrices, the submatrix problem is decidable in polynomial time.*

*Proof.* Let $G = (N_1, N_2, I, T, P_1, P_2, S, s, t, pic)$ be a context-free Siromoney matrix grammar and $M$ an $(m, n)$-matrix. We first show that there is a constant $K$ such that, if $M$ is a submatrix of some matrix of $M(G)$, then $M$ is a submatrix of some matrix in $M(G)$ which has at most $2K + m$ rows.

Let $n = \#(N_2)$. We set $K = (n+1)(nn!+1)$. Assume that $M$ is a submatrix of the $(m',n')$- matrix $M' \in M(G)$ and that $m' > 2K+m$. Let the $(m,n)$-matrix $M$ be in the rows $r+1, r+2, \ldots, r+m$. Then there are $r$ rows above $M$ and $m'-m-r$ rows below $M$. Obviously, $r > K$ or $m'-m-r > K$. We only discuss the former case $r > K$; the consideration for the other case are analogous.

Let $M' = (a_{i,j})_{(m',n')}$. Let $1 \leq j \leq n'$. Any word $a_{1,j}a_{2,j}\ldots a_{m',j}$ belongs to a regular languages $L(G_{A_j})$ with the set $N_2$ of nonterminals ($A_j$ is the intermediate letter from which the $j$-th column comes from). For $0 \leq i \leq nn!$, we consider the subwords

$$w_{i,j} = a_{i(n+1)+1,j}a_{i(n+1)+2,j}\ldots a_{(i+1)(n+1),j}$$

of length $n+1$. By the pumping lemma from any $w_{i,j}$ we can cancel a subword of some length $n_{i,j}$ with $1 \leq n_{i,j} \leq n$. Because we have $nn!+1$ lengths $n_{i,j}$ for any $j$, there is one length, say $s_j$, which occurs at least $n!$ times. Now we shorten $n!/s_j$ words $w_{i,j}$ by $s_j$ letters, which results in a column which has the length $m' - (n!/s_j) \cdot s_j = m' - n!$. We do this procedure for any $j$ and get a matrix $M'' \in M(G)$ which contains $M$ and has only $m' - n!$ rows where $m' - n! < m'$. We continue this process until the number of rows is smaller than $2K+m$.

The algorithm to decide the existence of a submatrix works as follows. First build the set $U$ of all $(u,n)$-matrices $M'$ such that $u \leq 2K+m$. For each such matrix $M'$ we determine the set $I_{M'}$ of all words $A_1A_2\ldots A_n \in I^*$ such that, for $1 \leq j \leq n$, we can generate from $A_j$ the $j$-th column of $M'$. To do this we determine for a given column $w^T$ all intermediate letters $A$ such that $w \in L(G_A)$ (which is possible since the membership problem for regular grammars is decidable in linear time). Let $I' = \bigcup_{M' \in U} I_{M'}$. Obviously, $I'$ is a finite set and with any $Y \in I'$ we can associate a finite set $U_Y$ containing those numbers $r$ such that $m \leq r \leq 2K+m$ and $Y$ generates an $(r,n)$-matrix which contains $M$. For $m \leq r \leq 2K+m$, let $I(r)$ be the set of all intermediate letters $B$ with $L(G_B)$ contains at least one word of length $r$. Clearly, $I(r)$ can be determined in polynomial time. Now we determine for an $Y \in I'$ and a number $r \in U_Y$ whether or not $L(G_1) \cap I(r)^*\{Y\}I(r)^*$ is empty, which can be done in polynomial time. It is easy to see that we get non-emptiness for at least one pair $(Y,r)$ if and only if $M$ is a submatrix of some matrix of $M(G)$. □

**Theorem 3.17** *For context-free Siromoney matrix grammars and arbitrary pictures, the subpicture problem is decidable.*

*Proof.* Let $p$ be a chain code picture of width $p$ and height $q$. Let $p = as+b$ and $q = a't+b'$ where $b < s$ and $b' < t$. Then $p$ is a subpicture of some matrix $M$ with at most $a+2$ columns and at most $a'+2$ rows (since a subpicture of width $as$ and height $a't$ corresponds to a matrix $M$ and the remaining parts of $p$ are subpictures from the row and column before and the row and column behind $M$). We build all $(a'+2, a+2)$-matrices whose associated pictures have $p$ as a subpicture and solve for all these matrices the submatrix problem. If we get one positive answer, then $p$ is a subpicture of some $Pic(M')$ where $M' \in M(G)$; otherwise, $p$ is not such a subpicture. □

**Theorem 3.18** *The subpicture problem is NP-complete for regular Siromoney matrix grammars.*

*Proof.* We repeat the proof of Theorem 3.13. It is easy to see that $Pic(M_K)$ is a subpicture of some $Pic(M_{K',\alpha})$ if and only if $Pic(M_K) = Pic(M_{K',\alpha})$. Hence for the considered picture and grammar, the subpicture problem and the membership problem coincide. □

We now consider the universal submatrix/subpicture problem. Since we are here interested in those matrices/pictures which are contained as a submatrix/subpicture in any matrix, it is of interest to consider the sets of matrices/pictures which contain or do not contain a certain matrix/picture.

**Definition 3.19** *For a matrix language $L$ and a matrix $M$ we set*

$$\begin{aligned} L_M &= \{M' \mid M' \in M(G),\ M \textit{ is a submatrix of } M'\},\\ L_{\neg M} &= \{M' \mid M' \in M(G),\ M \textit{ is not a submatrix of } M'\}. \end{aligned}$$

For a fixed matrix $M$, by Definition 3.19, we get unary operations $O_M$ and $O'_M$ which map any matrix language $L$ on $L_M$ and $L_{\neg M}$, respectively. We now present some nonclosure/closure results under these operations.

**Lemma 3.20** *There is a language $L \in \mathcal{M}(REG)$ and matrices $M$ and $M'$ such that $L_M \notin \mathcal{M}(CF)$ and $L_{\neg M'} \notin \mathcal{M}(CF)$.*

*Proof.* We consider the regular Siromoney matrix grammar

$$G = (\{S\}, \{A,B\}, \{A\}, \{a,b\}, \{S \to AA\}, \{A \to aA, A \to bB, B \to aB, B \to a\}, S, s, t, pic)$$

(we omit a definition of $s$, $t$ and $pic$ since we are only interested in the matrices) and take $L = M(G)$. It is easy to see that any matrix of $M(G)$ consists of exactly two columns and that each column contains exactly one $b$ (the remaining letters of the columns are $a$'s). Moreover, we choose $M = bb$, i.e., it consists of one row which contains exactly two $b$'s. Then $L_M$ is the set of all matrices which consist of two identical columns (since one row consists of two $b$'s and all other rows of two $a$'s), i.e., any matrix of $L_M$ has the form $uu$ for some column $u$ which contains exactly one $b$.

Let us assume that $L_M \in \mathcal{M}(CF)$. Then $L_M = M(H)$ for some context-free Siromoney matrix grammar $H$. We consider the matrix $uu \in M(G) = L_M$ of two columns of length $2n+1$ and containing the row $bb$ as the $(n+1)$-th row. If we choose $n$ sufficiently large, then we can analogous to the proof of Theorem 3.8, part iii) shorten the upper part $a^n$ of the first column by $k$ letters for some $k > 0$ and extend the lower part $a^n$ of the first column by $a^k$. Let $v$ be this column. Analogously to the proof of Theorem 3.8, part iii), we can show that the matrix $vu$ is in $M(H)$, too. However, the matrix $vu$ is not in $L_M$ because the first column $v$ contains $b$ in the row $n-k+1$ whereas the second column $u$ has $b$ in the $(n+1)$-th row. Thus we have a contradiction to $M(H) = L_M$.

We choose $M' = ba$. Since any column of $L$ contains exactly one letter $b$, the absence of $ba$ in a matrix is equivalent to the presence of $bb$. Thus $L_{\neg M'} = L_M$. Since $L_M \notin \mathcal{M}(CF)$, $L_{\neg M'}$ is also not contained in $\mathcal{M}(CF)$. □

**Lemma 3.21** *For $X \in \{REG, CF\}$, any Siromoney matrix language $L \in \mathcal{M}(X)$ and any $(m,1)$-matrix $M$, the languages $L_M$ and $L_{\neg M}$ are in $\mathcal{M}(X)$.*

*Proof.* Let $G = (N_1, N_2, I, T, P_1, P_2, S, s, t, pic)$ be an $X$ Siromoney matrix grammar such that $L = M(G)$ and let $M$ be a $(m, 1)$-matrix consisting of the column $w^T$. Then $M$ is a submatrix of some matrix $M' \in M(G)$ if and only if there is an intermediate word $A_1A_2 \dots A_n$ and $xwy \in L(A_i)$ for some $i$, $1 \leq i \leq n$, some $x \in T^*$ and some $y \in T^*$ ($M'$ is a matrix generated from $A_1A_2 \dots A_n$), or equivalently, $L(G_{A_j}) \cap T^*\{w\}T^*$ is non-empty. Moreover, $M$ is not a submatrix of any $M' \in M(G)$ if and only if, for all intermediate words $A_1A_2 \dots A_n$, we have $L(G_{A_i}) \subseteq T^* \setminus T^*\{w\}T^*$ for all $i$, $1 \leq i \leq n$. For $A \in I$, we set

$$L_A = L(G_A) \cap T^*\{w\}T^* \text{ and } L'_A = L(G_A) \cap (T^* \setminus T^*\{w\}T^*).$$

We note that $L_A$ and $L'_A$ are regular languages by the closure properties of $\mathcal{L}(REG)$ (see Theorem 1.12).

Let $I' = \{A' \mid A \in I\}$ and the substitution $\sigma : I \to I \cup I'$ be given by $\sigma(A) = \{A, A'\}$. We now construct the Siromoney matrix grammar $H = (N'_1, N'_2, I \cup I', T, P'_1.P'_2, S, s, t.pic)$ such that

$$L(H_1) = \sigma(L(G_1)) \cap I^*I'I^*, \ L(H_A) = L(G_A) \text{ for } A \in I, \text{ and } L(H_{A'}) = L_A \text{ for } A \in I.$$

It is easy to see that $M(H) = L_M$. Since $L(H_1)$ is context-free or regular, if $G_1$ is context-free or regular, respectively, $H$ is of the same type as $G$. Thus $L_M \in \mathcal{M}(X)$, too.

For $L_{\neg M}$ we modify $H$ by the settings

$$L(H_1) = L(G_1) \quad \text{and} \quad L(H_A) = L'_A \text{ for } A \in I$$

and get $M(H) = L_{\neg M}$. □

**Lemma 3.22** *For $X \in \{REG, CF\}$, any Siromoney matrix grammar $G$ of type $X$ such that $L(G_A)$ is finite for any $A \in I$ and any matrix $M$, the languages $M(G)_M$ and $M(G)_{\neg M}$ are in $\mathcal{M}(X)$*

*Proof.* Let $G = (N_1, N_2, I, T, P_1, P_2, S, s, t, pic)$ be an $X$ Siromoney matrix grammar such that, for any $A \in I$, $L(G_A)$ is finite. Let $M$ be an arbitrary $(m, n)$-matrix. Without loss of generality we can assume that $L(G_A)$ is a singleton, i.e., $L(G_A) = \{w_A\}$ for some word $w_A$ (take $A_1, A_2, \dots A_k$ instead of $A$ with $L(G_A) = \{w_1, w_2, \dots, w_k\}$ and choose the productions in such a way that $L(G_{A_i}) = \{w_i\}$ for $1 \leq i \leq k$).

Let $S_M$ be the set of all words $A_1A_2 \dots A_n \in I^*$ such that $|w_{A_1}| = |w_{A_i}|$ for $2 \leq i \leq n$ and $M$ is a submatrix of the matrix $w^T_{A_1}\ w^T_{A_2}\ \dots\ w^T_{A_n}$. Obviously, $S_M$ is finite. We now construct the Siromoney matrix grammars $H$ and $H'$ with the set $I$ of intermediates such that

$$\begin{aligned} &L(H_1) = L(G_1) \cap I^*S_MI^*, \quad L(H'_1) = L(G_1) \cap (I^* \setminus I^*S_MI^*), \\ &L(H_A) = L(H'_A) = \{w_A\}. \end{aligned}$$

It is easy to see that $M(H) = L_M$ and $M(H') = L_{\neg M}$. By the closure properties of the families $\mathcal{L}(CF)$ and $\mathcal{L}(REG)$, $L(H_1)$ and $L(H'_1)$ are of the same type as $L(G_1)$. Thus $L_M \in \mathcal{M}(X)$ and $L_{\neg M} \in \mathcal{M}(X)$. □

We now discuss the problems of the existence of universal submatrices/subpictures which can be given as follows.

*Universal Submatrix Problem:*
Given: Siromoney matrix grammar $G$, matrix $M$
Question: Is $M$ a submatrix of any $M' \in M(G)$?

*Universal Subpicture Problem:*
Given: Siromoney matrix grammar $G$, picture $p$
Question: Is $p$ a subpicture of $Pic(M')$ for any $M' \in M(G)$?

**Theorem 3.23** *For context-free Siromoney matrix grammars and arbitrary $(m,1)$-matrices, the universal submatrix problem is decidable.*

*Proof.* Let the context-free Siromoney matrix grammar $G$ and a $(m,1)$-matrix $M$ be given. Then it is obvious that $M$ is a universal submatrix of $M(G)$ if and only if the matrix language $M(G)_{\neg M}$ of all matrices of $M(G)$ which do not contain $M$ as a submatrix is empty. In the proof of Lemma 3.21 we have constructed a context-free Siromoney matrix grammar $H$ generating $L_{\neg M}$. Therefore we have only to decide whether $L(H)$ is empty. This can be decided by Theorem 3.14. □

**Theorem 3.24** *For context-free Siromoney matrix grammars such that $L(G_i)$ is finite for any $i \in I_1$ and arbitrary matrices, the universal submatrix problem is decidable.*

*Proof.* We can give a proof which is analogous to that of Theorem 3.23 using Lemma 3.22. □

Without a proof we state the following theorems which give the undecidability in the general case.

**Theorem 3.25** *For regular Siromoney matrix grammars and arbitrary matrices (with at most two columns), the universal submatrix problem is undecidable.* □

**Theorem 3.26** *For regular Siromoney matrix grammars, the universal subpicture problem is undecidable.* □

### 3.3.3 Decidability of geometric properties

Before we present the decidability results we add a further "geometric" property.

**Definition 3.27** *We say that a chain code picture $p$ is edge-colourable by $k$ colours, if there is a mapping from the set of unit lines of $p$ to $\{1, 2 \ldots, k\}$ such that any two different unit lines of $p$ which intersect in a node are mapped to different numbers.*

The classical interpretation of edge-colourability is that we map the edges to $k$ colours instead of $k$ numbers. Then it is required that edges with a common node have different colours. Obviously, any chain code picture can be edge-coloured by 4 colours. This colouring is given by the mapping which maps
– any vertical unit line whose lower node has an odd value in the second component to the first colour,
– any vertical unit line whose lower node has an even value in the second component to

the second colour,
– any horizontal unit line whose left node has an odd value in the first component to the third colour,
– any horizontal unit line whose left node has an even value in the first component to the fourth colour.
Again, if $p$ is a connected picture the edge-colourabilty by 1 colour is only possible if the picture consists of a single unit line.

If a picture is connected, then the notions of a simple closed curve and a 2-regular graph coincide. However, the pictures generated by a Siromoney matrix grammar are not necessarily connected (in contrast to chain code picture grammars) which makes it necessary to consider the notion of $k$-regularity. However it remains true, that only the values $k = 1$ and $k = 2$ are possible for chain code pictures (see the remark before Theorem 2.29).

**Theorem 3.28** *It is undecidable for regular Siromoney grammars whether or not $PM(G)$ contains*
*i) a connected picture,*
*ii) a 2-regular picture,*
*iii) a Eulerian picture,*
*iv) a Hamiltonian picture,*
*v) a tree.*

*Proof.* i) We reduce the problem to the (undecidable) emptiness problem for monotone grammars or equivalently for linearly bounded automata $A = (Q, \{0, 1, \#\}, q_0, F, \delta)$. We consider the automaton in a form nearly related to that we have used in the proof of Theorem 2.27, i.e., that the automaton has an input alphabet consisting of $0, 1$ and the endmarker $\#$, moves alternately from left to right and right to left between the two markers, makes a step without move reading an endmarker $\#$ and changes the direction. We denote the change of the direction by $LR$ and $RL$, the reading of $x$ and writing of $y$ by $r_x w_y$ and use an arrow above in order to present the direction which the head has, i.e., we use $\overrightarrow{r_x w_y}$ and $\overleftarrow{r_x w_y}$. Let

$$\begin{aligned} Right &= \{\overrightarrow{r_i w_j} \mid i, j \in \{0, 1\}\}, \\ Left &= \{\overleftarrow{r_i w_j} \mid i, j \in \{0, 1\}\}, \\ I &= Right \cup Left \cup \cup\{LR, RL\}, \end{aligned}$$

We consider the regular Siromoney matrix grammar

$$G = (Q \times \{r, l\}, N_2, I \cup \{A, E\}, T, P_1, P_2, (q_0, r), 3, 6, pic)$$

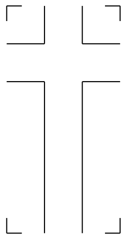
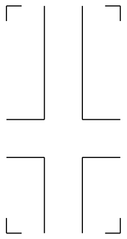
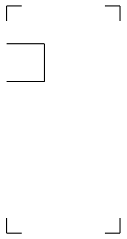
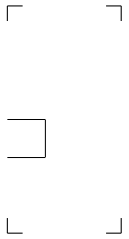
where

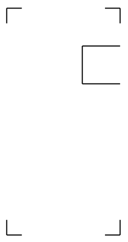
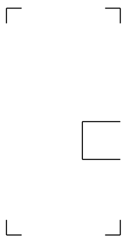
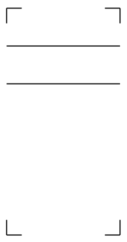
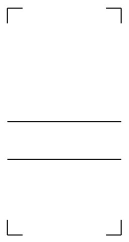
$$\begin{aligned} T &= \{v, x, y, y', y'', y''', z, a_o, a_1, b_0, b_1, b_0', b_1', c_0, c_1, d_0, d_1\}, \\ P_1 &= \{(q, r) \to \overrightarrow{r_i w_j}(q', r) \mid (q', j, r) \in \delta(q, i)\} \cup \{(q, r) \to RL(q', l) \mid (q', \#, l) \in \delta(q, \#)\} \\ &\quad \cup\{(q, l) \to \overleftarrow{r_i w_j}(q', l) \mid (q', j, l) \in \delta(q, i)\} \\ &\quad \cup\{(q, r) \to RL(q', l) \mid (q', \#, l) \in \delta(q, \#)\} \cup \{(q, l) \to \lambda \mid q \in F\}, \end{aligned}$$

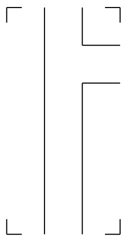
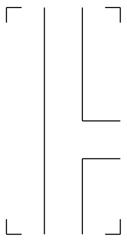
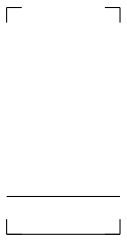
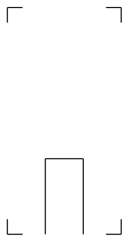
$N_2$ and $P_2$ are chosen in such a way that

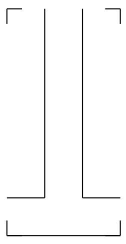
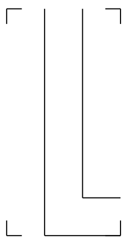
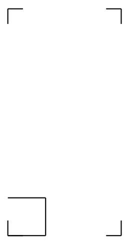
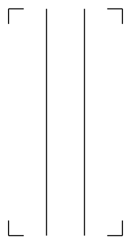
$$\begin{aligned}
L(G_A) &= \{\{x\}(\{d_0,d_1\}\{z\})^+\{y'\}, \\
L(G_E) &= \{\{v\}(\{v\}\{b_0,b_1\})^+\{y''\}, \\
L(G_{RL}) &= \{\{x\}(\{z\}\{a_0,a_1\})^+\{y\}, \\
L(G_{LR}) &= \{\{x\}(\{a_0,a_1\}\{z\})^+\{y\}, \\
L(G_{\overrightarrow{r_i w_j}}) &= \{\{v\}(\{v\}\{c_0,c_1\})^*\{b_i b'_j\}(\{c_0,c_1\}\{v\})^*\{w'\} \text{ for } i,j\in\{0,1\}, \\
L(G_{\overleftarrow{r_i w_j}}) &= \{\{v\}(\{v\}\{c_0,c_1\})^*\{b'_j b_i\}(\{c_0,c_1\}\{v\})^*\{w'\} \text{ for } i,j\in\{0,1\},
\end{aligned}$$

and $pic(v)$ is the empty picture and for the remaining letters, $pic$ is given by

$pic(a_0) =$  , $pic(a_1) =$  , $pic(b_0) =$  , $pic(b_1) =$  ,

$pic(b'_0) =$  , $pic(b'_1) =$  , $pic(c_0) =$  , $pic(c_1) =$  ,

$pic(d_0) =$  , $pic(d_1) =$  , $pic(w) =$  , $pic(x) =$  ,

$pic(y) =$  , $pic(y') =$  , $pic(y'') =$  , $pic(z) =$  ,

We first note that

$$\begin{aligned}
L(G_1) \;=\; \{ & Au_1(RL)v_1(RL)u_2(RL)v_2(LR)\ldots u_{n-1}(RL)v_{n-1}(LR)u_n(RL)E \mid \\
& u_i \in Right^+ \text{ for } 1\le i\le n,\ v_j\in Left^+ \text{ for } 1\le j\le n-1\}.
\end{aligned}$$

Let $w = u_1(RL)v_1(RL)u_2(RL)v_2(LR)\ldots u_{n-1}(RL)v_{n-1}(LR)u_n(RL)$. This word describes the work of linearly bounded automaton if and only if the following conditions hold:

1. $|u_k| = |v_k| = |u_{k+1}$ for $1 \leq k \leq n-1$,

2. for $1 \leq k \leq n-1$, if the $m$-th letter of $u_k^R$ is $\overrightarrow{r_i w_j}$, then the $m$-th letter of $v_k$ is $\overleftarrow{r_j w_x}$ for some $i, j, x \in \{0, 1\}$,

3. for $1 \leq k \leq n-1$, if the $m$-th letter of $v_k$ is $\overleftarrow{r_i w_j}$, then the $m$-th letter of $u_{k+1}$ is $\overrightarrow{r_j w_x}$ for some $i, j, x \in \{0, 1\}$,

Now assume that there is a $(m, n)$-matrix $M \in M(G)$ such that $Pic(M)$ is a connected picture. A typical such picture is shown in Figure 3.3. All the vertical words are of a form that the have a start and end letter and in between a certain number $r$ of words of length 2. Thus $m = 2r+2$. Now we regard the picture that is derived from the horizontal subword $v_l LR u_{l+1}$ for some $l$, $1 \leq l \leq n-1$. In the second row of the columns generated from $LR$ we find the symbol $a_{i_1}$ with $i_1 \in \{0, 1\}$. In the neighbouring right column the symbol $b_{i_1}$ has to occur and in the neighbouring left column stands $b'_{i_1}$. Otherwise the picture is not connected. This means that the first letter of $u_{l+1}$ is $\overrightarrow{r_{i_1} w_x}$ and the last letter of $v_l$ is $\overleftarrow{r_y w_{i_1}}$ for some $x, y \in \{0, 1\}$. It is easy to see that the remaining columns contain in their second rows only $v$'s. This means that the second condition for simulating a linearly bounded automaton is satisfied for $r = 1$. By induction on the rows of the columns one can analogously prove that this condition holds for any $m$. Therefore we also get $|u_{l+1}| = |v_l|$. Analogously, looking on the columns around $RL$ we get that the third condition is satisfied and $|u_l| = |v_l|$, too. Hence the first conditions also holds.

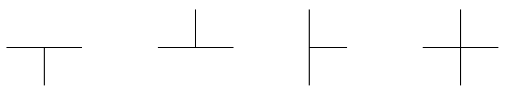
Thus $Pic(M)$ is connected if and only if $M$ corresponds to a run of a linearly bounded automaton, or equivalently, $PM(G)$ contains a connected picture if and only if the language accepted by the linearly bounded automaton is not empty. Since we cannot decide the emptiness of the language accepted by an linearly bounded automaton, the existence of a connected picture in $PM(G)$ is undecidable.

ii), iii) and iv) can be shown analogously because, for the Siromoney matrix grammar considered in part i), a picture is 2-regular or Eulerian or Hamiltonian if and only if it is connected.

iv) For the Siromoney matrix grammar considered in part i), we change the mapping to the pictures in such a way that we cancel the lower lines in the pictures of $w$, $y$, $y'$ and $y''$. Then it is easy to see that the generated picture of the new grammar is a tree if the corresponding picture of the original grammar is connected. □

**Theorem 3.29** *It is decidable for regular Siromoney grammars whether or not all picture of $PM(G)$ are*
*i) $k$-regular pictures for $k \geq 1$,*
*ii) edge-colourable by $k$ colours for $k \geq 1$.*

*Proof.* i) Obviously, a picture $p$ is 2-regular, if and only if it does not contain a node with a degree at most 3 or with degree 1 if and only it does not contain the following subpictures



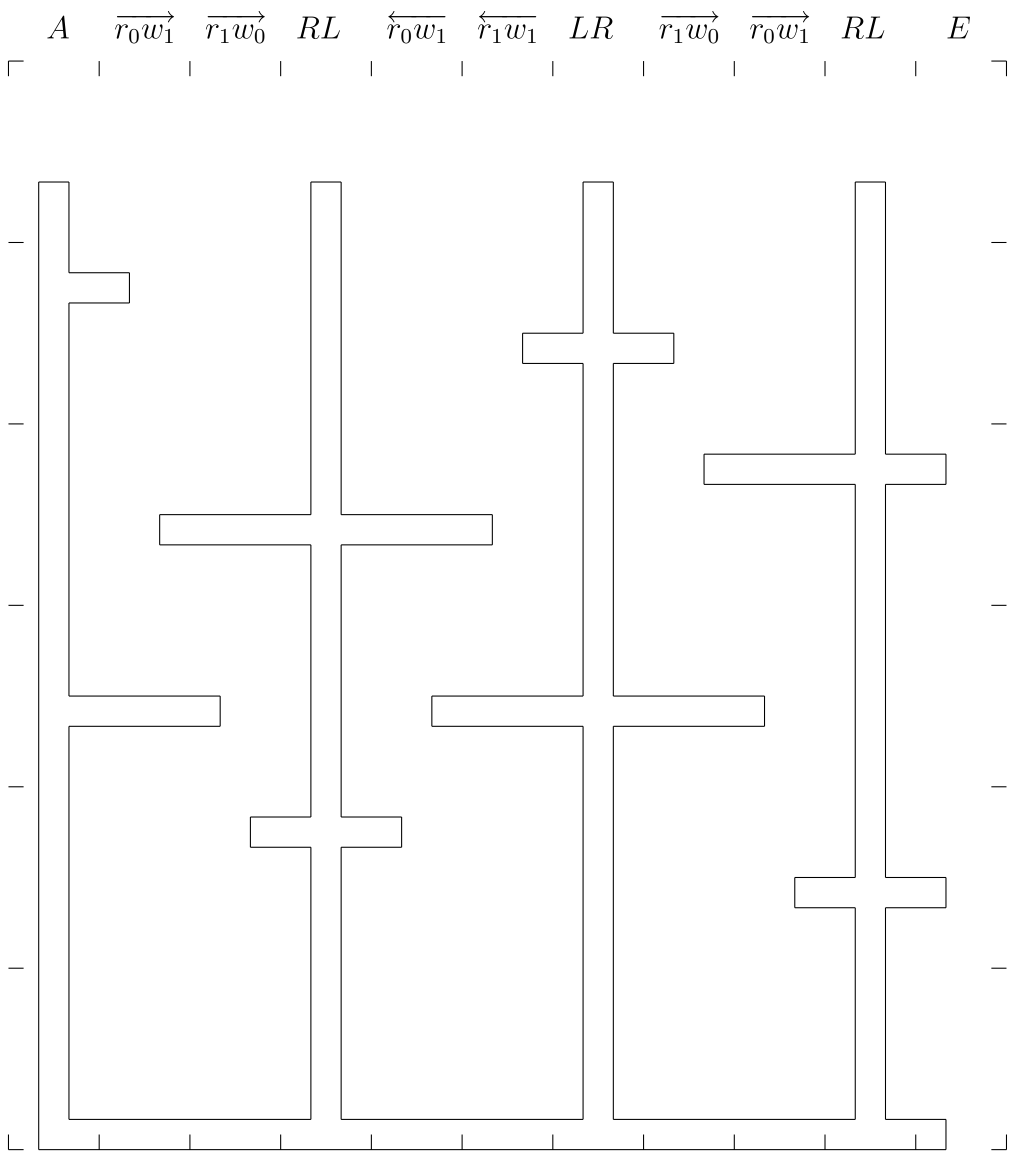
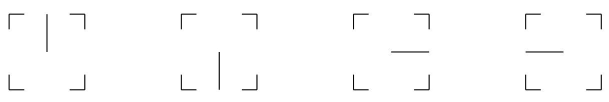
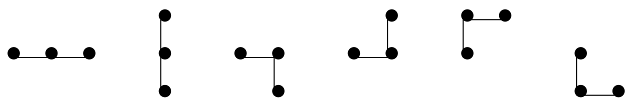

$A$
$\overrightarrow{r_0w_1}$
$\overrightarrow{r_1w_0}$
$RL$
$\overleftarrow{r_0w_1}$
$\overleftarrow{r_1w_1}$
$LR$
$\overrightarrow{r_1w_0}$
$\overrightarrow{r_0w_1}$
$RL$
$E$

Figure 3.3: A typical connected picture generated by the Siromoney matrix grammar $G$ given in the proof of Theorem 3.28; the additionally given angles show the columns and rows; above the intermediate word is given



Therefore, all pictures of $PM(G)$ are 2-regular if and only if the given pictures are not subpictures of $PM(G)$. By Theorem 3.17, the 2-regularity of all pictures in $PM(G)$ can be decided.

Analogously, all pictures of $PM(G)$ are 1-regular if and only if the pictures

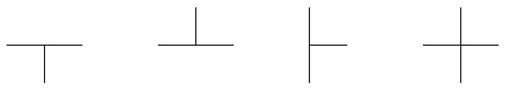


are not subpictures of $PM(G)$.

Let $k \geq 3$. Since no picture is $k$-regular, there is no grammar such that all generated pictures are $k$-regular.

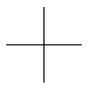
ii) Since a picture is edge-colourable by 1 colour if and only if it is 1-regular, we get the statement for $k = 1$ from part i).

A picture is edge-colourable by 2 colours if and only if it does not contain any of the pictures



Thus we can give a proof analogous to that for 2-regularity.

A picture is edge-colourable by 3 colours if and only if it does not contain the picture



Since any picture is edge-colourable by 4 colours, all pictures generated by a grammar are edge-colourable by 4 colours. □

Without proof we add the following result.

**Theorem 3.30** *It is undecidable for regular Siromoney grammars whether or not all picture of $PM(G)$ are*
*i) connected pictures,*
*ii) Eulerian pictures,*
*iii) Hamiltonian pictures.* □

# Bibliography

[1] K. Culik II and J. Kari, Digital images. In: [22], Vol. III, 599–616

[2] J. Dassow and F. Hinz, Decision problems and regular chain code picture languages. *Discrete Appl. Math.* **45** (1993) 29–49.

[3] F. Drewes, *Grammatical Picture Generation.* Springer-Verlag, Berlin, 2006.

[4] F. Drewes and H.-J. Kreowski, Picture generation by collage grammars. In: H. Ehrig, G. Engels, H.-J. Kreowski and G. Rozenberg (Eds.), *Handbook of Graph Grammars and Computing by Graph Transformations*, Vol. II, World Scientific, Singapore, 1999, 397–457.

[5] S. Ewert and A. P. J. van der Walt, Random context picture grammars. *Publicationes Math.* **54** (1999) 763–786.

[6] S. Ewert and A. P. J. van der Walt, Generating pictures using random permitting context. *Internat. J. Pattern Recognition* **12** (1999) 939–950.

[7] H. Freeman, On the encoding of arbitrary geometric configurations. *IRE Transactions on Electronic Computers* **10** (1961) 260–268.

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

[9] D. Giammarresi, Finite state recognazability for two-dimensional languages: a brief survey. In: J. Dassow, G. Rozenberg and A. Salomaa (Eds.), *Developments in Language Theory II*, World Scientific, Singapore, 1997, 299–308.

[10] D. Giammarresi and A. Restivo, Two-dimensional languages. In: [22], 215–267.

[11] A. Habel and H.-J. Kreowski, Collage grammars. In: H. Ehrig, H.-J. Kreowski and G. Rozenberg (Eds.), *Proc. 4th Internat. Workshop Graph Grammars and their Application to Computer Science*, Lecture Notes in Computer science 532, Springer-Verlag, Berlin, 1991, 411–329.

[12] G. T. Herman and G. Rozenberg, *Developmental Systems and Languages.* North-Holland, Amsterdam, 1975.

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

[14] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Complexity.* Addison-Wesley, Reading, 1979.

[15] K. Inoue and I. Takanami, A survey of two-dimensional automata theory. In: J. Dassow and J. Kelemen (Eds.), *Machines, Languages, and Complexity*, Lecture Notes in Computer Science 381, Springer-Verlag, Berlin, 1989, 72–91.

[16] Ch. Kim, Complexity and decidability for restricted classes of picture languages. *Theor. Comp. Sci.* **73** (1990) 295–311.

[17] D. C. Kozen *Automata and Computability.* Springer-Verlag, New York, 1997.

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

[19] P. Pruzinkiewicz, M. Hammel, J. Hanan and R. Mech, Visual models of plant development. In: [22], Vol. III, 1997, 535–597.

[20] P. Pruzinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants.* Springer-Verlag, New York, 1990.

[21] G. Rozenberg and A. Salomaa, *The Mathematical Theory of L Systems.* Academic Press, New York, 1980.

[22] G. Rozenberg and A. Salomaa (Eds.), *Handbook of Formal Languages.* Vol. I – III, Springer-Verlag, Berlin, 1997.

[23] A. Rosenfeld and R. Siromoney, Picture languages - a survey. *Languages of Design* **1** (1993) 229–245.

[24] I. H. Sudborough and E. Welzl, Complexity and decidability of chain code picture languages. *Theor. Comp. Sci.* **36** (1985) 175–202.

[25] C. Kim, Picture Iteration and Picture Ambiguity. *J. Comput. Syst. Sci.* **40** (1990) 289–306.

[26] A. Rosenfeld, Isotonic grammars, parallel grammars, and picture grammars. In: *Machine Intelligence* 6 (Eds. B. Meltzer and D. Michie), Edinburgh Univ. Press, 1971, 281–294.

[27] A. Rosenfeld, *Picture Languages – Formal Models for Picture Recognition.* Academic Press, New York, 1979.

[28] A. Rosenfeld, Array grammars. In: *Graph-Grammars and Their Application to Computer Science*; Springer, Berlin, 1986, 67–70.

[29] A. Saoudi, K. Rangarajan and V. R. Dare, Finite images generated by GL-systems. In: [36], 181–190.

[30] R. Siromoney On equal matrix languages. *Inform. Control* **14** (1969) 135–151.

[31] R. Siromoney, K. G. Subramanian and V. Rajkumar Dare, Infinite arrays and controlled deterministic table 0L array systems. Theor. Comput. Sci. **33** (1984) 3–11.

[32] R. Stiebe Picture Generation Using Matrix Systems. Proc. Intern. Meeting Young Computer Scientists, Bratislava, 1990, 251–260

[33] R. Stiebe Picture generation using matrix systems. *Journal of Information Processing and Cybernetics (EIK)* **28** (1992) 311–327.

[34] R. Stiebe, Subimage problems for Siromoney matrix languages. In: Proc. 14. Theorietag Automaten und formale Sprachen, 2004, Univ. Potsdam, 123–128.

[35] R. Stiebe, Subimage problems for Siromoney matrix languages. Manuscript, 2006.

[36] P. S.-P. Wang (Ed.), *Array Grammars, Patterns and Recognizers.*World Scientific, 1989.

[37] P. S.-P. Wang, Three-dimensional sequential/parallel universal array grammars and object pattern analysis. *ICPIA* (1992) 305–312.