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

Introduction

Pictures are an important aspect of our life. They occur almost everywhere as technical drawings describing machines or buildings etc., as paintings of an artist, as language-independent hints and so on. Moreover, pictures have a lot of advantages in comparison with words (we only mention the saying that a picture tells more than 1000 words). Thus there is large interest to generate pictures by computers. Therefore it is not surprising that a lot of picture-generating devices were introduced in the last four decades. They are based on two-dimensional automata (see e.g. [15]), on weighted finite automata (see e.g [1]), on a generalization of the concept of local languages from words (sequences of letters) to matrices of letters (see [9, 10]) and on different kinds of grammars.

Picture generating grammars are the subject of this lecture. The main focus will be given to the following three types of grammars:

- Chain code picture grammars

  Here, firstly, languages of words over certain alphabets are generated and, secondly, the letters of the alphabet are interpreted as directions. Thus a word can be interpreted as a sequence of movements along or of drawings of lines of a unit length (e.g. as it is done by a plotter) which form a picture. This approach was initiated by H. Freeman in [7, 8] in 1961 and studied intensively from point of formal languages by H. A. Maurer, E. Welzl, F. Hinz, I. Sudborough, Ch. Kim and others in the eighties (see e.g. [18, 24, 2, 16]). Chain code picture languages generated by Lindenmayer systems are used to produce pictorial descriptions of the development of plants (see [20, 19]).

- Picture grammars based on arrays

  Here one generates matrices of letters instead of words. There are some mechanisms to produce the arrays, e.g. arrays grammars (where nonterminals are replaced by matrices of letters instead of words) and Siromoney matrix grammars (where, first, a word is produced and then any letter generates a row of letters) (see e.g. [23]). Finally, any letter is interpreted as a (small) picture of fixed size such that a matrix of letters corresponds to a large picture. The research in this direction started in the sixties already. We shall investigate Siromoney matrix grammars as a typical example of this approach.

- Collage grammars.

  The basic idea of this grammar type is to generalize (context-free) graph grammars in such a way that one directly replaces a subpicture by another picture. This approach was initiated by H. J. Kreowski, A. Habel and F. Drewes in the beginning of the nineties (see [11, 4]).

Other grammatical devices, e.g. the random context picture grammars by S. Ewert and A. P. J. van der Walt (see [5, 6]) will be mentioned only shortly or omitted.

We shall discuss hierarchies within one type of generation of picture languages with respect to the type of the rules, relations between the different approaches and decidability of language theoretic as well as geometric properties.

Most topics of this lecture are covered by the book *Grammatical Picture Generation* by F. Drewes. However, Drewes' book develops the theory on the basis of tree grammars, tree automata and tree transformations. In contrast, in this lecture we only assume that the reader is familiar with the "classical" concepts of the theory of phrase structure grammars which can be found in most standard text books on theoretical computer science (see e.g. [14, 17]). In Chapter 1 we give a summary of the definitions and facts from the theory of phrase structure grammars and their formal languages used in the lecture. Besides definitions and statements we add some easy proofs that a reader gets an impression of the working of the grammars. Moreover, we need some very basic knowledge on Lindenmayer systems. Detailed information on Lindenmayer systems can be found in the textbooks [12, 21]. In Chapter 1 we give the definitions of this type of language generating devices, and mostly we need only these basic definitions.

Within a Chapter x all definitions, examples, theorems, lemmas, corollaries etc. have a common numbering, i.e., in the text the successor of Definition x.y can be Lemma x.y+1, which is followed by Example x.y+2 and so on. We conclude a proof by the symbol $\Box$. If the proof of a statement is omitted, then already the formulation of the statement ends with $\Box$. The end of an example is denoted by $\diamond$.

Jürgen Dassow Magdeburg, April 2011

Contents

Chapter 1

Basics of Formal Language Theory

The subject of this lecture is the generation of pictures by grammars or devices similar to grammars. Thus in many places we shall use definitions, concepts and statements of the theory of formal languages. Detailed information on formal languages can be found in [22]. Most material which is used in the lecture can be found in standard textbooks on theoretical computer science (see e.g. [14, 17]) and is presented in basic courses on theoretical computer science. Thus we assume that the reader is familiar with these basic notions, concepts and ideas.

In this chapter we give a summary of the definitions and facts which will be used in this lecture. Besides the notions and statements we also present some easy proofs and examples. Thus a reader without basic knowledge on formal language can consider this chapter as an introduction in the field.

1.1 Phrase Structure Grammars

An alphabet is a non-empty finite set. A word (over an alphabet V) is a finite sequence of letters (of V). By λ we denote the empty word (which consists of no letter). By V^* (and V^+) we designate the set of all (non-empty) words over V. We denote the number of occurrences of a letter a in a word w by $\#_a(w)$. The length of a word w is defined by

$$|w| = \sum_{a \in V} \#_a(w)\,.$$

A language (over V) is a subset of V^*.

We say that two languages L_1 and L_2 are equal iff $L_1 \setminus \{\lambda\} = L_2 \setminus \{\lambda\}$, i.e., if two languages only differ in the empty word, then the difference between them is ignored.

For an alphabet $V = \{a_1, a_2, \ldots, a_n\}$ (with an fixed enumeration of the letters) and a word $w \in V^*$, we define the Parikh vector of w by

$$\Psi(w) = (\#_{a_1}(w), \#_{a_2}(w), \ldots, \#_{a_n}(w)),$$

i.e., we map w onto a n-dimensional vector over the natural numbers, where the i-th component gives the number of occurrences of the i-th letter.

A set $M \subset \mathbb{N}^n$ is called semi-linear if and only if there are natural numbers $m \geq 1$ and $r_i \geq 1$, $1 \leq i \leq m$, and vectors $\underline{a_{ij}} \in \mathbb{N}^n =$, $1 \leq i \leq m,\ 0 \leq j \leq r_i$, such that

$$M = \bigcup_{i=1}^{m} \{\underline{a_{i0}} + \sum_{j=1}^{r_i} \alpha_{ij}\underline{a_{ij}} \mid \alpha_{ij} \in \mathbb{N} \text{ for } 1 \leq j \leq r_i\}\,.$$

A language L is called semi-linear if its Parikh set $\Psi(L) = \Psi(\{w \mid w \in L\}$ is semi-linear.

Example 1.1 We consider the alphabet

$$V = \{a,\ c,\ \int,\ \alpha,\ \circ,\ ;\}$$

consisting of the latin letters a and c, the symbol for the integral, the greek letter α, the symbol $\circ$ used for operations in mathematics, and the semicolon (alphabets are sets; it is not necessary that we associate some meaning with the letters and the words built from the letters; however, the use of alphabet comes from the fact that most concepts are overtaken from linguistics). Then we have the words

$$w_1 = a\,c\,c \int\!\!\int ;\,\alpha \quad \text{and} \quad w_2 = \int ;\,;\,;\,\circ\,\alpha$$

over V. Moreover,

$$\#_a(w_1) = \#_\alpha(w_1) = 1,\ \#_;(w_2) = 3,\ |w_1| = 7,\ |w_2| = 6$$

and

$$\Psi(w_1) = (1,2,2,1,0,1) \quad \text{and} \quad \Psi(w_2) = (0,0,1,1,1,3)\,.$$

Examples for languages over V are

$$L_1 = \{w_1, w_2\} \quad \text{and} \quad L_2 = \{a\alpha^n c \mid n \geq 1\} = \{a\alpha c, a\alpha\alpha c, \dots\}$$

(where we write α^n for the sequence of length n which consists of α only).

The language L_2 is semi-linear since

$$\Psi(L_2) = \{(1,1,0,n,0,0) \mid n \geq 0\} = \{(1,1,0,0,0,0) + \alpha(0,0,0,1,0,0) \mid \alpha \in \mathbb{N}\}\,.$$

$\diamond$

Theorem 1.2 *The intersection of two semi-linear sets is semi-linear, too.* $\square$

We now introduce one of the basic concepts of this lecture – the phrase structure grammars.

Definition 1.3 *i) A phrase structure grammar is a quadruple $G = (N, T, P, S)$, where*
— N and T are alphabets (sets of nonterminals and terminals, resp.),
— $N \cap T = \emptyset$,
— P is a finite subset of $(V^* \setminus T^*) \times V^*)$ (set of rules/productions),

(instead of (α, β) we write $\alpha \to \beta$),
— $S \in N$ *(axiom/start symbol).*

ii) We say that x directly derives (generates) y (written as $x \Longrightarrow_G y$) iff

$$x = x_1 \alpha x_2, \quad y = x_1 \beta x_2, \quad \alpha \to \beta \in P.$$

iii) The language generated by G is defined as

$$L(G) = \{z \mid z \in T^* \text{ and } S \Longrightarrow_G^* z\}$$

where $\Longrightarrow_G^*$ is the reflexive and transitive closure of $\Longrightarrow_G$.

If the grammar under consideration is clear from the context, we omit the subscript G and write $\Longrightarrow$ and $\Longrightarrow^*$.

Example 1.4 i) We consider the grammar

$$G_1 = (\{S\},\ \{(,), [,]\},\ P_1, S)$$

with

$$P_1 = \{S \to SS,\ S \to (S),\ S \to [S],\ S \to (\ ),\ S \to [\ ]\}.$$

We show that $L(G_1)$ is the set of all correctly bracketed expressions over the pairs $(,)$ and $[,]$ of brackets.

We first prove (by induction on number of derivation steps) that only corrected bracketed expressions can be generated. If we generate a word by one derivation step, then we only have the derivations $S \Longrightarrow (\ )$ or $S \Longrightarrow [\ ]$ (since, otherwise, we do not derive a word over the terminal alphabet $\{[,], (,)\}$). Therefore we get correctly bracketed expressions. Now assume, that we have a derivation with $n \geq 2$ derivation steps. Then we get derivations of the form

$$\begin{array}{lll} S & \Longrightarrow & SS \Longrightarrow^* w_1 S \Longrightarrow^* w_1 w_2, \\ S & \Longrightarrow & (\,S\,) \Longrightarrow^* (\,w_1\,), \\ S & \Longrightarrow & [\,S\,] \Longrightarrow^* [\,w_1\,] \end{array} \tag{1.1}$$

where the derivations $S \Longrightarrow^* w_1$ and $S \Longrightarrow^* w_2$ have a length at most $n-1$. By induction hypothesis, w_1 and w_2 are correctly bracketed. Thus $w_1 w_2$, (w_1) and $[w_1]$ are also correctly bracketed.

Conversely, we prove by induction on the length of the expression that all correctly bracketed expression can be obtained. If the length is 2 (by the correctness, the words have an even length), then the only possible expression are $(\ )$ and $[\ ]$. Both expression can be generated by application of the rules $S \to (\ )$ and $S \to [\ ]$ to the start element S. Now let w be an correctly bracketed expression of length $n \geq 4$. Then $w = w_1 w_2$ or $w = (w_1)$ or $w = [w_1]$ for some correctly bracketed expressions w_1 and w_2. By induction hypothesis, we have derivations $S \Longrightarrow^* w_1$ and $S \Longrightarrow^* w_2$. Now w can be generated by derivations of the forms given in (1.1).

ii) We consider the grammar

$$G_2 = (\{S, \#, \S, A, B, C,\},\ \{a, b\},\ P_2,\ S)$$

with

$$\begin{aligned} P_2 \;=\; \{&S \to bbabb,\ S \to \#Aa\S,\ \#Aa \to \#aaA,\ aAa \to aaaA,\ aA\S \to aB\S,\\ &aB \to Ba, \#B \to \#A, \#B \to \#C,\ \#Ca \to bbaC, aCa \to aaC, aC\S \to abb\} \end{aligned}$$

We consider a word $w = \#Aa^{2^n}\S$ (the word with $n = 0$ can be derived from the axiom by application of the second rule). We can use only the third rule and get $\#aaAa^{2^n-1}\S$. Now we can apply only the fourth rule and obtain the derivation

$$\#aaAa^{2^n-1}\S \Longrightarrow \#aaaaA^{2^n-2}\S \Longrightarrow \#a^6Aa^{2^n-3}\S \Longrightarrow^* \#a^{2^{n+1}}A\S\,.$$

Now we have to apply once the fifth rule and then some times $aB \to Ba$ which leads to

$$\#a^{2^{n+1}}A\S \Longrightarrow \#a^{2^{n+1}}B\S \Longrightarrow \#a^{2^{n+1}-1}Ba\S \Longrightarrow \#a^{2^{n+1}-2}Baa\S \Longrightarrow^* \#Ba^{2^{n+1}}\S\,.$$

Now the rules $\#B \to \#A$ and $\#B \to \#C$ are applicable. In the former case we get $\#Aa^{2^{n+1}}\S$, i.e., we obtain a word of the same form as w and we can iterate the process. In the latter case we have the derivation

$$\#Ba^{2^{n+1}}\S \Longrightarrow \#Ca^{2^{n+1}}\S \Longrightarrow bbaCa^{2^{n+1}-1}\S \Longrightarrow bbaaCa^{2^{n+1}-2}\S \Longrightarrow^* bba^{2^{n+1}}C\S \Longrightarrow bba^{2^{n+1}}bb\,.$$

Taking into consideration the application of the first rule we get

$$L(G_2) = \{bba^{2^n}bb \mid n \geq 0\}\,.$$

$\diamond$

Definition 1.5 *i) G is called monotone, if $|\alpha| \leq |\beta|$ holds for all rules $\alpha \to \beta$ of P.*

ii) G is called context-free, if all rules of P are of the form $A \to w$ with $A \in N$ and $w \in V^*$.

iii) G is called regular, if all rules of P are of the form $A \to wB$ or $A \to w$ with $A, B \in N$ and $w \in T^*$.

iv) A language L is called monotone or context-free or regular, iff $L = L(G)$ for some monotone or context-free or regular grammar G, respectively.

The grammar G_1 is context-free (the left hand side of any rule only consist of the nonterminal S), it is also monotone (the right hand sides have a length at least two and therefore greater than the length one of the left hand sides). Obviously, G_1 is a phrase structure grammar. However, G_1 is not regular, since the rule $S \to SS$ and some others do not have the required form.

The grammar G_2 is a monotone phrase structure grammar. But it is neither context-free nor regular since its production set P_2 contains rules with a left hand side consisting of a word of length ≥ 2.

We denote the families of all regular, context-free and monotone languages by $\mathcal{L}(REG)$, $\mathcal{L}(CF)$ and $\mathcal{L}(CS)$[1], respectively. $\mathcal{L}(RE)$[2] denotes the family of all languages which can be generated by phrase-structure grammars.

[1]We use the usual notation which refers to context-sensitive grammars. This can be done because a language is context-sensitive ii it is monotone. We omit the definition of context-sensitive grammars/languages since we use always monotone grammars

[2]The abbreviation RE stands for recursively enumerable sets. This notation is taken from recursion theory and justified by Theorem 1.20.

We now give some normal forms for grammars, i.e., we put some restriction to the form of rules without restricting the family of languages.

Theorem 1.6 *i) For any language $L \in \mathcal{L}(RE)$, there is a phrase-structure grammar $G = (N, T, P, S)$ such that $L = L(G)$ and P has only rules of the forms $A \to B$, $A \to BC$, $AB \to CD$, $A \to a$ and $A \to \lambda$, where $A, B, C, D \in N$ and $a \in T$.*

ii) For any language $L \in \mathcal{L}(CS)$, there is a monotone grammar $G = (N, T, P, S)$ such that $L = L(G)$ and P has only rules of the forms $A \to B$, $A \to BC$, $AB \to CD$ and $A \to a$, where $A, B, C, D \in N$ and $a \in T$.

iii) For any language $L \in \mathcal{L}(CF)$, there is a context-free grammar $G = (N, T, P, S)$ such that $L = L(G)$ and P has only rules of the forms $A \to BC$ and $A \to a$, where $A, B, C \in N$ and $a \in T$.

iv) For any language $L \in \mathcal{L}(REG)$, there is a regular grammar $G = (N, T, P, S)$ such that $L = L(G)$ and P has only rules of the forms $A \to aB$ and $A \to a$, where $A, B \in N$ and $a \in T$.

Proof. We only prove the statement iv). Let L be a regular language. Then there is a regular grammar $G = (N, T, P, S)$ with $L(G) = L$. We first construct a regular grammar $G' = (N, T, P', S)$ such that $L(G') = L$ and P' contains no rules of the form $A \to B$ with $A, B \in N$.

For any letter A, we consider all derivations in G which have the form

$$A \Longrightarrow_G A_1 \Longrightarrow_G A_2 \Longrightarrow_G \ldots \Longrightarrow_G A_r \Longrightarrow_G x \tag{1.2}$$

with $A, A_1, A_2, \ldots, A_r \in N$ and $x \notin N$ (note that x is a terminal word or a non-empty terminal word followed by exactly one nonterminal). We define the set P_A as the set of all rules $A \to x$ such that there is a derivation of the form (1.2). Obviously, P_A contains no rules of the form $A \to B$ with $B \in N$. We now define the grammar

$$G' = (N, T, (P \setminus \{A \to B \mid A \to B \in P,\ A, B \in N\}) \cup \bigcup_{A \in N} P_A, S).$$

Clearly, G' is regular and its set of rules contains no rule $A \to B$ with $A, B \in N$.

We now prove that $L(G') = L(G) = L$. Assume that there is a derivation in G which uses a rule $A \to B$ with $A, B \in N$, i.e.,

$$D\ :\ S \Longrightarrow_G^* xA \Longrightarrow_G xB \Longrightarrow_G^* xy \in T^*\,.$$

Since D terminates, there is a $k \geq 0$ such that the derivation D has the form

$$D:\ S \Longrightarrow_G^* xA \Longrightarrow_G xB \Longrightarrow_G xB_1 \Longrightarrow xB_2 \Longrightarrow_G \ldots \Longrightarrow_G xB_k \Longrightarrow_G xz \Longrightarrow^* xy'$$

with $z \notin N$ und $z \Longrightarrow_G^* y$. Then in G' we have the derivation

$$D':\ S \Longrightarrow_G^* xA \Longrightarrow_{G'} xz \Longrightarrow_G^* xy$$

using $A \to z \in P_A$. By iterated application of this procedure we get a derivation $S \Longrightarrow xy$ G' which uses only rules from G, which do not have the form $A \to B$ with $A, B \in N$, and rules from $\bigcup_{A \in N} P_A$. Thus it is a derivation in G'. Thus $L(G) \subseteq L(G')$.

On the other hand, let D' be a derivation in G' using some rules $A \to x \in P_A$. For each such rule there is a derivation of the form (1.2) in G. If we substitute any derivation step $uA \Longrightarrow_{G'} ux$ by the derivation $uA \Longrightarrow_G uA_1 \Longrightarrow_G uA_2 \Longrightarrow_G \ldots \Longrightarrow_G uA_r \Longrightarrow_G ux$ in G we get a derivation in G which derives the same word as G. Hence $L(G') \subseteq L(G)$.

Analogously, we can show that all rules of the form $A \to \lambda$ (instead of $B \Longrightarrow wA \Longrightarrow w$ take $B \Longrightarrow w$ by using the new rule $B \to w$). Let G'' be the obtained regular grammar.

Finally we construct the grammar G''' from G'' by replacing each rule $A \to a_1a_2 \ldots a_rB$ with $r \geq 2$, $A, B \in N$ and $a_i \in T$ for $1 \leq i \leq r$ by the rules

$$A \to a_1A_1,\ A_1 \to a_2A_2, \ldots,\ A_{r-2} \to a_{r-1}A_{r-1},\ A_{r-1} \to a_rB$$

and by replacing each rule $A \to a_1a_2 \ldots a_s$ with $s \geq 2$, $A \in N$ and $a_j \in T$ for $1 \leq j \leq s$ by the rules

$$A \to a_1B_1,\ B_1 \to a_2B_2, \ldots,\ B_{s-2} \to a_{s-1}B_{s-1},\ B_{s-1} \to a_r$$

where $A_1, A_2, \ldots, A_{r-1}, B_1, B_2, \ldots, B_{s-1}$ are new nonterminals. Then we have a derivation

$uA \Longrightarrow_{G''} ua_1a_2 \ldots a_rB$ if and only if $uA \Longrightarrow_{G'''} ua_1A_1 \Longrightarrow_{G'''} ua_1a_2A_2 \Longrightarrow^*_{G'''} ua_1a_2 \ldots a_rB$

and

$uA \Longrightarrow_{G''} ua_1a_2 \ldots a_r$ if and only if $uA \Longrightarrow_{G'''} ua_1B_1 \Longrightarrow_{G'''} ua_1a_2B_2 \Longrightarrow^*_{G'''} ua_1a_2 \ldots a_r$.

Hence $L(G'') = L(G''')$.

Summarizing we get $L(G''') = L$ and G''' has all required properties. □

Theorem 1.7 *a) Let L be a regular language. Then there is a constant k (which depends on L) such that, for any word $z \in L$ with $|z| \geq k$, there are words u, v, w which satisfy the following properties:*
i) $z = uvw$,
ii) $|uv| \leq k$, $|v| > 0$, and
iii) $uv^iw \in L$ for all $i \geq 0$.

b) Let L be context-free language. Then there is a constant k (which depends on L) such that, for any word $z \in L$ with $|z| \geq k$, there are words u, v, w, x, y which satisfy the following properties:
i) $z = uvwxy$,
ii) $|vwx| \leq k$, $|vx| > 0$, and
iii) $uv^iwx^iy \in L$ for all $i \geq 0$.

Proof. We only prove i). Let $L \in \mathcal{L}(REG)$. By Theorem 1.6, there is a regular grammar $G = (N, T, P, S)$ such that $L(G) = L$ and all its rules are of the form $A \to aB$ or $A \to a$ with $A, B \in N$ and $a \in T$. Thus any derivation has the form

$$S = A_0 \Longrightarrow a_1A_1 \Longrightarrow a_1a_2A_2 \Longrightarrow \ldots \Longrightarrow a_1a_2 \ldots a_{n-1}A_{n-1} \Longrightarrow a_1a_2 \ldots a_{n-1}a_n\,. \quad (1.3)$$

Let $k = \#(N) + 1$. If $n \geq k$, then in (1.3) there are two nonterminals A_i and A_j such that $i < j \leq k$ and $A_i = A_j$. We set

$$u = a_1a_2 \ldots a_i,\ v = a_{i+1}a_{i+2} \ldots a_j,\ w = a_{j+1}a_{j+2} \ldots a_n\,.$$

Then $|uv| \leq k$ and $|v| > 0$. Moreover, for any $m \geq 0$, we have the derivation

$$S \Longrightarrow^* uA_i \Longrightarrow^* uvA_j = uvA_i \Longrightarrow uvvA_j = uvvA_i \Longrightarrow^* uv^mA_j \Longrightarrow^* uv^mw$$

which proves that $uv^mw \in L(G) = L$ for any $m \geq 0$. □

Finally in this section we mention a property of the Parikh-sets of context-free languages.

Theorem 1.8 *For any context-free language L, $\Psi(L)$ is semi-linear.* □

1.2 Lindenmayer Systems

A characteristic property of the derivation process of the context-free grammars is that in any step exactly one letter is replaced and using the normal forms (see Theorem 1.6 for general phrase structure grammars and monotone grammars exactly one subword of length at most two is replaced. This contrast processes in biology where mostly all cells develop in one step. Therefore A. Lindenmayer introduced a new type of grammar-like devices where the characteristic feature is a parallel replacement of all letters.

Definition 1.9 *i) An extended tabled Lindenmayer system (abbr. by ET0L) with n tables is an $(n+3)$-tuple $G = (V, T, P_1, P_2, \ldots, P_n, w)$, where*
– V is a finite alphabet, T is a non-empty subset of V,
– for $1 \leq i \leq n$, P_i is a finite subset of $V \times V^*$ such that, for any $a \in V$, there is a pair (a, w_a) in P_i,
– $w \in V^+$.
ii) We say that x directly derives (generates) y (written as $x \Longrightarrow_G y$) iff there is an i, $1 \leq i \leq n$, such that

$$\begin{aligned} &x = x_1x_2\ldots x_m,\ x_j \in V \text{ for } 1 \leq j \leq m,\\ &y = y_1y_2\ldots y_m, \text{ and}\\ &x_j \to y_j \in P_i \text{ for } 1 \leq j \leq m. \end{aligned}$$

iii) The language generated by G is defined as

$$L(G) = \{z \mid z \in T^* \text{ and } w \Longrightarrow_G^* z\}$$

where $\Longrightarrow_G^*$ is the reflexive and transitive closure of $\Longrightarrow_G$.

The set T is called the terminal alphabet of G. The sets P_i, $1 \leq i \leq n$, are called tables; they are set of productions (a, v) for which we write $a \to v$ (as in the case of grammars).

By this definition the most importance difference to the classical (sequential) phrase structure grammars is the parallelism in the derivation: any letter of a sentential form is replaced according to the rules of some set P_i, $1 \leq i \leq n$. However, there is also a difference with respect to the set T of terminals. It is only used to filter out the words in the language which consist of those sentential forms which only contain terminal symbols;

but there are also rules for the letters of $V \setminus T$ (i.e., with respect to the applicability of rules there is no difference between the letters). Moreover, we start the derivation with a non-empty word and not necessarily with a single letter from the nonterminal alphabet.

We give some examples.

Example 1.10 i) Let

$$H_1 = (\{a, b\},\ \{a, b\},\ \{a \to aa, b \to b\},\ bbabb)$$

be an ET0L system. Since there is only one rule for any letter we get a unique derivation

$$bbabb \Longrightarrow bbaabb \Longrightarrow bbaaaabb \Longrightarrow bbaaaaaaaabb \Longrightarrow bba^{16}bb \Longrightarrow \dots$$

from which immediately follows that

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

ii) We consider the ET0L system

$$H_2 = (\{a, b\},\ \{a\},\ \{a \to a,\ a \to aa,\ b \to b,\ b \to \lambda\},\ ab)\,.$$

If w can be generated from the axiom ab, i.e., $ab \Longrightarrow^* w$, then $w = a^n b$ or $w = a^n$ for some $n \geq 1$. This follows from the fact that such words only generate words of this form which can be seen as follows. Let $v = a^m b$ for some $m \geq 1$. Let $m = m_1 + m_2$. We now apply $a \to a$ to m_1 occurrences of a and $a \to a^2$ to the remaining m_2 occurrences of a and to b one of the possible rules, then we get

$$a^m b \Longrightarrow a^{m_1+2m_2} b \quad \text{or} \quad a^m b \Longrightarrow a^{m_1+2m_2}$$

which both have the required form. Analogously, $a^m \Longrightarrow a^{m_1+2m_2}$ gives words of the asked form, too. On the other hand, applying $a \to a^2$ to only one occurrence of a, for any $n \geq 1$, we get the derivations

$$ab \Longrightarrow aab \Longrightarrow aaab \Longrightarrow a^4 b \Longrightarrow \dots a^n b$$

and

$$ab \Longrightarrow aab \Longrightarrow aaab \Longrightarrow a^4 b \Longrightarrow \dots a^{n-1} b \Longrightarrow a^n$$

(using $b \to b$ with the exception of the last mentioned derivation step where $b \to \lambda$ is applied). This proves that all words of the forms can be generated.

Taking into consideration that a is the only terminal letter, we get

$$L(H_2) = \{a^n \mid n \geq 1\}\,.$$

iii) Let

$$H_3 = (\{a, b, c\},\ \{a, b\},\ P_1,\ P_2,\ ca)$$

with

$$P_1 = \{a \to aa, b \to b, c \to ca\} \text{ and } P_2 = \{a \to b, b \to bbb, c \to a\}\,.$$

We start any derivation with ca, a word over $\{a, c\}$. As long as we apply only the rules from P_1, this situation is not changed, we only generate words over $\{a, c\}$. The application of rules from P_2 leads to words over $\{a, b\}$, a situation which is again not changed by applications of P_1. A second application of P_2 gives words over the singleton alphabet $\{b\}$, which is not changed by any rule. In order to determine the language we therefore assume that the derivation starts with applications of P_1. This gives all words of the form ca^{2^n-1} and only such words, because we start with ca of that form and

$$ca^{2^n-1} \Longrightarrow_{P_1} caa^{2\cdot(2^n-1)} = ca^{1+2^{n+1}-2} = ca^{2^{n+1}-1}.$$

The application of P_2 to such a word gives ab^{2^n-1}. If we now apply sometimes P_1 we get

$$ab^{2^n-1} \Longrightarrow_{P_1} a^2b^{2^n-1} \Longrightarrow_{P_1} a^4b^{2^n-1} \Longrightarrow_{P_1} \Longrightarrow_{P_1} \ldots \Longrightarrow_{P_1} a^{2^m}b^{2^n-1}$$

for some $m \geq 0$. The application of P_2 leads to $a^{2^m}b^{2^n-1} \Longrightarrow_{P_2} b^{2^m}b^{3(2^n-1)} = b^{2^m+3(2^n-1)}$. Now the application of P_1 does not change the word. Moreover, by applications of P_2,

$$b^{2^m+3(2^n-1)} \Longrightarrow b^{3(2^m+3(2^n-1))} \Longrightarrow b^{3^2(2^m+3(2^n-1))} \Longrightarrow b^{3^3(2^m+3(2^n-1))} \Longrightarrow \ldots$$

If we take into consideration that the terminal set does not contain c, we obtain

$$L(H_3) = \{a^{2^m}b^{2^n-1} \mid m \geq 1, n \geq 1\} \cup \{b^{3^k(2^m+3(2^n-1))} \mid n \geq 1, m \geq 0, k \geq 0\}\}.$$

◇

By $\mathcal{L}(ET0L)$ we denote the family of all languages generated by ET0L systems.

We now define special types of ET0L systems. We omit the letter E if the generating system satisfies $V = T$. We omit the letter T if the generating system satisfies $n = 1$ (non-tabled case). We add the letter D if the generating system is deterministic, i.e., for all $1 \leq i \leq n$ and all $a \in V$, there is exactly one rule with left side a in P_i. Such we get D0L, ED0L, EDT0L, 0L, E0L, T0L and EDT0L systems.

Let $X \in \{ET, EDT, ED, E, T, DT, D, \lambda\}$. We call a language a X0L language if it is generated by some X0L system. By $\mathcal{L}(X0L)$ we denote the set of all X0L languages.

The system H_1 is deterministic, satisfies $V = T$ and is not tabled; therefore H_1 is a D0L system, but it is also an X0L system for any X defined above. The system H_2 is a T0L system and H_3 is a EDT0L system.

1.3 Hierarchies and Closure Properties

One of the most investigated question concerns the relations between all the families of languages which we have defined in the two preceding sections. We summarize the known facts in the following theorem.

Theorem 1.11 *The diagram of Figure 1.1 holds where $\mathcal{L}(X) \subset \mathcal{L}(Y)$ if and only if there is a (directed) path from $\mathcal{L}(X)$ to $\mathcal{L}(Y)$ and two families are incomparable if they are not connected.*

$$\begin{tikzcd}
 & & & \mathcal{L}(RE) \\
 & & & \mathcal{L}(CS) \arrow[u] \\
 & \mathcal{L}(ET0L) \arrow[urr] & & \\
\mathcal{L}(EDT0L) \arrow[ur] & \mathcal{L}(T0L) \arrow[u] & \mathcal{L}(E0L) \arrow[ul] & \\
\mathcal{L}(DT0L) \arrow[u] \arrow[ur] & \mathcal{L}(ED0L) \arrow[ul] \arrow[ur] & \mathcal{L}(0L) \arrow[ul] \arrow[u] & \mathcal{L}(CF) \arrow[ul] \arrow[uuu] \\
 & \mathcal{L}(D0L) \arrow[ul] \arrow[u] \arrow[ur] & & \mathcal{L}(REG) \arrow[u]
\end{tikzcd}$$

Figure 1.1: Hierarchy of language families

Proof. We do not prove all relations. We only give proofs for some inclusions, some strictnesses of inclusions and some incomparabilities.

i) *Inclusions*

First we mention that all inclusions – with exception of $\mathcal{L}(ET0L) \subseteq \mathcal{L}(CS)$ and $\mathcal{L}(CF) \subseteq \mathcal{L}(E0L)$ – follow from the definition of the systems and grammars (in the normal form (see Theorem 1.6). We omit the proof of the first exceptional inclusion and prove $\mathcal{L}(CF) \subseteq \mathcal{L}(E0L)$.

Let $L \in \mathcal{L}(CF)$. Then there is a context-free grammar $G = (N, T, P, S)$ such that $L(G) = L$. We now construct

$$H = (N \cup T, T, P', S) \text{ with } P' = P \cup \{A \to A \mid A \in N\} \cup \{a \to a \mid T\}\,.$$

By the rules added to P it is obvious that, for any $x \in N \cup T$, there is a rule with left hand side x in the production set of H. Moreover, H has only one table. Therefore H is an E0L system. We now prove that $L(H) = L(G) = L$ which implies $L \in \mathcal{L}(E0L)$.

We consider the derivation step $xAy \Longrightarrow_G xwy$ using $A \to w \in P$. Since we have $z \to z \in P'$ for any letter $z \in N \cup T$ and $A \to w \in P \subseteq P'$, we can apply $z \to z$ to all letters z in x and y and $A \to w$ and get $xAy \Longrightarrow_H xwy$. Hence any derivation in G can be simulated in H, which proves $L(G) \subset L(H)$ (since in both devices we start with S and take into the language only the words over T).

Conversely, let

$$x_0 A_1 x_1 A_2 x_2 \ldots A_n x_n \Longrightarrow_H x_0 w_1 x_1 w_2 x_2 \ldots w_n x_n$$

be a derivation in H where we apply to all letters z of x_i, $0 \le i \le n$, the rules $z \to z \in P'$ and to A_j, $1 \le j \le n$, a rule $A_j \to w_j \in P'$ with $A_j \ne w_j$. Then $A_j \to w_j \in P$, and we have in G the derivation

$$\begin{aligned}x_0A_1x_1A_2x_2\dots A_nx_n &\Longrightarrow_G x_0w_1x_1A_2x_2\dots A_nx_n\\ &\Longrightarrow_G x_0w_1x_1w_2x_2\dots A_nx_n\\ &\quad\dots\\ &\Longrightarrow_G x_0w_1x_1w_2x_2\dots w_nx_n\,.\end{aligned}$$

Thus we can simulate any derivation in H by a derivation in G which gives $L(H) \subseteq L(G)$.

Therefore $L(G) = L(H)$.

ii) $\mathcal{L}(CF) \subset \mathcal{L}(CS)$.

By Example 1.4 ii), $L = \{bba^{2^n}bb \mid n \geq 0\} \in \mathcal{L}(CS)$. We now prove that $L \notin \mathcal{L}(CF)$.

Let us assume that $L \in \mathcal{L}(CF)$. Let k be the constant which exist by Theorem 1.7 b) and $n = k + 3$. Then $n < 2^n$. We consider the word $bba^{2^n}bb \in L$. By Theorem 1.7 b), there is a decomposition $bba^{2^n}bb = uvwxy$ with $|vwx| \leq k$, $vx \neq \lambda$, and $uv^iwx^iy \in L$ for $i \geq 0$. If v or x contain the letter b, then uv^5wx^5y contains at least five occurrences of the letter b which contradicts $uv^5wx^5y \in L$. Hence $v = a^r$ and $x = a^s$ for some r and s with $0 < r + s \leq k < n$. Then $uv^2wx^2y = bba^{2^n+r+s}bb$. Obviously, $bba^{2^n+r+s}bb \in L$ if and only $2^n + r + s$ is a power of 2. However, this is impossible by $2^n < 2^n + r + s \leq 2^n + n < 2^n + 2^n = 2^{n+1}$.

This contradiction proves that our assumption is false.

We can give another proof for $L \notin \mathcal{L}(CF)$ by showing that $\Psi(L) = \{(2^n, 4) \mid n \geq 0\}$ is not a semi-linear set and taling into consideration Theorem 1.8.

iii) $\mathcal{L}(REG) \subset \mathcal{L}(CF)$

By Example 1.4 i), the language L' of all correctly bracketed expression over two pairs of brackets is in $\mathcal{L}(CF)$. We now show that $L' \notin \mathcal{L}(REG)$.

Let us assume that L' is a regular language. Let k be the constant which exist by Theorem 1.7 a) and $n = k + 2$. We consider the word $(^n)^n \in L'$. By Theorem 1.7 a), there is a decomposition $(^n)^n = uvw$ with $|uv| \leq k$, $v \neq \lambda$, and $uv^iw \in L$ for $i \geq 0$. By our choice of n, $v = (^r$ for some r. Thus $uv^2w = (^{n+r})^n \notin L'$ in contrast to the above statement.

Hence our assumption is false.

iv) All subsets of $\mathcal{L}(T0L)$ are incomparable with $\mathcal{L}(CF)$ and $\mathcal{L}(REG)$ are incomparable.

Obviously, it is sufficient to prove that there are languages K and K' such that

$$K \in \mathcal{L}(REG),\ K \notin \mathcal{L}(T0L) \quad\text{and}\quad K' \in \mathcal{L}(D0L),\ K' \notin \mathcal{L}(CF)$$

.

Let $K = \{a^2, a^4\}$. Because the regular grammar $(\{S\}, \{a\}, \{S \to a^2, S \to a^4\}, S)$ obviously generates K, we have $K \in \mathcal{L}(REG)$.

Let us assume that $K = L(G)$ for a T0L system $G = (\{a\}, \{a\}, P_1, P_2, \dots, P_n, w)$. By the definition of the language generated by a T0L system, $w \in L(G)$. We now distinguish two cases.

Case 1. $w = a^2$. Then there is a table P_i, $1 \leq i \leq n$, such that $a^2 \Longrightarrow_{P_i} a^4$. Therefore $a \to a^m \in P_i$ where $m \in \{2, 3\}$. Then we have $a^4 \Longrightarrow_{P_i} a^{4m}$, and therefore $a^{4m} \in L(G)$. But $m \geq 2$ implies $a^{4m} \notin K$ in contradiction to $L(G) = K$.

Case 2. $w = a^4$. Then there is a table P_j, $1 \leq j \leq n$, such that $a^4 \Longrightarrow_{P_j} a^2$. Since we can produce a contradiction as in Case 1, if P_j contains a rule $a \to a^m$ with $m \geq 2$, we can assume that $P_j = \{a \to \lambda, a \to a\}$ (if we only have $a \to \lambda$ or only $a \to a$, then $a^4 \Longrightarrow_{P_j} a^2$ is impossible). However, then we also have $a^4 \Longrightarrow_{P_j} a^3$, i.e., we can generate a word which does not belong to K. Again we have a contradiction to $L(G) = K$.

This proves that our assumption is false and therefore $K \notin \mathcal{L}(T0L)$.

We choose $K' = \{bba^{2^n}bb \mid n \geq 0\}$. By Example 1.10 i), $K \in \mathcal{L}(D0L)$. On the other hand, by part ii) of this proof, $K' \notin \mathcal{L}(CF)$. □

Let X and Y be two alphabets. Furthermore, let L, L_1 and L_2 be languages over X, and let K be a language over Y. Then we set

$$\begin{aligned}
&L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1,\ w_2 \in L_2\} \quad \text{(product, concatenation)},\\
&L^0 = \{\lambda\} \text{ and } L^{i+1} = L^i \cdot L \text{ for } i \geq 0 \quad \text{(power)},\\
&L^+ = \bigcup_{i\geq 1} L^i \text{ and } L^* = \bigcup_{i \geq 0} L^i \quad \text{(Kleene-closure)}
\end{aligned}$$

A mapping $h : X^* \to Y^*$ is a homomorphism if $h(w_1w_2) = h(w_1)h(w_2)$ for all $w_1, w_2 \in X^*$. In order to define a homomorphism h it is sufficient to give $h(a)$ for any $a \in X$ since we have $h(a_1a_2 \dots a_n) = h(a_1)h(a_2)\dots h(a_n)$. For a homomorphism h, we set

$$h(L) = \{h(w) \mid w \in L\} \text{ and } h^{-1}(K) = \{w \mid h(w) \in K\}\,.$$

A substitution $\sigma : X^* \to 2^{Y^*}$ is defined inductively as follows:
– $\sigma(\lambda) = \{\lambda\}$,
– $\sigma(a)$ is a finite subset of Y^* for any $a \in X$,
– $\sigma(wa) = \sigma(w)\sigma(a)$ for $w \in X^*$ and $a \in X$.
Thus, for $w = a_1a_2 \dots a_n$ with $a_i \in X$ for $1 \leq i \leq n$, $\sigma(w)$ consists of all words $z_1z_2 \dots z_n$ where $z_i \in \sigma(a_i)$ for $1 \leq i \leq n$. Moreover, for a language $L \subseteq X^*$, we set

$$\sigma(L) = \bigcup_{w \in L} \sigma(w)\,.$$

Obviously, homomorphisms can be considered as special substitutions, where $\sigma(a)$ consists of exactly one element for each $a \in X$.

A substitution σ (or homomorphism h) is called λ-free iff $\lambda \notin \sigma(a)$ (or $h(a) \neq \lambda$) for all $a \in X$.

Let τ be an n-ary operation on languages. A family $\mathcal{L}$ is closed under τ, if $\tau(L_1, L_2, \dots, L_n) \in \mathcal{L}$ holds for all $L_1, L_2, \dots, L_n \in \mathcal{L}$.

The following theorem summarizes some known closure properties.

Theorem 1.12 *The table of Figure 1.2 holds where a + or a – in the intersection of the row associated with the family $\mathcal{L}(X)$ and the column associated with the operation τ indicates that $\mathcal{L}(X)$ is closed or not closed under τ. Moreover, in the affirmative case we can construct a X grammar or an X system for $\tau(L_1, L_2 \dots, L_n)$ if X grammars or X systems for L_i, $1 \leq i \leq n$, are given.*

	union	product	Kleene-closure	homomorph.	inverse homomorph.	intersect. with reg. sets	substitution
$\mathcal{L}(RE)$	+	+	+	+	+	+	+
$\mathcal{L}(CS)$	+	+	+	−	+	+	−
$\mathcal{L}(CF)$	+	+	+	+	+	+	+
$\mathcal{L}(REG)$	+	+	+	+	+	+	+
$\mathcal{L}(ET0L)$	+	+	+	+	+	+	+
$\mathcal{L}(EDT0L)$	+	+	+	+	−	+	+
$\mathcal{L}(E0L)$	+	+	+	+	−	+	+
$\mathcal{L}(T0L)$	−	−	−	−	−	−	−
$\mathcal{L}(DT0L)$	−	−	−	−	−	−	−
$\mathcal{L}(0L)$	−	−	−	−	−	−	−
$\mathcal{L}(D0L)$	−	−	−	−	−	−	−

Figure 1.2: Table of closure properties

Proof. We only prove some of the properties.

i) $\mathcal{L}(CF)$ *is closed under homomorphisms and substitutions.*

Let $L \in \mathcal{L}(CF)$ be a language over the alphabet X and let $h : X^* \to Y^*$ be a homomorphism. Then there is a context-free grammar $G = (N, T, P, S)$ with $L(G) = L$. Then the context-free grammar

$$G' = (N \cup X, Y, P \cup \{a \to h(a) \mid a \in X\}, S$$

generates $h(L(G)) = h(L)$ because we can continue any derivation $S \Longrightarrow_G^* a_1a_2 \dots a_n$, where $a_i \in X$ for $1 \leq i \leq n$ to a derivation

$$\begin{aligned} S &\Longrightarrow_{G'}^* a_1a_2 \dots a_n \Longrightarrow_{G'} h(a_1)a_2a_3 \dots a_n \Longrightarrow_{G'} h(a_1)h(a_2)a_3 \dots a_n \Longrightarrow_{G'} \dots \\ &\Longrightarrow_{G'} h(a_1)h(a_2) \dots h(a_n)\,. \end{aligned}$$

The proof for substitutions follow the same lines, we only add all rules of the $a \to z$ with $z \in \sigma(a)$.

ii) $\mathcal{L}(ET0L)$ *is closed under homomorphisms and substitutions.*

Let $L \in \mathcal{L}(ET0L)$ be a language over the alphabet X and let $h : X^* \to Y^*$ be a homomorphism. Then there is an ET0L system $G = (V, X, P_1, P_2, \dots P_n, w)$ with $L(G) = L$. Without loss of generality we assume that $V \cap Y = \emptyset$ (the modifications for the general case can be done easily by renaming the letters of V in G). Then we construct the ET0L system

$$G' = (V \cup Y \cup \{F\}, Y, P_1', P_2', \dots, P_n', P_{n+1}', w)$$

with

$$\begin{aligned} P_i' &= P_i \cup \{a \to F \mid a \in Y\} \cup \{F \to F\} \quad \text{for } 1 \leq i \leq n, \\ P_{n+1}' &= \{a \to h(a) \mid a \in X\} \cup \{a \to F \mid a \in V \setminus X\} \cup \{a \to F \mid a \in Y\} \cup \{F \to F\}\,. \end{aligned}$$

As long as we do not apply the table P_{n+1}', we have $x \Longrightarrow_{P_i'} y$ if and only if $x \Longrightarrow_{P_i} y$ since the application only concerns letters of V for which P_i and P_i' contain the same rules. Thus any derivation in G' has the form

$$w \Longrightarrow_G^* v \Longrightarrow_{P'_{n+1}} h(v) \Longrightarrow_{P'_j} F^{|h(v)|} \quad \text{ if } v \in X^*$$

or

$$w \Longrightarrow_G^* v \Longrightarrow_{P'_{n+1}} z_1 F z_2 \text{ for some } z_1, z_2 \in (Y \cup \{F\})^* \quad \text{ if } v \notin X^*$$

(since v contains at least one letter not in X, P'_{n+1} produces at least one F). The result of both derivations is a word containing at least one letter F. Since $F \to F$ is the only rule for F in all tables of G', F will occur in all words which are derived in the sequel. Now it is easy to see that the only words over Y are the words $h(v)$ with $v \in L(G)$. Hence $L(G') = h(L(G)) = h(L)$.

The changes for substitution are analogous to those done in part i) of this proof.

iii) $\mathcal{L}(T0L)$ *is not closed under union, product, homomorphisms, and intersections with regular sets.*

In part iv) of the proof of Theorem 1.11 we have shown that $\{a^2, a^4\} \notin \mathcal{L}(T0L)$.

On the other hand,
– $\{a^2\} \in \mathcal{L}(T0L)$ (generated by $(\{a\}, \{a\}, \{a \to a\}, a^2)$),
– $\{a^4\} \in \mathcal{L}(T0L)$ (generated by $(\{a\}, \{a\}, \{a \to a\}, a^4)$),
– $\{a^2, \lambda\} \in \mathcal{L}(T0L)$ (generated by $(\{a\}, \{a\}, \{a \to \lambda\}, a^2)$),
– $\{a^2, b^4\} \in \mathcal{L}(T0L)$ (generated by $(\{a, b\}, \{a, b\}, \{a \to b^2, b \to b\}, a^2)$),
– $\{a^n \mid n \geq 1\} \in \mathcal{L}(T0L)$ (generated by $(\{a\}, \{a\}, \{a \to a, a \to a^2\}, a)$).

Now the non-closure properties follow from
– $\{a^2\} \cup \{a^4\} = \{a^2, a^4\} \notin \mathcal{L}(T0L)$,
– $\{a^2\} \cdot \{a^2, \lambda\} = \{a^2, a^4\} \notin \mathcal{L}(T0L)$,
– $h(\{a^2, b^4\}) = \{a^2, a^4\} \notin \mathcal{L}(T0L)$ for the homomorphism h given by $h(a) = h(b) = a$,
– $\{a^n \mid n \geq 1\} \cap \{a^2, a^4\} = \{a^2, a^4\} \notin \mathcal{L}(T0L)$ (since $\{a^2, a^4\} \in \mathcal{L}(REG)$, see part iv) of the proof of Theorem 1.11).

iv) $\mathcal{L}(D0L)$ *is not closed under Kleene-closure and inverse homomorphisms.*

The D0L system $(\{a\}, \{a\}, \{a \to a\}, a))$ generates $L = \{a\}$. We show that

$$L^* = \{a^n \mid n \geq 0\} \notin \mathcal{L}(D0L).$$

Let assume that there is a D0L system $G = (\{a\}, \{a\}, P, w)$ such that $L(G) = L^*$. Because G is a deterministic system, $P = \{a \to a^s\}$ for some $s \geq 0$, and $w = a^t$ for some $t \geq 1$. Now it follows that

$$L(G) = \begin{cases} \{a^{ts^m} \mid m \geq 0\} & \text{for } s \geq 1 \\ \{a^t, \lambda\} & \text{for } s = 0 \end{cases}$$

Obviously, we have a contradiction to our assumption $L(G) = L^*$ in all cases.

Let $g : \{a, b\}^* \to \{a\}^*$ be the homomorphism given by $h(a) = a$ and $h(b) = \lambda$. We show that

$$g^{-1}(L) = \{b^n a b^m \mid n \geq 0, m \geq 0\} \notin \mathcal{L}(D0L)\,.$$

Let us assume that there is a D0L system

$$G' = (\{a, b\}, \{a, b\}, \{a \to w_a, b \to w_b, b^r a b^s)$$

such that $L(G') = g^{-1}(L)$ (since G' is deterministic there is only one rule for each letter and by the structure of the words in $g^{-1}(L)$ the axiom has to have that form). If w_b contains an a, then from $b^2a \in g^{-1}(L) = L(G')$ we can generate a word w containing at least two occurrences of a which contradicts $w \in L(G') = g^{-1}(L)$. Thus $w_b = b^t$ for some t. Analogously, we can see that $w_a = b^pab^q$ for some p and q.

If $t = 0$, then $L(G') = \{b^rab^s, b^pab^q\}$ which contradicts $L(G') = g^{-1}(L)$. Thus let $t \geq 1$. Then b^rab^s is the shortest word in $L(G')$. By $L(G') = g^{-1}(L)$, the unique shortest word is a which implies $r = s = 0$. If $p = 0$ or $q = 0$, we can only generate words of the form ab^n or b^na, respectively, which is a contradiction to $L(G') = g^{-1}(L)$, again. If $p \geq 1$ and $q \geq 1$, then we are not able to derive ba and have a contradiction to $L(G') = g^{-1}(L)$. Thus in all cases we get a contradiction which proves that our assumption is false. □

The family $\mathcal{L}(CS)$ is not closed under homomorphisms and substitutions. This situation changes if one restricts to λ-free versions.

Theorem 1.13 *For any monotone grammar G and any λ-free homomorphism h and any λ-free substitution σ, one can construct monotone grammars G_1 and G_2 such that $L(G_1) = h(L(G)$ and $L(G_2) = \sigma(L(G))$.* □

Without proof we mention the following result.

Theorem 1.14 *The families $\mathcal{L}(REG)$ and $\mathcal{L}(CS)$ are closed under complement, but $\mathcal{L}(CF)$ and $\mathcal{L}(RE)$ are not closed under complement.* □

By Theorem 1.12 the family $\mathcal{L}(REG)$ is closed under union, product, and Kleene-closure. The well-known Theorem by Kleene says that the family $\mathcal{L}(REG)$ can be characterized by closure under these three operations.

Theorem 1.15 *A language L over X is regular if and only if L can be generated by a finite number of iterated applications of the operations union, product and Kleene-closure $*$ starting with the sets $\emptyset$, $\{\lambda\}$ and $\{x\}$, $x \in X$.* □

For a language L, we denote the set of all subwords of words of L by $sub(L)$. Formally, we get

$$sub(L) = \{w' \mid w = w_1w'w_2 \text{ for some } w \in L\}.$$

Analogously, by $pref(L)$ and $suff(L)$ we denote the sets of all prefixes and suffixes, respectively, of words in L.

Theorem 1.16 *i) For any regular language L, the sets $sub(L)$, $pref(L)$ and $suff(L)$ are regular, too.*

ii) For any context-free language L, the sets $sub(L)$, $pref(L)$ and $suff(L)$ are context-free, too.

Proof. i) Let $L \in \mathcal{L}(REG)$ and $L \subseteq X^*$. Let $X' = \{a' \mid a \in X\}$. For $w = a_1a_2\ldots a_n$ with $a_i \in X$ for $1 \leq i \leq n$, we set $w' = a_1a_2'\ldots a_n'$. We define the substitution $\sigma : X^* \to (X \cup X')^*$ by $\sigma(a) = \{a, a'\}$ for $a \in X$. By Theorem 1.12,

$$L' = \sigma(L) \cap \{w_1'w_2w_3' \mid w_1, w_2, w_3 \in X^*\} = \{x_1'x_2x_3' \mid x_1x_2x_3 \in L, x_1, x_2, x_3 \in X^*\}$$

is regular because $\{w_1'w_2w_3' \mid w_1, w_2, w_3 \in X^*\} \in \mathcal{L}(REG)$ (the easy proof for this fact is left to the reader). Now let $h : (X \cup X')^* \to X^*$ be the homomorphism given by $h(a) = a$ for $a \in X$ and $h(a') = \lambda$ for $a' \in X'$. Then

$$h(L') = \{x_2 \mid x_1'x_2x_3' \in L'\}$$

is regular by Theorem 1.12. Since it is easy to see that $h(L') = sub(L)$, the assertion is shown.

The easy modifications to prove that $pref(L)$ and $suff(L)$ are regular are left to the reader (one has to filter out x_1 or x_3 instead of x_2).

ii) The proof is identical to part i) because $\mathcal{L}(CF)$ has all the closure properties used in i), too. $\square$

1.4 Turing Machines, Decidability and Complexity

In this section we introduce some basic notions on computability ad complexity, i.e., we discuss concepts which allow statements whether or not a problem is solvable by algorithms and how complex such algorithms are in the affirmative case. As a formalization of the notion of an algorithm we introduce Turing machines.

Definition 1.17 *i) A (non-deterministic) Turing machine is a seven-tuple*

$$\mathcal{M} = (\Gamma, X, *, Z, z_0, Q, F, \delta),$$

where

- *Γ is an alphabet (of tape symbols), $X \subseteq \Gamma$ is an alphabet (of input symbols), and $*$ is a special symbol not in Γ,*
- *Z is a finite set (of states), $z_0 \in Z$ is the initial state, $Q \subseteq Z$ is the set of halt states, $F \subseteq Q$ is the set of accepting states, and*
- *$\delta : (Z \setminus Q) \times (\Gamma \cup \{*\}) \to 2^{Z \times ((\Gamma \cup \{*\}) \times \{R, L, N\}}$ is a (total) function.*

ii) A Turing machine $\mathcal{M}$ is called deterministic if δ maps $(Z \setminus Q) \times (\Gamma \cup \{*\})$ into $Z \times ((\Gamma \cup \{*\}) \times \{R, L, N\}$.

Intuitively, a Turing machine consists of a unit (storing the state), an infinite tape which cells are filled with letters from $\Gamma \cup \{*\}$ and a read/write head. If a machine is in a state z and reads the symbol a in some cell and $(z', a', r) \in \delta(z, x)$, then it changes the state from z to z', replaces a by a' and moves the head one cell to the right, if $r = R$, or one cell to the left, if $r = L$, or does not move the head, if $r = N$. $\mathcal{M}$ halts if it reaches a state of Q.

Definition 1.18 *The set $T(\mathcal{M})$ of words accepted by a Turing machine $\mathcal{M}$ consists of all words w such that $\mathcal{M}$ reaches a state in F if it starts in state z_0 with w written on the tape (i.e., the letters of w are written in consecutive cells and all other cells are filled with $*$) and the head is positioned on the first letter of w.*

We now give an example.

Example 1.19 We consider the deterministic Turing machine

$$\mathcal{M} = (\{a, b\},\ \{a\},\ *,\ Z,\ \{z_5, z_6\},\ \{z_5\},\ \delta)$$

with

$$Z = \{z_0,\ z_1,\ z_2,\ z_3,\ z_4,\ z_5,\ z_6\}$$

and δ given by the following table (in the intersection of the row associated with a tape symbol x and the column associated with a state z we present $\delta(z, x)$)

δ	z_0	z_1	z_2	z_3	z_4
$*$	$(z_0, *, N)$	$(z_5, *, N)$	$(z_4, *, L)$	$(z_6, *, N)$	$(z_0, *, R)$
a	(z_1, a, R)	(z_2, b, R)	(z_3, a, R)	(z_2, b, R)	(z_4, a, L)
b	$(z_0, *, N)$	(z_1, b, R)	(z_2, b, R)	(z_3, b, R)	(z_4, b, L)

The machine works as follows: It reads the word from left to right and replaces every second a by b and do not change the b's at the tape. We have three situation, if it reaches the $*$ after the word:

- it has read an odd number $k \geq 3$ of a's ($\mathcal{M}$ is in state z_3), then it stops the work in state z_6, i.e., the word is not accepted,
- it has read an even number $k \geq 2$ of a's ($\mathcal{M}$ is in state z_2), then the head moves back to the first letter, and the process is started again; this work corresponds to a division of the number of occurrences of a's by 2,
- it has read exactly one a ($\mathcal{M}$ is in state z_1), then it accepts.

Therefore a word is accepted if the machine performs a certain number of divisions by 2 and reads exactly one a during the last move from left to right. Thus we get

$$T(\mathcal{M}) = \{a^{2^n} \mid n \geq 0\}$$

(Because we want to replace every second occurrence of a, to perform a division by 2, we need an additional letter b which is not in the input alphabet, but in the tape alphabet.) $\diamond$

We now give (without proof) a relation between Turing machines and languages generated by phrase structure grammars.

Theorem 1.20 *A language L is in $\mathcal{L}(RE)$ if and only if there is a (deterministic or non-deterministic)* Turing *machine $\mathcal{M}$ such that $\mathcal{M}$ accepts the language L (i.e., $T(\mathcal{M}) = L$ holds).* $\square$

We now give a modification of the Turing machine which can be used to characterize the languages generated by monotone grammars.

Definition 1.21 *A non-deterministic* TURING *machine is called a linearly bounded automaton if, for any w, the head position while working on the input w is restricted to the cells in which the letters of w are written, the cell before w and the cell after w.*

The Turing machine given in Example 1.19 is a linearly bounded automaton.

Theorem 1.22 *A language L is in $\mathcal{L}(CS)$ if and only if there is a linearly bounded automata $\mathcal{M}$ such that $\mathcal{M}$ accepts the language L (i.e., $T(\mathcal{M}) = L$).* $\square$

A decision problem is a question which only allows the answers "yes" or "no". Such a problem can be described as the language formed by all instances for which the answer is "yes".

Definition 1.23 *We say that a language L is decidable if there exists a deterministic* TURING *machine $\mathcal{M}$ such that*
— $L = T(\mathcal{M})$ and
— $\mathcal{M}$ halts on any input.

From the point of formal language theory the following problems are of special interest.

Membership Problem:	Given grammar/system G and word w, decide whether or not $w \in L(G)$.
Emptiness Problem:	Given grammar/system G, decide whether or not $L(G) = \emptyset$.
Finiteness Problem:	Given grammar/system G, decide whether or not $L(G)$ is a finite language.
Equivalence Problem:	Given grammars/systems G_1 and G_2, decide whether or not $L(G_1) = L(G_2)$.

The following theorem summarizes the results on the decidability status of these problems for phrase structure grammars and Lindenmayer systems.

Theorem 1.24 *The table of Figure 1.3 holds, where a + or – denotes that the problem (given in that column) is decidable or undecidable for the grammar/system type (given in the row) and t stands for triviality, i.e., the answer is always "yes".*

Theorem 1.25 *i) The membership problem for semi-linear sets is decidable.*
ii) For two semi-linear sets M_1 and M_2 (given by their sets of vectors), it is decidable whether or not $M_1 \subseteq M_2$ holds. $\square$

For the membership problem we give two results which give a statement on the complexity of the problem.

	membership problem	emptiness problem	finiteness problem	equivalence problem
arbitrary grammars	$-$	$-$	$-$	$-$
monotone grammars	$+$	$-$	$-$	$-$
context-free grammars	$+$	$+$	$+$	$-$
regular grammars	$+$	$+$	$+$	$+$
ET0L systems	$+$	$+$	$+$	$-$
EDT0L systems	$+$	$+$	$+$	$-$
E0L systems	$+$	$+$	$+$	$-$
T0L systems	$+$	t	$+$	$-$
DT0L systems	$+$	t	$+$	$-$
0L systems	$+$	t	$+$	$-$
D0L systems	$+$	t	$+$	$+$

Figure 1.3: Decidability status of some problems for phrase structure grammars and Lindenmayer systems.

Theorem 1.26 *i) Let G be a regular grammar. Then there exists an algorithm which decides whether or not $w \in L(G)$ with a time bound $O(|w|)$, i.e., there is a constant c such that the algorithm stops after at most $c|w|$ steps).*

ii) Let G be a context-free grammar. Then there exists an algorithm which decides whether or not $w \in L(G)$ with a time bound $O(|w|^3)$. □

Mostly we are not interested in the exact complexity of the problems. We only want to know whether or not the algorithm works with a polynomial time bound. Thus we introduce the following concepts.

Definition 1.27 *i) The set* **P** *is defined as the set of all languages which are decidable in polynomial time by deterministic* TURING *machines.*

ii) The set **NP** *is defined as the set of all languages which can be accepted in polynomial time by non-deterministic* TURING *machines.*

It is obvious by the definitions that $\mathbf{P} \subseteq \mathbf{NP}$. It is an open question whether or not equality between **P** and **NP** holds. In order to answer this question the so-called **NP**-complete problems are of interest.

Definition 1.28 *A language L is called* **NP***-complete if the following two conditions are satisfied:*
— $L \in \mathbf{NP}$ and
— any language $L' \in \mathbf{NP}$ can be polynomially transformed to L (i.e., there is a mapping h such that $h(w)$ can be computed with a polynomial time bound and $h(w) \in L$ holds if and only if $w \in L'$).

By the definition, the **NP**-complete languages can be considered as the "hardest" languages in **NP** (because the decision of $w \in L' \in \mathbf{NP}$ is not more complicated than the decision of $h(w) \in L$ up to an polynomial to compute $h(w)$).

The satisfiability problem known from logic is an example of an **NP**-complete problem/language. We present it here in a special form.

Theorem 1.29 *The problem 3-SAT defined as*

> *Given a finite set of disjunctions of three literals*[3]*,*
> *decide whether there is an assignment such that any disjunction gets true.*

is **NP**-*complete.* $\square$

The importance of the **NP**-complete language comes from the following statement.

Theorem 1.30 *The following assertions are equivalent:*
i) $\mathbf{P} = \mathbf{NP}$.
ii) All **NP**-*complete language are in* **P**.
iii) There is an **NP**-*complete language which is in* **P**.

Proof. i) $\Longrightarrow$ ii) and ii) $\Longrightarrow$ iii) are trivial.

Thus assume that iii) holds, i.e., there is an **NP**-complete language L which satisfies $L \in \mathbf{P}$. Let L' be an arbitrary language from **NP**. Then there is a function h computable in polynomial time such that $h(w) \in L$ if and only if $w \in L$. Therefore we can decide whether $w \in L'$ by a computation of $h(w)$ and a decision whether $h(w) \in L$. Since both can be done in polynomial time, we have $L \in \mathbf{P}$. This implies $\mathbf{NP} \subseteq \mathbf{P}$, which gives i). $\square$

The following theorem gives a method to prove that a certain problem or equivalent a certain language L is **NP**-complete. It is called reduction.

Theorem 1.31 *If a language L' is a* **NP**-*complete and L' can be polynomially transformed to $L \in \mathbf{NP}$, then L is* **NP**-*complete, too.*

Proof. By supposition $L \in \mathbf{NP}$ holds. Therefore we only have to show that any language $L'' \in \mathbf{NP}$ can be polynomially transformed to L. Since L' is **NP**-complete, there is a polynomial transformation h from L'' to L', and by supposition, there is a polynomial transformation from L' to L. Then the composition $h \circ g$ gives a polynomial transformation from L'' to L. $\square$

[3]A literal is a variable or a negated variable.

Chapter 2

Chain Code Picture Languages

2.1 Chain Code Pictures

We consider the grid over the integers. In terms of sets, we regard $\mathbb{Z} \times \mathbb{Z}$. With any point $z = (m, n) \in \mathbb{Z} \times \mathbb{Z}$, we associate its four neighbours which are given by

$$u(z) = (m, n + 1),\ d(z) = (m, n - 1),\ r(z) = (m + 1, n),\ l(z) = (m - 1, n)$$

where we use the notations u, d, r and l, because we have to go *u̲p*, *d̲own*, *r̲ight* and *l̲eft* to reach the neighbouring point.

By π we denote the set formed by the four directions, i.e., $\pi = \{u, d, r, l\}$. For any $b \in \pi$, we designate the direction opposite to b by $\overline{b}$. Thus we have $\overline{u} = d$, $\overline{d} = u$, $\overline{r} = l$ and $\overline{l} = r$.

By a *unit line* we understand a finite part of a straight line connecting two neighbouring points. Therefore, for any unit line, there are a point $z \in \mathbb{Z} \times \mathbb{Z}$ and a direction $b \in \{u, d, r, l\}$ such that the unit line connects z and $b(z)$. We denote this unit line by $(z, b(z))$. From the geometrical point of view, $(z, b(z))$ is equal to $(b(z), z)$.

A *picture* is a finite set of unit lines. An example of a picture is presented in Figure 2.1. The picture given in Figure 2.1 consist of the unit lines

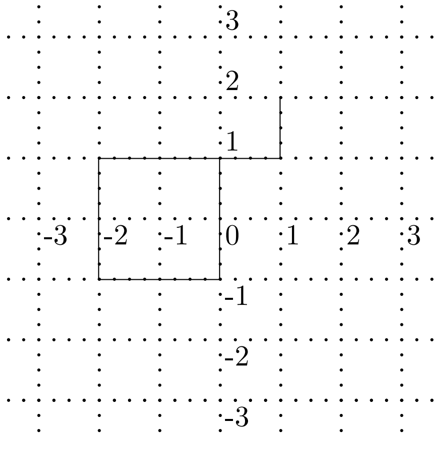

Figure 2.1: Example of a picture consisting of ten unit lines

$$\begin{aligned}&((1,2),(1,1)),\ ((1,1),(0,1)),\ ((0,1),(-1,1)),\ ((-1,1),(-2,1)),\\ &((-2,1),(-2,0)),\ ((-2,0),(-2,-1)),\ ((-2,-1),(-1,-1)),\\ &((-1,-1),(0,-1)),\ ((0,-1),((0,0)),\ ((0,0),(0,1)).\end{aligned} \tag{2.1}$$

Given a picture p, we define the set $V(p)$ of its points as

$$V(p) = \{z \mid (z,z') \in p \text{ or } (z',z) \in p\}.$$

For the picture of Figure 2.1, we obtain the point set

$$\{(1,2),\ (1,1),\ (0,1),\ (-1,1),\ (-2,1),\ (-2,0),\ (-2,-1),\ (-1,-1),\ (0,-1),\ (0,0)\}.$$

In order to give the picture of Figure 2.1, we have presented its unit lines in (2.1) by a special enumeration. Obviously, any other enumeration of the unit lines different from that given in (2.1) describes the picture of Figure 2.1, too. This situation changes if we are not only interested in the picture, but in its drawing/construction, too. The above enumeration presents a drawing of the picture where we start in the point $(1,2)$, go for every unit line (z,z') from z to z', and end in the point $(0,1)$. Clearly, for a given picture there exist some different drawings. For instance, we can draw the picture of Figure 2.1 by starting in $(0,1)$, following the unit lines in the order reverse to the enumeration above and from z' to z for a unit line (z,z') and finishing in the point $(1,2)$. Moreover, it is not necessary, that any unit line occurs exactly once in the drawing process as it is in the case of a set representation. For instance, if we start in $(0,1)$ and draw the unit lines in the order of (2.2), then we stop in $(0,1)$ and have drawn the picture of Figure 2.1, again:

$$\begin{aligned}&((0,1),(1,1)),\ ((1,1),(1,2)),\ ((1,2),(1,1)),\ ((1,1),(0,1)),\\ &((0,1),(-1,1)),\ ((-1,1),(-2,1)),((-2,1),(-2,0)),\ ((-2,0),(-2,-1)),\\ &((-2,-1),(-1,-1)),\ ((-1,-1),(0,-1)),\ ((0,-1),((0,0)),\ ((0,0),(0,1)).\end{aligned} \tag{2.2}$$

Therefore we say that a sequence

$$(z_0,z_1),(z_1,z_2),(z_2,z_3),\dots,(z_{r-2},z_{r-1}),(z_{r-1},z_r)$$

is a drawing of a picture p with start point z_0 and end point z_r if and only if, for $1 \le i \le r$, there are $b_i \in \pi$ such that $z_i = b_i(z_{i-1})$ and $p = \bigcup_{i=1}^{r}(z_{i-1},z_i)$.

Obviously, if a picture p can be drawn, then it is *connected*, i.e., for any two points $z,z' \in V(p)$ there is a path from z to z' along unit lines belonging to p.

If we are interested in the drawing of a picture p, we do not present the whole drawing process, but we mention the start point and the end point of the drawing. Thus a drawn picture is a triple

$$(z,p,z') \text{ with a basic picture } p \text{ and } z,z' \in V(p).$$

Therefore the picture p of Figure 2.1 can be described as the drawn picture $((1,2),p,(0,1)$, if we use the drawing corresponding to (2.1).

Given a drawn picture $q=(z,p,z')$, then we define the *shift* of q by $sh(q) = z' - z$ (where z and z' are considered to be elements of $\mathbb{N}^2$)

We now define two relations on basic and drawn pictures.

We say that two basic pictures p and p' are *equivalent* if and only if there are integers m and n such that $(z, z') \in p$ if and only if $(z + (m, n), z' + (m, n)) \in p'$.

We say that two drawn pictures $q = (z_1, p, z_2)$ and $q' = (z'_1, p', z'_2)$ are *equivalent* if and only if there are integers m and n such that

$$(z, z') \in p \text{ if and only if } (z + (m, n), z' + (m, n)) \in p',$$
$$z'_1 = z_1 + (m, n) \text{ and } z'_2 = z_2 + (m, n)$$

We write $p \equiv_b p'$ and $q \equiv_d q'$ if the basic pictures p and p' and the drawn pictures q and q' are equivalent, respectively. If it is clear from the context that basic or drawn pictures are considered, we omit the indices b and d sometimes.

It is easy to see that $\equiv_b$ and $\equiv_d$ are equivalence relations on the set of basic and drawn pictures, respectively. We give a short proof for the case of basic pictures.

Reflexivity: Because $(z, z') = (z + (0, 0), z' + (0, 0))$, we have $p \equiv_b p$.

Symmetry: Because $(z, z') \in p$ if and only if $(z + (m, n), z' + (m, n)) \in p'$ implies $(u, u') \in p'$ if and only if $(u + (-m, -n), u' + (-m, -n)) \in p$, we obtain that $p \equiv_b p'$ implies $p' \equiv_b p$.

Transitivity: Let $p \equiv_b p'$ and $p' \equiv_b p''$. Then there are integers m, n, m' and n' such that $(z, z') \in p$ if and only if $(z + (m, n), z' + (m, n)) \in p'$ and $(u, u') \in p'$ if and only if $(u + (m', n'), u' + (m', n')) \in p''$. Therefore $(z, z') \in p$ if and only if $(z + (m, n) + (m', n'), z' + (m, n) + (m', n')) \in p''$, i.e., $p \equiv_b p''$.

The equivalence classes of the basic picture p and the drawn picture q are denoted by $[p]$ and [q], respectively.

As word languages are sets of words over some alphabet, we say that a set of drawn (or basic) pictures is a drawn (or basic) picture language.

In the case of basic pictures we are not interested in the (exact) place of the picture in the grid. Thus we intuitively assume that all pictures of $[p]$ are contained in the language if we have the basic picture p in the language. Thus we define the equality of two basic picture languages L and L' by

> $L = L'$ if and only if, for any $p \in L$ and any $q \in L'$, there are pictures $p' \in L'$ and $q' \in L$ such that $p \equiv_b p'$ and $q \equiv_b q'$.

Equivalently, $L = L'$ iff $\bigcup_{p \in L}[p] = \bigcup_{q \in L'}[q]$.

In order to present a drawn picture we can also give the start point and the end point and the directions in which we have to draw, i.e., instead of giving the line $(z, b(z))$, $b \in \pi$, we present the direction b itself. The picture of Figure 2.1 with start point $(1, 2)$, end point $(0, 1)$ and drawn according to given (2.1) can be given by the word *dllllddrruu*. Formally, with any drawing

$$(z_0, z_1), (z_1, z_2), \ldots (z_{r-1}, z_r) \text{ , where } z_i = b_i(z_{i-1}) \text{ for } 1 \leq i \leq r,$$

of a picture p, we associate the word $b_1 b_2 \ldots b_r \in \pi^*$. Since the drawing remembers to a chain (the end point of a unit line is the start point of the following unit line), $b_1 b_2 \ldots b_r$ is called the chain code of the given drawing.

Conversely, with any word $w \in \pi^*$, we now associate a drawn picture $dccp(w)$ by the following inductive settings:

- if $w = \lambda$, then $dccp(w) = ((0,0), \emptyset, (0,0))$, and
- if $w = w'b$, $w' \in \pi^*$, $b \in \pi$ and $dccp(w') = ((0,0), p, z)$, then $dccp(w) = ((0,0), p \cup \{(z, b(z))\}, b(z))$.

Moreover, if $dccp(w) = ((0,0), p, z)$, then we define the basic picture associated with w by $bccp(w) = p$.

Here b and d stand for basic and drawn, respectively, and *ccp* is an abbreviation for chain code picture. In the sequel we shall use the term *basic (drawn) chain code picture* for any connected basic (drawn) picture p since there is a word $w \in \pi^*$ such that $p \in [bccp(w)]$ ($p \in [dccp(w)]$).

For a word $w \in \pi^*$, we set $sh(w) = sh(dccp(w))$. It is easy to see that, for $w_1 \in \pi^*$ and $w_2 \in \pi^*$,

$$sh(w_1w_2) = sh(w_1) + sh(w_2). \tag{2.3}$$

With any chain code picture we now associate the sets of its descriptions by words over π.

Definition 2.1 *For a drawn chain code picture q and and basic chain code picture p, we set*

$$des(q) = \{w \in \pi^* \mid q \in [dccp(w)]\}$$

and

$$des(p) = \{w \in \pi^* \mid p \in [bccp(w)]\}.$$

Example 2.2 Let $q = ((0,0), \{((0,0),(0,1)), ((0,1), ((0,2))\}, (0,2))$, i.e., q has the start point $(0,0)$, consists of the first unit line of the y-axis in positive direction, and the endpoint is $(0,2)$. Then any drawing of q has to start in the origin, any prefix of the drawing process has to lead to $(0,0)$ or $(0,1)$ or $(0,2)$ and finally we have to reach $(0,2)$. Thus $des(q)$ consists of all words w which satisfy the following properties:
– for any prefix v of w, i.e., $w = vv'$ for some v', $0 \le \#_u(v) - \#_d(v) \le 2$, and
– $\#_u(w) = \#_d(w) + 2$. $\diamond$

We now prove that the sets of descriptions of a chain code picture are regular.

Theorem 2.3 *For a drawn chain code picture q and and basic chain code picture p, $des(q)$ and $des(p)$ are regular languages.*

Proof. First, we give the proof for the case of drawn pictures. Let (z, p, z') be a drawn picture. We construct the regular grammar

$$G = (V(p) \times 2^p, \pi, P, (z, \emptyset))$$

where

$$P = \{(u, p') \to b(b(u), p' \cup \{(u, b(u)\}) \mid u \in V(p), p' \subseteq p, (u, b(u)) \in p\} \cup \{(z', p) \to \lambda\}.$$

The grammar simulates a drawing of p. The first component of a nonterminal gives the point of $V(p)$ reached by the derivation and the second component remembers the drawn part of p. The simulation starts with $(z, \emptyset)$, i.e., in the start point of p and nothing is drawn. If the nonterminal is (u, p'), i.e., the point u is reached and the subset p' of lines of p is drawn, then the grammar produces a letter b if and only if the drawing can be continued by going in the direction b, which is only possible if $(u, b(u))$ belongs to the picture, and it reaches the state $(b(u), p' \cup \{(u, b(u))\})$, i.e., the drawing reaches $b(u)$ and the drawn unit line $(u, b(u))$ is added to the set of drawn lines. A derivation can only terminate if and only if the rule $(z', p) \to \lambda$ is applied, i.e., if the end point is reached and the whole picture p is drawn. □

Let $w = w_1 b\bar{b}bw_2$ with $b \in \pi$ be a word over π. Obviously, we have $dccp(w) = dccp(w_1bw_2)$ and $bccp(w) = bccp(w_1bw_2)$. Thus in some situations we can cancel special subwords without a change of the picture. In the sequel we shall use this fact sometimes. Hence we study it as an operation. Let

$$Ret = \{ud, du, lr, rl\}$$

be the set of retreats.

A word w is called *retreat-free*, if no word of Ret is a subword of w.

Definition 2.4 *i) For a word $w \in \pi^*$, we define the retreat deletion image $red(w)$ inductively as follows:*
(1) $w \in red(w)$,
(2) If $z \in red(w)$ and $z = z_1sz_2$ for some $s \in R$, then $z_1z_2 \in red(w)$.
(3) A word belongs to $red(w)$ if and only if it is constructed by steps (1) or (2).
ii) For a language $L \subseteq \pi^*$, we set $red(L) = \bigcup_{w\in L} red(w)$.

Intuitively, the set $red(w)$ consists of all words which can be obtained from w by an iterated deletion of retreats. We denote such a deletion step from z_1sz_2 with $s \in Ret$ to z_1z_2 by $w_1sw_2 \vdash w_1w_2$ and its transitive and reflexive closure by $\vdash^*$.

Example 2.5 Let D_1 be the language generated by context-free grammar

$$G = (\{S\}, \pi, \{S \to SS,\ S \to uSd,\ S \to \lambda\}, S).$$

D_1 is the set of all correctly bracketed expressions over the "opening bracket" u and the "closing bracket" d (see Example 1.4). We prove that $red(D_1) = D_1$. Obviously, by Definition 2.4, part (1), we have $D_1 \subseteq red(D_1)$. Now let $w \in D_1$ be a non-empty word. Let $w = w_1udw_2$. Clearly, the mentioned occurrence of u and d form a pair of corresponding "brackets". Therefore the deletion of the retreat ud leads to w_1w_2 which belongs to D_1. Now assume that we cancel an occurrence of the retreat du, i.e, $w = v_1duv_2$. Then there is an opening u belonging to d and a closing d belonging to u. Therefore $w = v_1'uv_2'duv_3'dv_4$ such that $v_1'v_4'$, v_2' and v_3' are correctly bracketed. Therefore $v_2'v_3'$ and $v_1'uv_2'v_3'dv_4'$ are also correctly bracketed and thus the word obtain from w by deletion of the mentioned retreat du belongs to D_1. ◇

Lemma 2.6 *Let $w \in \pi^*$.*
i) Then $red(w)$ contains exactly one retreat-free word.
ii) For any word $z \in red(w)$, $sh(z) = sh(w)$.

Proof. i) We give a proof by induction on the length of the word.

Induction basis: If $|w| \leq 1$, then w contains no retreat and thus $red(w) = \{w\}$.

Induction step: Let $|w| \geq 2$. If w is retreat-free, then $red(w) = \{w\}$ and the statement holds trivially. There we assume that $w = w_1abw_2$ with $ab \in Ret$. Let $w \vdash^* z$ where z is retreat-free. Then z does not contain the occurrence of ab. Assume that ab is cancelled at some iteration step. Then we have

$$w = w_1abw_2 \vdash^* w_1'abw_2' \vdash w_1'w_2' \vdash^* z \text{ with } w_1 \vdash^* w_1' \text{ and } w_2 \vdash^* w_2'.$$

Then we have

$$w = w_1abw_2 \vdash w_1w_2 \vdash^* w_1'w_2' \vdash^* z,$$

too. Now let us assume that ab is not deleted, but the b is cancelled sometimes. Then we have

$$w = w_1abw_2 \vdash^* w_1'abw_2' = w_1'abaw_2'' \vdash w_1'aw_2'' \vdash^* z \text{ with } w_1 \vdash^* w_1' \text{ and } w_2 \vdash^* w_2',$$

which implies the existence of

$$w = w_1abw_2 \vdash w_1w_2 \vdash^* w_1'w_2' = w_1'aw_2'' \vdash^* z.$$

Analogously, we handle the case that ab is not cancelled, but a is cancelled sometimes. Thus in all cases we can derive the a retreat-free word z by starting with a cancellation of ab. Therefore $red(w)$ and $red(w_1w_2)$ contain the same retreat-free words. By induction hypothesis, $red(w_1w_2)$ contains only one retreat-free word. Thus $red(w)$ contains only one retreat-free word, too.

ii) By (2.3), $sh(w_1abw_2) = sh(w_1w_2)$ for any $ab \in Ret$. By iterated application of this fact we get immediately the statement. □

Definition 2.7 *i) For $w \in \pi^*$, let $ref(w)$ be the only retreat-free word in $red(w)$.*

ii) For a language $L \subseteq \pi^*$, we set $ref(L) = \{ref(w) \mid w \in L\}$.

For the language D_1 given in Example 2.5, we get $ref(D_1) = \{\lambda\}$.

Lemma 2.8 *$ref(\pi^*)$ is a regular language.*

Proof. Obviously, $ref(\pi^*)$ is the set of all retreat-free words. Therefore

$$\pi^* \setminus \{z_1sz_2 \mid z_1, z_2 \in \pi^*,\ s \in Ret\} = ref(\pi^*).$$

The regular grammar $G = (\{S, A\}, \pi, P, S)$ with

$$\begin{aligned} P &= \{S \to bS \mid b \in \pi\} \cup \{S \to sA \mid s \in Ret\} \\ &\quad \cup \{A \to bA \mid b \in \pi\} \cup \{A \to \lambda\} \end{aligned}$$

generates $\{z_1sz_2 \mid z_1, z_2 \in \pi^*,\ s \in Ret\}$. By the closure of $\mathcal{L}(REG)$ under complement (see Theorem 1.14, we obtain that $ref(\pi^*)$ is regular, too. □

Lemma 2.9 $D_\pi = \{w \mid w \in \pi^*,\ ref(w) = \lambda\}$ *is context-free.*

Proof. It is easy to see that the context-free grammar $G = (\{S\}, \pi, P, S)$ with

$$P = \{S \to SS,\ S \to uSd,\ S \to dSu,\ S \to rSl,\ S \to lSr,\ S \to \lambda\}$$

generates the set D_π (see Example 1.4). □

Theorem 2.10 *i) If $L \subseteq \pi^*$ is a regular language, then $red(L)$ and $ref(L)$ are regular, too (i.e., $\mathcal{L}(REG)$ is closed under the operations red and ref).*
ii) $\mathcal{L}(CF)$ and $\mathcal{L}(CS)$ are not closed under red and ref.

Proof. i) Let $L = L(G)$ for some regular grammar $G = (N, \pi, P, S)$. For $A \in N$ and $B \in N$ we define the sets

$$L_{A,B} = \{w \mid w \in D_\pi \text{ and } A \Longrightarrow^* wB \text{ is a derivation in } G\}$$

and

$$L_A = \{w \mid w \in D_\pi \text{ and } A \Longrightarrow^* w \text{ is a derivation in } G\}.$$

It is easy to see that

$$L_{A,B} = D_\pi \cap L(G_{A,B}),$$

where

$$G_{A,B} = (N, \pi, (P \setminus \{A \to w \mid w \in \pi^*\}) \cup \{B \to \lambda\}, A)$$

(the only terminating derivation in $G_{A,B}$ are of the form $A \Longrightarrow^* wB \Longrightarrow w$). Because $G_{A,B}$ is a regular grammar, $L_{A,B}$ is an intersection of a context-free language (see Lemma 2.9) and a regular language. By the closure properties of $\mathcal{L}(CF)$ (see Theorem 1.12), the emptiness of $L_{A,B}$ is decidable (see Theorem 1.24). Analogously, we can show that the emptiness L_A is decidable.

We construct now the the regular grammar

$$H = (N, \pi, P \cup \{A \to B \mid L_{A,B} \neq \emptyset\} \cup \{A \to \lambda \mid L_A \neq \emptyset\}, S).$$

Now let $w \in L$ and $v \in red(w)$. Then $w = w_0v_1w_1v_2w_2 \dots v_nw_n \vdash^* v_1v_2 \dots v_n = v$ by an cancellation of the words $w_0, w_1, \dots, w_n$ by iterated deletions of retreats, i.e., $w_i \in D_\pi$ for $0 \leq i \leq n$. Since $w \in L$ we have a derivation

$$\begin{aligned} S = A_0 &\Longrightarrow^* w_0B_0 \Longrightarrow^* w_0v_1A_1 \Longrightarrow^* w_0v_1w_1B_1 \\ &\Longrightarrow^* w_0v_1w_1v_2A_2 \Longrightarrow^* w_0v_1w_1v_2w_2B_2 \Longrightarrow^* \dots \\ &\Longrightarrow^* w_0v_1w_1 \dots w_{n-1}B_{n-1} \Longrightarrow^* w_0v_1w_1 \dots w_{n-1}v_nA_n \\ &\Longrightarrow^* w_0v_1w_1 \dots w_{n-1}v_nw_n \end{aligned}$$

in G. Thus we get $w_i \in L_{A_i,B_i}$ for $0 \leq i \leq n-1$ and $w_n \in L_{A_n}$. Therefore we have the following derivation in H

$$S = A_0 \Longrightarrow B_0 \Longrightarrow^* v_1A_1 \Longrightarrow v_1B_1 \Longrightarrow^* v_1v_2A_2 \Longrightarrow \dots \Longrightarrow^* v_1v_2 \dots v_nA_n \Longrightarrow v_1v_2 \dots v_n.$$

Hence $v_1 v_2 \dots v_n = v \in L(H)$.

By analogous arguments one can show that $v \in L(H)$ implies $v \in red(w)$ for some $w \in L$.

Combining these relations we get $red(L) = L(H)$. Thus $red(L)$ is regular.

The statement for $ref(L)$ follows from $ref(L) = red(L) \cap ref(\pi^*)$, Lemma 2.8 and the closure properties of $\mathcal{L}(REG)$ (see Theorem 1.12).

ii) Let us assume that $ref(U)$ is context-free for any context-free language U.

The context-free grammar $G = (\{S\},\ \pi,\ \{S \to u^2Sd,\ S \to uld\},\ S)$ generates the language $L' = \{u^{2i}ld^i \mid i \geq 1\}$. We consider the language $L = (L')^+$ which is context-free by the closure properties of $\mathcal{L}(CF)$ (see Theorem 1.12). By definition,

$$L = \{u^{2i_1}ld^{i_1}u^{2i_2}ld^{i_2} \dots u^{2i_n}ld^{i_n} \mid n \geq 1, i_j \geq 1 \text{ for } 1 \leq j \leq n\}.$$

Let

$$K' = \{u^m l^n d \mid m \geq 1, n \geq 1\} \text{ and } K = ref(L) \cap K'.$$

Because K' is regular (it is left to the reader to construct a regular grammar generating K'), K is context-free by our assumption and the closure properties of $\mathcal{L}(REG)$ (see Theorem 1.12).

On the other hand, $ref(u^{2i_1}ld^{i_1}u^{2i_2}ld^{i_2} \dots u^{2i_n}ld^{i_n})$ is in K' if and only if $i_n = 1$ and $2i_{j-1} = i_j$ for $2 \leq i \leq n$ (we can delete only retreats du between two occurrences of l, and no u and no d remains between two occurrences of l). Thus $i_{n-1} = 2,\ i_{n-2} = 4$ and so on and finally $i_1 = 2^n$. Hence

$$ref(u^{2i_1}ld^{i_1}u^{2i_2}ld^{i_2} \dots u^{2i_n}ld^{i_n}) = u^{2^n}l^n d.$$

This implies

$$K = \{u^{2^n}l^n d \mid n \geq 1\}.$$

Let k be the constant of the pumping lemma for context-free languages (see Theorem 1.7 b)) and $n > k$. Then $u^{2^n}l^n d \in L$ and $|u^{2^n}l^n d| > k$. Therefore there are z, v, w, x, y such that $u^{2^n}l^n d$=$zvwxy$, $vx \neq \lambda$, $|vwx| \leq k$ and $t = zv^2wx^2y \in K$. If v contains two different letters, then v^2 contains a subword lu or dl which is impossible by $t \in K$. If $v = d$, then we get a contradiction because t contains two times the letter d. These two fact hold for x, too. Therefore we have the following three cases:

Case 1: $v = u^r$ and $x = u^s$ and $r + s > 0$. Then $t = u^{2^n+r+s}l^n d$ which contradicts $t \in K$ because $2^n + r + s > 2^n$.

Case 2: $v = u^r$ and $x = l^s$ and $r + s > 0$. Then $t = u^{2^n+r}l^{n+s}d$. If $s > 0$, then $2^n + r < 2^n + 2^n = 2^{n+1} \leq 2^{n+s}$ (because $r \leq |vwx| \leq k < n < 2^n \leq$) in contradiction to $t \in K$. If $s = 0$, then $r > 0$ and $2^n + r > 2^n = 2^{n+s}$ which is contradiction to $t \in K$, again.

Case 3: $v = l^r$ and $x = l^s$ and $r + s > 0$. Then $t = u^{2^n}l^{n+r+s}d$ which contradicts $t \in K$ because $2^n < 2^{n+r+s}$.

Since we get a contradiction in all cases, our assumption is false. Hence there is a context-free language U such that $ref(U)$ is not context-free.

Let U be a context-free language such that $ref(U)$ is not context-free. Assume that $red(U)$ is context-free. Then $ref(U) = red(U) \cap ref(\pi^*)$ is context-free by the regularity of $ref(\pi^*)$ (see Lemma 2.8) and the closure properties of $\mathcal{L}(CF)$ (see Theorem 1.12) which contradicts our choice of U. $\square$

2.2 Hierarchy of Chain Code Picture Languages

In the preceding section we have associated a chain code picture with any word over the alphabet π. We now extend this idea to languages. For any language $L \subseteq \pi^*$, we set

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

Especially we are interested in the sets of pictures which are generated by phrase structure grammars or Lindenmayer systems. Let G be a phrase structure grammar or ET0L system such that $L(G) \subseteq \pi^*$, i.e., that the set of terminals of G is π. Then we set

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

Example 2.11 i) Let the regular grammar

$$G_1 = (\{S\},\ \pi,\ \{S \to urdluS,\ S \to urdlu\}, S)$$

be given. Then we have

$$L(G_1) = \{(urdlu)^n \mid n \geq 1\} = \{urdlu, urdluurdlu, urdluurdluurdlu, \dots\}\,.$$

Consequently, the drawn picture language $dccp(G_1)$ consists of the pictures shown in Figure 2.2, where the startpoint $(0,0)$ is marked by a circle and the endpoint by a square. Thus the chain code picture language $bccp(G_1)$ consists of all towers of unit squares with an arbitrary height.

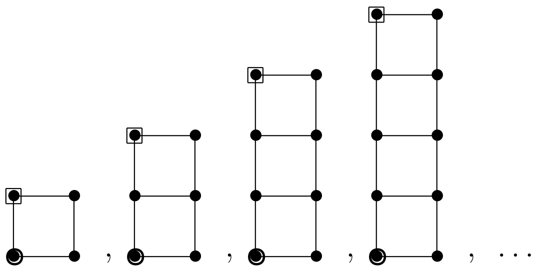

Figure 2.2: Pictures of the language $dccp(G_1)$

ii) We consider the context-free grammar

$$G_2 = (\{S, A\},\ \pi,\ \{S \to lA,\ A \to urlAd,\ A \to urd\},\ S)$$

The word language generated by G_2 is

$$L(G_2) = \{l(url)^n urdd^n \mid n \geq 0\}\,.$$

The corresponding drawn pictures are given in Figure 2.3. Obviously, the pictures of $dccp(G_2)$ differ from those of $dccp(G_1)$ only in the start and end points. Hence we get $bccp(G_2) = bccp(G_1)$.

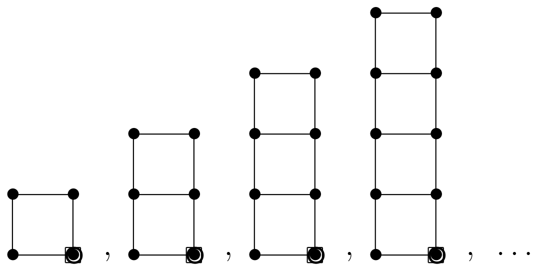

Figure 2.3: Pictures of the language $dccp(G_2)$

iii) Let the context-free grammar

$$G_3 = (\{S, A\},\ \pi,\ \{S \to SrrS,\ S \to lA,\ A \to urlAd,\ A \to urd\}$$

be given. Then

$$\begin{aligned} L(G_3) \ &= \ \{l(url)^{n_1}urdd^{n_1}rrl(url)^{n_2}urdd^{n_2}rr \ldots l(url)^{n_{k-1}}urdd^{n_{k-1}}rrl(url)^{n_k}urdd^{n_k} \mid \\ &\qquad k \geq 1,\ n_j \geq 0 \text{ for } 1 \leq j \leq k\} \end{aligned}$$

(because by the first rule we generate words of the form $(Srr)^{k-1}S$ and then from any S we derive words $l(url)^n urdd^n$ as in part ii)). Consequently, the pictures generated by G_3 are some towers of unit squares (of arbitrary height) (see part ii)) with a distance of one unit line and connected in the base line. A typical picture of $bccp(G_3)$ is shown in Figure 2.4.

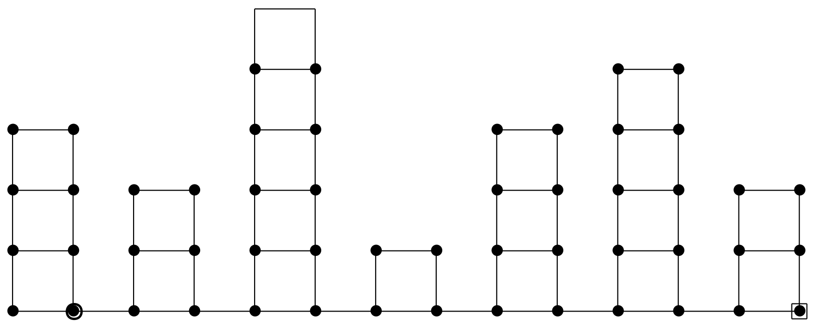

Figure 2.4: A typical picture of $bccp(G_3)$

iv) We consider the context-free grammar

$$G_4 = (\{S\},\ \pi,\ \{S \to urdlS,\ S \to ldruS,\ S \to urdl,\ S \to ldru\},\ S)\,.$$

then the generated word language is

$$L(G_4) = \{w_1 w_2 \ldots w_n \mid n \geq 1, w_i \in \{urdl, ldru\} \text{ for } 1 \leq i \leq n\}\,.$$

If $w \in \{urdl, ldru\}$, then $dccp(w)$ is the unit square in the first or third quadrant which contains the origin. Hence $dccp(G_4)$ consists of the three pictures given in Figure 2.5.

We mention that $bccp(G_4)$ only consists of two pictures since the two unit squares of $dccp(G_4)$ are in the same equivalence class of $\equiv_b$. $\diamond$

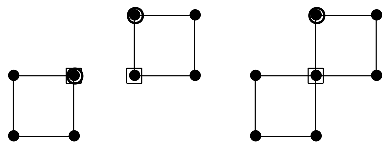

Figure 2.5: Pictures of $dccp(G_4)$

We now classify the chain code picture languages in analogy to the Chomsky classification for word languages.

Definition 2.12 *A chain code picture language B is called regular or context-free or monotone or recursively enumerable if there is a regular or context-free or monotone grammar or a phrase structure grammar, respectively, such that $B = bccp(L(G))$.*

By $\mathcal{CCP}(REG)$, $\mathcal{CCP}(CF)$, $\mathcal{CCP}(CS)$, and $\mathcal{CCP}(RE)$, we denote the families of all regular, context-free, monotone and recursively enumerable basic chain code picture languages.

Theorem 1.11 gives a hierarchy of the families of word languages. Our aim is to give an analogous result for picture languages. First we mention that $L \notin \mathcal{L}(X)$ for some family X of grammars does not imply that $bccp(L) \notin \mathcal{CCP}(X)$. This can be seen from the grammars G_1 and G_2 given in Example 2.11. We know that $L(G_2)$ is a context-free languages, which is not regular (one can use the pumping lemma to prove the non-regularity of $L(G_2)$). But this does not imply $bccp(G_2) \notin \mathcal{CCP}(REG)$ because we have $bccp(G_2) = bccp(G_1)$ and G_1 is a regular grammar, i.e., $bccp(G_1) \in \mathcal{CCP}(REG)$.

Theorem 2.13 $\mathcal{CCP}(REG) \subset \mathcal{CCP}(CF) \subset \mathcal{CCP}(CS) = \mathcal{CCP}(RE)$

Proof. i) $\mathcal{CCP}(REG) \subset \mathcal{CCP}(CF)$.

The inclusion $\mathcal{CCP}(REG) \subseteq \mathcal{CCP}(CF)$ holds by the definition of the families.

Let $L = \{ru^nrd^nr \mid n \geq 1\}$. L is context-free since it is generated by the context-free grammar with the rules $S \to rAr$, $A \to uAd$ and $A \to r$. Thus $bccp(L) \in \mathcal{CCP}(CF)$.

The pictures of $bccp(L)$ consist of two parallel vertical lines with the distance 1 which are connected by a unit line at the upper end of the lines, and there is a unit line to the left and right at the lower point of the left and right line, respectively. Let q and q' be the end points of the lower lines which are not at the vertical lines. Obviously, the degree of q und q' is 1. Let w be a word which describes such a picture, i.e., $bccp(w) \in L$. Then the drawing according to w starts somewhere and reaches one of the points q and q', say q, before the other one. Therefore $w = w_1w_2w_3$ where w_2 corresponds to a drawing of the whole picture starting in q and finishing in q'. Let $b\bar{b}$ be a retreat. If $b\bar{b}$ occurs in w_2, i.e., $w_2 = w_2'b\bar{b}w_2''$, then $w_2'w_2''$ describes also a drawing of the picture from q to q'. Therefore $ref(w_2) = lu^nld^nl$ (we start in the right point) or $ref(w_2) = ru^nrd^nr$ (we start in the left point), where n is the length of the vertical lines.

Let the homomorphism h be given by $h(r) = h(l) = r$, $h(u) = u$ and $h(d) = d$. Then $h(ru^nrd^nr) = h(lu^nld^nl) = ru^nrd^nr$.

Let us assume, that there is a regular grammar G such that $bccp(G) = bccp(L)$. From the above considerations we get

$$K = h(ref(sub(L(G))) \cap \{ru^nrd^mr \mid n \geq 1, m \geq 1\} = \{ru^nrd^nr \mid n \geq 1\}\,.$$

Since $\{ru^nrd^mr \mid n \geq 1, m \geq 1\} = \{r\}\{u\}^*\{r\}\{d\}^*\{r\}$, it is a regular language by Theorem 1.15. Therefore K is a regular language by the closure properties of $\mathcal{L}(REG)$ (see Theorems 1.12 and 1.16) and Theorem 2.10 i). On the other hand, by the pumping lemma (see Theorem 1.7) it is easy to prove that $\{ru^nrd^nr \mid n \geq 0\} \notin \mathcal{L}(REG)$ (see part iii) of the proof of Theorem 1.11).

This contradiction proves $bccp(L) \notin \mathcal{CCP}(REG)$.

ii) $\mathcal{CCP}(CF) \subset \mathcal{CCP}(CS)$

The inclusion $\mathcal{CCP}(CF) \subseteq \mathcal{CCP}(CS)$ follows from the definitions, again.

Let $L' = \{rru^{2^n}rr \mid n \geq 0\}$. Since L' is a monotone language (see Example 1.4 ii), $bccp(L') \in \mathcal{CCP}(CS)$.

Any picture of $bccp(L')$ consists of a vertical line of length 2^n for some $n \geq 1$ and at each end point a horizontal line of length 2 is added. As in the preceding proof any word w with $bccp(w) \in bccp(L')$ contains a subword w_2 which corrresponds to a drawing from one end point of the picture to the other end point. Clearly, $w_2 = rw_2'r$ or $w_2 = lw_2'l$. Moreover, in the former case $\#_u(w_2) - \#_d(w_2) = 2^n$ and in the latter case $\#_d(w_2) - \#_u(w_2) = 2^n$ since we have to draw the complete vertical line.

Let us assume that there is a context-free grammar G' such that $bccp(G') = bccp(L')$. Let

$$K_1 = sub(L(G')) \cap \{rzr \mid z \in \pi^+\} \text{ and } K_2 = sub(L(G')) \cap \{lzl \mid z \in \pi^+\}\,.$$

By the closure of $\mathcal{L}(CF)$ under sub and intersections by regular sets (see Theorems 1.16 and 1.12), K_1 and K_2 are context-free languages. Let h' and h'' be the homomorphisms defined by $h'(u) = u$, $h'(d) = d$, $h'(r) = h'(l) = \lambda$ and $h''(u) = d$, $h''(d) = u$, $h''(r) = h''(l) = \lambda$. Again, by Theorem 1.12, $K' = h'(K_1) \cup h''(K_2) \in \mathcal{L}(CF)$.

On the other hand,
– if $v \in K'$, then $\#_u(v) - \#_d(v) = 2^n$ for some $n \geq 0$, and
– for any $n \geq 0$, there is a $v \in K'$ such that $\#_u(v) - \#_d(v) = 2^n$.
This implies that,
– for $v \in K'$, $\Psi(v) = (2^n + m, m)$ for some $n \geq 1$ and $m \geq 1$, and
– for any $n \geq 0$, there are a $v \in K'$ and an $m \geq 0$ such that $\Psi(v) = (2^n + m, m)$. Now it is easy to see that $\Psi(K')$ is not semi-linear (the adding of a vector (x, y) to $(2^n + m, m)$ with sufficient large n do not give a vector of the form $(2^{n'} + m', m')$). Hence by Theorem 1.8, K' is not a context-free language.

This contradiction proves the falsity of our assumption and thus $bccp(L) \notin \mathcal{CCP}(CF)$.

iii) $\mathcal{CCP}(CS) = \mathcal{CCP}(RE)$

By the definition of the families of picture languages, $\mathcal{CCP}(CS) \subseteq \mathcal{CCP}(RE)$.

We now prove the converse inclusion. Let $L \in \mathcal{CCP}(RE)$. Then there is a phrase structure grammar $H = (N, T, P, S)$ such that $bccp(G) = L$ and all rules of P are of the form $AB \to CD$ or $A \to BC$ or $A \to B$ or $A \to a$ or $A \to \lambda$, where $A, B, C, D \in N$ and $a \in \pi$ (see Theorem 1.6 i)). Without loss of generality we can assume that any derivation in H has the form

$$S \Longrightarrow^* A_1A_2 \ldots A_m \Longrightarrow^* a_1a_2 \ldots a_n\,, \tag{2.4}$$

where we only use rules of the form $AB \to CD$ or $A \to BC$ or $A \to B$ in the first phase and only rules of the form $A \to a$ or $A \to \lambda$ in the second phase.

We construct the phrase structure grammar $H' = (N \cup \{\$\}, T, P', S)$ by the setting

$$\begin{aligned} P' &= \{\alpha \to \beta \mid \alpha \to \beta \in P,\ \beta \neq \lambda\} \cup \{A \to \$ \mid A \to \lambda \in P\} \\ &\quad \cup \{\$A \to A\$ \mid A \in N\} \cup \{\$b \to b\bar{b}b \mid b \in \pi\} \cup \{b\$ \to b\bar{b}b \mid b \in \pi\}\,. \end{aligned}$$

By construction, all rules of P' are monotone, and thus H' is a monotone grammar.

From a derivation (2.4) in H we get a derivation

$$S \Longrightarrow^* A_1A_2 \ldots A_m \Longrightarrow^* \$^{i_1}a_1\$^{i_2}a_2 \ldots \$^{i_n}a_n\$^{i_{n+1}}\,, \tag{2.5}$$

where we used $A \to \$$ instead of $A \to \lambda$. By iterated applications of $\$b \to b\bar{b}b$, we get $\$^nb \Longrightarrow^* (b\bar{b})^nb$. Obviously, $bccp(b) = bccp((b\bar{b})^nb)$. Analogously, by applications of $b\$ \to b\bar{b}b$, we have $b\$^n \Longrightarrow^* b(\bar{b}b)^n$ and $bccp(b) = bccp(b(\bar{b}b)^n)$. Therefore

$$\$^{i_1}a_1\$^{i_2}a_2 \ldots \$^{i_n}a_n\$^{i_{n+1}} \Longrightarrow^* a_1'a_2' \ldots a_r' \tag{2.6}$$

such that

$$bccp(a_1a_2 \ldots a_n) = bccp(a_1'a_2' \ldots a_r').$$

This proves $bccp(H) \subseteq bccp(H')$.

Conversely, without loss of generality we can assume that any derivation in H' is of the form

$$S \Longrightarrow^* A_1A_2 \ldots A_m \Longrightarrow^* \$^{i_1}a_1\$^{i_2}a_2 \ldots \$^{i_n}a_n\$^{i_{n+1}} \Longrightarrow^* a_1'a_2' \ldots a_r'$$

by a combination of derivation of the form (2.5) and (2.6). Now it easy to show that $a_1a_2 \ldots a_n \in L(H)$ and $bccp(a_1a_2 \ldots a_n) = bccp(a_1'a_2' \ldots a_r')$. Thus $bccp(H') \subseteq bccp(H)$ holds, too.

Hence $L = bccp(H) = bccp(H')$ and therefore $\mathcal{CCP}(RE) \subseteq \mathcal{CCP}(CS)$. □

We see that with respect to pictures the situation is simpler because the hierarchy has only three families instead of four in the word case. However, in the word case, most decision problems are undecidable for $\mathcal{L}(RE)$. Thus we can expect that the corresponding problems for pictures are undecidable for $\mathcal{CCP}(RE)$, too. By $\mathcal{CCP}(CS) = \mathcal{CCP}(RE)$, we can expect that the problems are already undecidable for the monotone grammars, although the corresponding word problem is decidable for monotone grammars. Therefore we have a hint that picture grammars will have more worse decidability properties. In the following section we shall give a verification of this expectation.

2.3 Decision Problems for Chain Code Picture Languages

2.3.1 Classical Decision Problems

In this section we consider the four problems known from word languages in their variants for picture languages.

We begin with the membership problem. If one transfers the membership problem from word languages to picture languages one gets the following version.

Membership problem (P1)
Given a grammar $G = (N, \pi, P, S)$ and a chain code picture p,
decide whether or not $p \in bccp(G)$.

Considering (P1) we ask whether or not there is a word $w \in L(G)$ such that $p = bccp(w)$. Obviously, this problem can be reformulated as the question whether or not $L(G) \cap des(p) = \emptyset$. Clearly, one can refine the question and ask on the cardinality of the set $L(G) \cap des(p)$ or equivalently of the set of words in $L(G)$ which describe p. Two special cases are of interest. Are there infinitely many different descriptions of p in $L(G)$, and is there a unique description of p in $L(G)$. Formally we get the following two problems.

Problem (P2)
Given a grammar $G = (N, \pi, P, S)$ and a chain code picture p,
decide whether the set $\{w \mid w \in L(G) \text{ and } bccp(w) = p\}$ is finite.

Problem (P3)
Given a grammar $G = (N, \pi, P, S)$ and a word $w \in \pi^*$,
decide whether the set $\{w \mid w \in L(G) \text{ and } bccp(w) = p\}$ is a singleton.

Theorem 2.14 *i) The problems (P1), (P2), and (P3) are decidable for context-free grammars G.*
ii) The problems (P1), (P2), and (P3) are undecidable for monotone grammars G.

Proof. i) (P1) and (P2) are equivalent to the questions whether or not $L(G) \cap des(p)$ is empty and finite, respectively. Since $des(p)$ is regular by Theorem 2.3, we get that $L(G) \cap des(p)$ is a context-free language by Theorem 1.12. Since the emptiness and the finiteness are decidable for context-free languages (see Theorem 1.24), we get the decidability of (P1) and (P2).

To decide whether or not $L(G) \cap des(p)$ consists of exactly one word we proceed as follows. First we check whether or not $L(G) \cap des(p)$ is finite. If the answer is "no", then $L(G) \cap des(p)$ does not contain exactly one element. In the affirmative case we determine the constant k of the pumping lemma for the language $L(G) \cap des(p)$. The language $L(G) \cap des(p)$ cannot contain a word longer than k (such a word z would imply a decomposition $z = uvwxy$ and the infinite set of words $uv^iwx^iy \in L(G) \cap des(p)$, $i \geq 0$, in contrast to the finiteness of this set). Therefore we check all words which are not longer than k whether or not they are in $L(G) \cap des(p)$. If we find exactly one such word, the answer to (P3) is "yes"; otherwise the answer is "no".

ii) By the closure properties of $\mathcal{L}(CS)$ (see Theorem 1.12) $L(G) \cap des(p) \in \mathcal{L}(CS)$. But we cannot say that $L(G) \cap des(p) = \emptyset$ is undecidable because the emptiness problem for monotone grammars is undecidable, because languages of the form $L(G) \cap des(p)$ form a subset $\mathcal{L}(CS)$, and for this subset the emptiness problem can be decidable.

Let $G = (N, \pi, P, S)$ be an arbitrary monotone grammar. We consider the λ-free homomorphism h given by $h(u) = h(d) = h(r) = h(l) = u$. Then $h(L(G))$ consists of all words u^n such that there is a word w of length n in $L(G)$. Clearly,

$$L(G) = \emptyset \text{ if and only if } h(L(G)) = \emptyset\,. \tag{2.7}$$

Let σ be the λ-free substitution defined by $\sigma(u) = \{u, r\}$. Then

$$K = \sigma(h(L(G)) \cap \{ur^m \mid m \geq 0\} = \{ur^{m-1} \mid m = |w| \text{ for some } w \in L(G)\}.$$

Furthermore,

$$h(L(G)) = \emptyset \text{ if and only if } K = \emptyset\,. \tag{2.8}$$

Let g be the λ-free homomorphism with $g(u) = u$ and $g(r) = du$. Then

$$K' = g(K) = \{u(du)^{m-1} \mid m = |w| \text{ for some } w \in L(G)\}.$$

Then

$$K = \emptyset \text{ if and only if } K' = \emptyset\,. \tag{2.9}$$

Moreover, K' is obtained from $L(G)$ by a sequence of applications of λ-free homomorphisms and substitutions. By Theorem 1.13, $K' = L(G')$ for some monotone grammar.

We consider G' and the picture p consisting of a vertical unit line. Because

$$des(p) = \{u(du)^k \mid k \geq 0\} \cup \{u(du)^k d \mid k \geq 0\} \cup \{d(ud)^k \mid k \geq 0\} \cup \{d(ud)^k u \mid k \geq 0\}\,,$$

we have

$$L(G') \cap des(p) = \emptyset \text{ if and only if } K' \cap des(p) = \emptyset \text{ if and only if } K' = \emptyset\,. \tag{2.10}$$

Thus by a combination of (2.7) – (2.10),

$$L(G') \cap des(p) = \emptyset \text{ if and only if } L(G) = \emptyset\,. \tag{2.11}$$

Let us assume that (P1) is decidable for monotone grammars. Then we can decide the emptiness of $L(G') \cap des(p)$. By (2.11), we can decide the emptiness of $L(G)$ which contradicts Theorem 1.24. Thus our assumption has to fail, i.e., (P1) is undecidable for monotone grammars.

Let G be a monotone grammar and p a picture. Then, for $b \in \pi$, we consider the languages

$$L_b = L(G) \cap des(p) \cap \{bw \mid w \in \pi^*\}$$

of all words of $L(G)$ which begin with the letter b and describe p. Furthermore, we set

$$L'_b = \{(b\bar{b})^m \mid m \geq 0\}L_b = \{(b\bar{b})^m bw \mid m \geq 0,\ bw \in L(G)\}\,.$$

By the closure properties of $\mathcal{L}(CS)$ (see Theorem 1.12), there is a monotone grammar G' with

$$L(G') = L'_u \cup L'_d \cup L'_r \cup L'_l\,.$$

Now assume that $L(G) \cap des(p)$ is not empty. Then there is a non-empty word $v \in L(G) \cap des(p)$ such that $bccp(v) = p$. Clearly, $v = bw$ for some $b \in \pi$. Then all words $(b\bar{b})^m bw$ with $m \geq 0$ are in L'_b and therefore in $L(G')$. Because $bccp(bw) = bccp((b\bar{b})^m bw)$ for any $m \geq 0$, $L(G')$ contains an infinite set of words which describe p. Hence we have show that the non-emptiness of $L(G) \cap des(p)$ implies the infinity of $L(G') \cap des(p)$.

Conversely, if $L(G) \cap des(p)$ is empty, then $L(G') \cap des(p)$ is empty, too, and hence $L(G') \cap des(p)$ is finite.

Hence $L(G) \cap des(p) = \emptyset$ if and only if $L(G') \cap des(p)$ is finite. Assume that (P2) is decidable for monotone grammars. Then the finiteness of $L(G') \cap des(p)$ is decidable, and thus the emptiness of $L(G) \cap des(p)$ is decidable, which contradicts the undecidability of (P1).

Therefore our assumption has to fail, i.e., (P2) is undecidable for monotone grammars.

We omit the proof of the undecidability of (P3) for monotone grammars. □

Comparing the decidability status of the membership problem for word languages (see Theorem 1.24) and for picture languages (see Theorem 2.14) we see that picture languages have a more worse behaviour since the membership problem is undecidable already for monotone grammars.

If we take into consideration the complexity also with respect to context-free and regular grammars the situation in case of picture languages is more worse. By Theorem 1.26 the complexity of the membership problem for word languages is cubic for context-free grammars and linear and regular grammars. In the case of picture languages one has to expect that the membership problem for regular grammars has a non-polynomial complexity. This follows from the following statement (if one expects that $\mathbf{P} \neq \mathbf{NP}$).

Theorem 2.15 *The membership problem (P1) is* $\mathbf{NP}$*-complete for regular grammars.*

Proof. i) First we prove that (P1) is in $\mathbf{NP}$.

Let $G = (N, T, P, S)$ be a regular grammar. Without loss of generality we assume that G is in the normal form given in Theorem 1.6, i.e., all rules have the form $A \to aB$ or $A \to a$ with $A, B \in N$ and $a \in T$. We show that, for any $p \in bccp(G)$, there is a word w with

$$bccp(w) = p \quad \text{and} \quad |w| \leq \#(N) \cdot \#(p) \cdot \#(V(p)) \tag{2.12}$$

(note that p is a finite set of unit lines and hence $\#(p)$ denotes the number of unit lines of p). In order to prove this let $w \in L(G)$ be a word of minimal length which describes p. Let

$$D : S \Longrightarrow a_1A_1 \Longrightarrow a_1a_2A_2 \Longrightarrow \ldots \Longrightarrow a_1a_2\ldots a_{n-1}A_{n-1} \Longrightarrow a_1a_2\ldots a_{n-1}a_n = w$$

be a derivation of w. Let $w_i = a_1a_2\ldots a_i$ and $dccp(w_i) = ((0,0), p_i, e_i)$.

We now prove the following fact.

Fact: For $1 \leq i < j \leq n$, *$dccp(w_i) = dccp(w_j)$ implies $A_i = A_j$.*

Let us assume that this assertion does not hold, i.e., we have $dccp(w_i) = dccp(w_j)$ and $A_i \neq A_j$. Then the derivation

$$D' : S \Longrightarrow^* w_iA_i \Longrightarrow w_ia_{j+1}A_{j+1} \Longrightarrow w_ia_{j+1}a_{j+2}A_{j+2} \Longrightarrow^* w_ia_{j+1}a_{j+2}\ldots a_n$$

gives a word $w' = w_ia_{j+1}a_{j+2}\ldots a_n \in L(G)$. Moreover, since $w = w_ja_{j+1}a_{j+2}\ldots a_n$, $p_i = p_j$ and $e_i = e_j$, we add the unit lines corresponding to the letters $a_{j+1}, a_{j+2}, \ldots, a_n$ at the same positions to draw $dccp(w')$ or $dccp(w)$. Hence $dccp(w') = dccp(w)$. This contradicts our assumption that w has minimal length of all words describing p because $|w'| = |w| - (j - i)$.

By the fact, all elements of the sequence

$$(p_1, e_1, A_1),\ (p_2, e_2, A_2),\ \ldots,\ (p_{n-1}, e_{n-1}, A_{n-1})$$

are different. Hence n is bounded by the product of the number of pictures p_i, the number of endpoint e_i and the number of nonterminals A_i. Because $p_i \subseteq p_j$ for $i < j$ by the construction of a drawn picture, the number of possible p_i is bounded by the size of p. Since any endpoint belongs to $V(p)$, $\#(V(p))$ bounds the number of possible endpoints. Now the estimation of (2.12) follows.

Hence we generate nondeterministically all words of length $\leq \#(N) \cdot \#(p) \cdot \#(V(p))$ and check whether it describes p and whether it belongs to $L(G)$. Both parts can be performed in polynomial time. Thus we can nondeterministically check in polynomial time (with respect to the size of the given picture p and the size of the given grammar G) whether or not $p \in bccp(G)$.

ii) We now give a reduction of 3-SAT to (P1). By Theorems 1.29 and 1.31, we then get the **NP**-completeness of (P1).

Let an instance of 3-SAT be given by the disjunctions C_j, $1 \leq j \leq n$ over the variables $x_1, x_2, \ldots, x_m$. Let

$$C_j = x_{j_1}^{k_1} \vee x_{j_2}^{k_2} \vee x_{j_3}^{k_3}$$

where $x^1 = x$ and $x^0 = \overline{x}$ is the negation of x.

We associate with this problem the regular grammar

$$G = (\{S, S', T, F, X_1, X_2, X_3, U, G, H, Y\}, \pi, P, S)$$

where P consists of all rules of the forms

$$\begin{aligned}
&S \to d^2r^2u^2S',\\
&S' \to T,\ S' \to F,\ S' \to rd^3ru^3,\\
&T \to uU,\ T \to rF,\ T \to d^2ru^2S',\\
&F \to uG,\ F \to rT,\ F \to d^2ru^2S',\\
&U \to uU,\ U \to ru^iluX_i \text{ for } 1 \leq i \leq 3,\\
&X_i \to rX_i,\ X_i \to lX_i,\ X_i \to drd^ilY \quad \text{for} \quad 1 \leq i \leq 3,\\
&Y \to dY,\ Y \to rulrF,\\
&G \to uG,\ G \to ru^iludrd^ilH,\\
&H \to dH,\ H \to rulrT
\end{aligned}$$

and the picture p which is constructed as follows:

1. p consists of m parts, each associated with a variable x_i, each of these parts has the form $bccp(r^{4n}d^2ru^2)$, we label the point reached after drawing $u^{4(i-1)}$ and $u^{4(i-1)+1}$ by x_i^j and $\overline{x_i}^j$, respectively,
2. if $C_j = x_{j_1}^{k_1} \vee x_{j_2}^{k_2} \vee x_{j_3}^{k_3}$, then
   - we start in $x_{j_1}^j$ if $k_1 = 1$ or in $\overline{x_{j_1}}^j$ if $k_1 = 0$ the drawing of $u^{5(j-1)+3}rulu$ and dru,

- we start in $x_{j_2}^j$ if $k_2 = 1$ or in $\overline{x_{j_2}}^j$ if $k_2 = 0$ the drawing of $u^{5(j-1)+2}ru^2lu$ and dru,
- we start in $x_{j_3}^j$ if $k_3 = 1$ or in $\overline{x_{j_3}}^j$ if $k_3 = 0$ the drawing of $u^{5(j-1)+1}ru^3lu$ and dru, and
- we connect the upper ends of these drawings by a line,

3. we add in the beginning and in the end the drawings of $d^2r^2u^2$ and rd^3ru^3, respectively.

In Figure 2.6 we give an example for the construction of the picture from a 3-SAT instance.

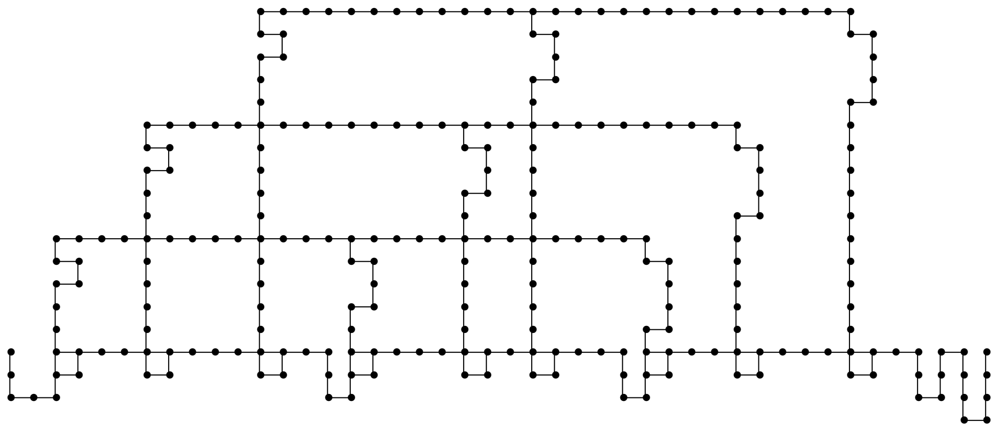

Figure 2.6: The picture associated with the instance of 3-SAT consisting of the disjunctions $x \vee y \vee z$, $x \vee \overline{y} \vee z$ and $\overline{x} \vee y \vee \overline{z}$

Now we prove that $p \in bccp(G)$ if and only if there is an assignment of the variables $x_1, x_2, \ldots, x_m$ such that all disjunctions $C_1, C_2, \ldots, C_n$ get the value true, i.e., the instance of 3-SAT has a solution. In order to get this result we analyse the derivations in G. Using the only rule for the axiom S we get $d^2r^2u^2S'$, i.e., we draw the picture of $d^2r^2u^2$ which we have added in point 3 of the construction of p and we are in a position labelled by x_1^1. Let us assume we are in a position labelled by x_i^1 and the only nonterminal (at he end) of the sentential form is S'. Now we use a rule $S' \to T$ or $S \to F$, which assigns the truth value T for true or F for false to the variable x_i. In the sequel we move to the right and change by each such move the truth value. Therefore in any position labelled by x_i^j or $\overline{x_i}^j$ the assigned truth value is correct (since the distance between points labelled by x_i^j and $x_i^{j'}$ is even, and the same holds for the negated versions). If the value T holds in some position, then we can draw pictures of the form shown in Figure 2.7 using the nonterminals U, X_1, X_2, X_3, Y where the right part of Figure 2.7 holds if the upward and downward move are performed in the same position. Moreover, if the truth value is F, then we only get the pictures shown in Figure 2.8. The difference to the situation where the truth value is T (see Figure 2.7) is that the upper straight line produced by the nonterminal X_1 or X_2 or X_3 is missing. By application of $T \to d^2ru^2S'$ or $F \to d^2ru^2S'$ we start the same procedure for the next variable and finish the derivation by $S' \to rd^3ru^3$.

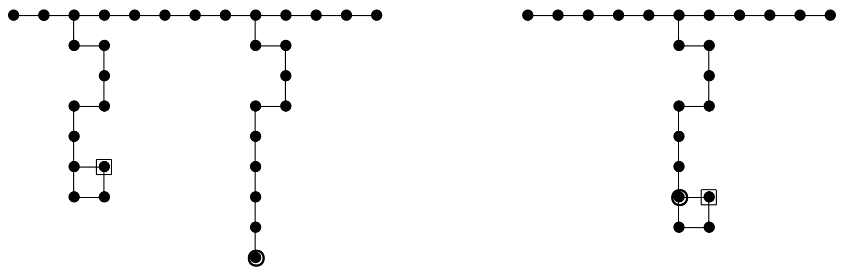

Figure 2.7: Possible pictures associated with the truth value T

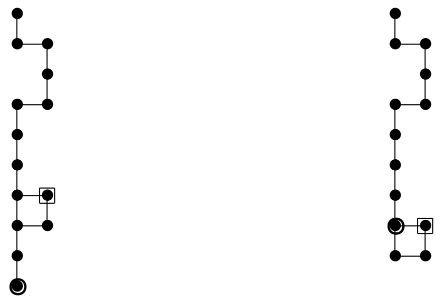

Figure 2.8: Possible picture associated with the truth value F

By this explanation it is obvious that we produce p if and only if we have a upper lines connecting the vertical lines from the positions associated with the variables of the disjunction fore all disjunctions, which holds if and only if each disjunction contains a literal whose truth value is T. Therefore p can be generated if and only if there is an assignment such that all disjunctions get true. □

We now discuss the emptiness problem.

Theorem 2.16 *The emptiness problem*

> *Given a phrase structure grammar $G = (N, \pi, P, S)$,*
> *decide whether or not the picture set bccp(G) is empty?*

is decidable for context-free grammars and undecidable for monotone grammars.

Proof. It is obvious that *bccp*$(G) = \emptyset$ if and only if $L(G) = \emptyset$ because *bccp*(G) consist of the pictures which are associated with the words of $L(G)$. Hence both statements follow from Theorem 1.24. □

With respect to the finiteness we cannot give such an easy reduction from picture languages to word languages as given for the emptiness problem. This can be seen from the grammar G_4 of Example 2.11. The word language generated by G_4 is infinite whereas its picture language is finite.

Definition 2.17 *A context-free grammar $G = (N, \pi, P, S)$ is called normal if, for every nonterminal $A \in N$ and any derivation $A \Longrightarrow^* xAy$ with $x, y \in \pi^*$, $sh(x) = sh(y) = (0, 0)$.*

Lemma 2.18 *For any normal context-free grammar G, there exist a constant c such that, for any $w \in L(G)$, $\sqrt{m^2+n^2} \leq c$ holds where $(m,n) = sh(w)$.*

Proof. Let $G = (N, \pi, P, S)$ be a normal context-free grammar. Without loss of generality we can assume that G is in the normal form given in Theorem 1.6 iii) (it is easy to see that the transformation in the normal form preserves normality). Thus all rules of P have the $A \to BC$ or $A \to a$. Hence the derivation trees are binary trees besides the last application of $A \to a$ to get the leafs. Thus the derivation tree associated with a derivation of w has $|w|$ leaves and therefore a height at least $2^{|w|}$. Therefore the derivations trees of words with $|w| > 2^{\#(N)}$ have height $> \#(N)$. Hence any such tree contains a path from the root S to a leave in which some nonterminal A occurs at least two times. Therefore we have the derivation

$$S \Longrightarrow^* z_1Az_2 \Longrightarrow^* z_1xAyz_2 \Longrightarrow z_1xz_3yz_2 = w \in \pi^* .$$

Obviously, we also have the derivation

$$S \Longrightarrow^* z_1Az_2 \Longrightarrow^* z_1z_3z_2 \in \pi^*.$$

Note that $sh(w) = sh(z_1z_3z_2)+sh(x)+sh(y)$ by (2.3). Since G is normal, $sh(x) = sh(y) = (0,0)$ and, consequently, $sh(w) = sh(z_1z_3z_2)$. Therefore, for any w with $w > 2^{\#(N)}$ we have a shorter word with the same shift. This implies that all shifts can already be obtained by words of $L(G)$ with a length $\leq 2^{\#(N)}$. Since there are only finitely many words in $L(G)$ with a length $\leq 2^{\#(N)}$,

$$c = \max\{\sqrt{m^2+n^2} \mid w \in L(G),\ |w| \leq 2^{\#(N)},\ sh(w) = (m,n)\}$$

can be computed. By our considerations above, for any word $w \in L(G)$, $\sqrt{m^2+n^2} \leq c$ where $sh(w) = (m,n)$. □

Lemma 2.18 says that, for any drawn picture $((0,0), p, (m,n))$ the end point (m,n) is in a circle with radius c and centre $(0,0)$. However, the intermediate points of the drawing can be outside of the circle. The following lemma shows that one can get a circle such that any point of the drawing is in the circle, because any intermediate point belongs to a prefix of a word of $L(G)$.

Lemma 2.19 *Let $G = (N, \pi, P, S)$ be a normal context-free grammar. Then there exist a normal context-free grammar $G' = (N', \pi, P', S')$ such that $L(G') = pref(L(G))$.*

Proof. Let $G = (N, \pi, P, S)$ be a normal context-free grammar. without loss of generality we can assume that G is in the normal form given in Theorem 1.6 iii). Thus all rules of P have the form $A \to BC$ or $A \to a$ with $A, B, C \in N$ and $a \in \pi$. We construct the context-free grammar

$$G' = (N \cup \{A' \mid A \in N\}, \pi, P', S'),$$

where P' consist of all productions obtained in the following way:

– if $A \to BC \in P$, then $A \to BC \in P'$, $A' \to BC' \in P'$ and $A' \to B' \in P'$,
– if $A \to a \in P$, then $A \to a \in P'$, $A' \to a \in P'$ and $A' \to \lambda \in P'$.

We first note that $L(G') = pref(L(G))$. This follows easily from the fact that $S' \Longrightarrow^* wA'$ holds in G' if and only if $S \to wAv$ holds in G for some v. Since any terminating derivation in G' has – without loss of generality – the form

$$S \Longrightarrow^* wA' \Longrightarrow wa \in \pi^* \quad \text{or} \quad S \Longrightarrow^* wA' \Longrightarrow w \in \pi^*,$$

the words in $L(G')$ are prefixes of words in $L(G)$. It is easy to prove that any prefix of $L(G)$ can be generated by G'. Thus $L(G') = pref(L(G)$.

Ne now prove that G' is normal. If we have a derivation $A \Longrightarrow^* xAy$ with $x, y \in \pi^*$in G', then no primed version of a symbol of N can occur in this derivation. Therefore $A \Longrightarrow^* xAy$ is a derivation in G, too. Since G is normal, $sh(x) = sh(y) = (0,0)$. If we have a derivation $A' \Longrightarrow xA'y$ with $x, y \in \pi^*$ in G', then we have $y = \lambda$ and therefore $sh(y) = sh(\lambda) = (0,0)$. Moreover, there is a derivation $A \Longrightarrow^* xAz$ for some $z \in \pi^*$ in G. By the normality of G, $sh(x) = (0,0)$. □

Corollary 2.20 *Let G be a context-free grammar. Then $bccp(G)$ is finite if and only if G is normal.*

Proof. Let G be a normal context-free grammar. Then, by Lemma 2.19, there is a normal context-free grammar G' such that $L(G') = pref(L(G))$. By Lemma 2.18, there is a constant c such that, for any $w \in L(G')$, $\sqrt{m^2 + n^2} \le c$ where $sh(w) = (m, n)$.

Now let $p \in bccp(G)$. Then $p = bccp(w)$ for some $w \in L(G)$. Any point $(m, n) \in V(p)$ is reached by the drawing process according to w with start point $(0,0)$. Therefore $(m, n) \in V(p)$ is the end point of the drawing according of some prefix v of w. Since $v \in pref(L(G)) = L(G')$, we get $\sqrt{m^2 + n^2} \le c$. Thus, for any picture $p \in bccp(G)$, any point of $V(p)$ is in the circle with radius c and centre $(0,0)$. Because there are only finitely many unit lines in the circle, there are only finitely many pictures in the circle. Hence $bccp(G)$ is finite.

Let G be a context-free grammar which is not normal. Then there is a derivation $A \Longrightarrow^* xAy$ with $sh(x) \ne (0,0)$ or $sh(y) \ne (0,0)$. We discuss the former case, the considerations for the latter case are analogous. Let $sh(x) = (k, l)$ with $k + l > 0$. Without loss of generality we assume that $k > 0$.

Let $r \in \mathbb{N}$ be an arbitrary number. For any $n \in \mathbb{N}$, in G we have a derivation

$$S \Longrightarrow^* uAu' \Longrightarrow^* uxAyu' \Longrightarrow^* ux^2Ay^2u' \Longrightarrow^* \ldots \Longrightarrow^* ux^nAy^nu' \Longrightarrow^* ux^nzy^nu' \in \pi^*.$$

Hence $p = bccp(ux^nzy^nu') \in bccp(G)$. The drawing according to the prefix ux^n ends in a point (s,t) of $V(p)$. Because $(s,t) = sh(ux^n) = sh(u) + n(k,l)$, by an appropriate choice of n, we get $|s| > |r|$, i.e., the distance of $(s,t) \in V(p)$ from the origin is larger than any given threshold. Hence $bccp(G)$ is infinite. □

In order to use this corollary for a decision of the finiteness of $bccp(G)$ we need some information whether or not the normality of a context-free grammar is decidable. The answer is given in the following lemma.

Lemma 2.21 *It is decidable whether or not a given context-free grammar G is normal.*

Proof. Let $G = (N, \pi, P, S)$ be a context-free grammar. For any $A \in N$, we define

$$L_A = \{x \mid x \in \pi^*,\ A \Longrightarrow^* xAy \text{ for some } y \in \pi^*\}$$

and

$$L'_A = \{x \mid x \in \pi^*,\ A \Longrightarrow^* yAx \text{ for some } y \in \pi^*\}\,.$$

It is easy to see that the grammar

$$G_A = \{N \cup \{A'\}, \pi \cup \{\$\}, P_A \cup \{A \to \$\} \cup \{A' \to w \mid A \to w \in P\}, A')$$

(where \$ is not in $V \cup \pi \cup \{A'\}$) generates

$$L(G_A) = \{x\$y \mid x, y \in \pi^*,\ A \Longrightarrow xAy\} \cup \{z \mid z \in \pi^*,\ A \Longrightarrow^* z\}\,.$$

By Theorems 1.12 and 1.16,

$$pref(L(G_A)) \cap \pi^*\{\$\} = \{x\$ \mid x \in \pi^*,\ A \Longrightarrow xAy \text{ for some } y \in \pi^*\}$$

is context-free. Let h be the homomorphism given by $h(b) = b$ for $b \in \pi^*$ and $h(\$) = \lambda$. Since $L_A = h(pref(L(G_A)) \cap \pi^*\{\$\}$, L_A is context-free. Thus L_A is semi-linear.

Note that the set M of all words with $sh(w) = (0,0)$ is semi-linear, too (for $sh(w) = (0,0)$, $\#_u(w) = \#_d(w)$ and $\#_r(w) = \#_l(w)$, and therefore $\Psi(w) = \#_u(w)(1,1,0,0) + \#_r(w)(0,0,1,1)$).

Obviously, $L_A \subseteq M$ iff $\Psi(L_A) \subseteq \psi(M)$. By Theorem 1.25 ii), we can decide whether or not $L_A \subseteq M$.

Analogously, we can decide whether $L'_A \subseteq M$.

The given grammar G is normal if and only if L_A and L'_A are contained in M for all A. By the above considerations, this is decidable. □

Theorem 2.22 *The finiteness problem*

> *Given a phrase structure grammar $G = (N, \pi, P, S)$,*
> *decide whether or not the picture set bccp(G) is finite*

is decidable for context-free grammars and undecidable for monotone grammars.

Proof. The statement with respect to context-free grammars follows immediately from the Lemmas 2.20 and 2.21.

Let G be a monotone grammar. We construct a monotone grammar G' such that $L(G') = L(G)\{r^n \mid n \geq 1\}$ (see Theorem 1.12). Obviously, if $bccp(G) = \emptyset$, then $bccp(G') = \emptyset$ and hence $bccp(G')$ is finite. On the other hand, if $bccp(G) \neq \emptyset$, then $bccp(G')$ is infinite since the added tails r^n, $n \geq 1$, produce an infinite set from a picture in $bccp(G)$. Thus $bccp(G) = \emptyset$ if and only if $bccp(G')$ is finite. Hence the decidability of the finiteness of $bccp(G')$ implies the decidability of the emptiness of $bccp(G)$ which is undecidable by Theorem 2.16. Thus the finiteness of the set of pictures generated by a monotone grammar is undecidable. □

Without proof we mention the following theorem (for a proof we refer to [24]; the proof uses a simulation of the work of a Turing machine by chain code pictures; an analogous idea is the basis of the proof of Theorem **??**, see below).

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.