Grammars with Regulated Rewriting

Jürgen Dassow
Otto-von-Guericke-Universität Magdeburg
PSF 4120, D – 39016 Magdeburg
`e-mail: dassow@iws.cs.uni-magdeburg.de`

Abstract: Context-free grammars are not able to cover all linguistic phenomena. Thus we define some types of grammars where context-free rules are used and restriction imposed on the derivations. We illustrate the concepts by example, compare the generative power, give some closure and decidability properties and basic facts on syntactic complexity.

0. Introduction

The regular and context-free grammars/languages are the most investigated types of formal languages which, in addition, have a lot of nice properties (see [9, 11, 14] and the corresponding chapters of this volume). However, these types of grammars/languages are not able to cover all aspects which occur in modelling of phenomena by means of formal languages. Here we only mention an example from natural languages. Let us consider the following sequence of a German dialect spoken in Switzerland:

S1=*Jan säit das mer em Hans hälfed.*
(Jan says that we helped Hans.)
S2=*Jan säit das mer em Hans es Huus hälfed aastriche.*
(Jan said that we helped Hans to paint the house.)
S3=*Jan säit das mer d'chind em Hans es Huus lönd hälfed aastriche.*
(Jan said that we allowed the children to help Hans to paint the house.)

Further, let h be the morphisms which maps *Hans* and *hälfed* to the letter a, *Huus, aastriche, d'chind* and *lönd* to b and all other words of the sentences to the empty word. Then we get

$$h(S1) = aa,\ h(S2) = abab,\ h(S3) = babbab\,.$$

It is easy to see that sentences of the above structure form a sublanguage L of that German dialect with

$$h(L) = \{ww \mid w \in \{a,b\}^*\}\,.$$

It is well-known that $\{ww \mid w \in \{a,b\}^+\}$ is neither a regular nor a context-free language.

Analogous phenomena can be given using programming languages instead of natural languages.

Obviously, one can construct a context-sensitive or length-increasing grammar which generates $\{ww \mid w \in \{a,b\}^+\}$, and the same statement holds for other languages obtained by modelling aspects of natural or programming languages. However, the corresponding classes of grammars and languages have bad features, e.g. for context-sensitive grammars, the emptiness problem is undecidable and only exponential algorithms are known for the membership problem. Moreover, such concepts as derivation tree, which is an important tool for the analysis of context-free and natural languages, cannot be transformed to context-sensitive and length-increasing grammars.

Therefore one is interested in classes of languages which on the one hand only use context-free production rules and a sequential derivation process and on the other hand have a larger generative capacity by some additional mechanisms. In this chapter we present some of such classes where the mechanisms select some derivations as successful and take the corresponding terminal words into the language whereas the words obtained by other derivations are not taken into the language.

We finish this section with some notations. For a word w we denote by $Perm(w)$ the set of words which are obtained from w by a permutation of the letters. For a language L we define $Perm(L)$ as union of all sets $Perm(w)$ taken over all words $w \in L$. Let $U \subseteq V$, then the morphism $h_U : V^* \to U^*$ is defined by $h_U(a) = a$ for $a \in U$ and $h_U(b) = \lambda$ for $b \notin U$. By $\Longrightarrow^*$ we denote the reflexive and transitive closure of a relation $\Longrightarrow$.

Moreover, in Section 5 we shall assume that the reader is familiar with the basic concepts of computational complexity. We refer to [11].

1. Control by Prescribed Sequences of Productions

We start with a type of grammars where we require that the sequence of productions applied in a derivation belong to a given regular language associated with the grammar. Formally we get the following definition.

Definition 1 *i) A regularly controlled grammar is a quintuple $G = (N, T, S, P, R)$ where*
— N, T, P and S are specified as in a context-free grammar,
— R is a regular set over P.

ii) The language $L(G)$ generated by G consists of all words $w \in T^*$ such that there is a derivation

$$S \Longrightarrow_{p_1} w_1 \Longrightarrow_{p_2} w_2 \Longrightarrow_{p_3} \ldots \Longrightarrow_{p_n} w_n = w$$

with

$$p_1 p_2 p_3 \ldots p_n \in R.$$

Example 1 Let

$$G_1 = (\{S, A, B\}, \{a, b\}, S, \{p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8\}, R)$$

be a regulary controlled grammar where

$$p_0 = S \to AB, p_1 = A \to aA,\ p_2 = B \to aB,\ p_3 = A \to bA,\ p_4 = B \to bB,$$

$p_5 = A \to a,\ p_6 = B \to a,\ p_7 = A \to b,\ p_8 = B \to b,$
$R = p_0\{p_1p_2, p_3p_4\}^*\{p_5p_6, p_7p_8\}.$

Any derivation has to start with p_0 which yields AB. By the application of p_1p_2 or p_3p_4 we generate from A and B the same letter a or b, respectively. Furthermore, this can be iterated. By p_5p_6 and p_7p_8 we produce the same letter from A and B and stop the derivation. Thus the generated language is

$$L(G_1) = \{ww \mid w \in \{a, b\}^+\}.$$

Thus we can generate the non-context-free language which is of interest by the introduction of this chapter.

Assume that we want to apply the sequence $q_1q_2 \dots q_r \in R$ of productions and we have already applied $q_1q_2 \dots q_k$, $k < r$. If the resulting sentential form does not contain the left hand side of the production q_{k+1}, then by the above definition the derivation is blocked, i.e., we cannot use any string of R. In order to overcome this situation we give the following definition.

Definition 2 *We say that $x \in V^+$ directly derives $y \in V^*$ in appearance checking mode by application of $p = A \to w \in P$ (written as $x \Longrightarrow_p^{ac} y$) if one of the following conditions hold:*

$$x = x_1Ax_2 \text{ and } y = x_1wx_2 \quad \text{or} \quad A \text{ does not appear in } x,\ p \in F \text{ and } x = y.$$

Definition 3 *i) A regularly controlled (context-free) grammar with appearance checking is a 6-tuple $G = (N, T, S, P, R, F)$ where*
— N, T, P, S and R are specified as in a regularly controlled grammar and
— F is a subset of P.

ii) The language $L(G)$ generated by G with appearance checking consists of all words $w \in T^*$ such that there is a derivation

$$S \Longrightarrow_{p_1}^{ac} w_1 \Longrightarrow_{p_2}^{ac} w_2 \Longrightarrow_{p_3}^{ac} \dots \Longrightarrow_{p_n}^{ac} w_n = w$$

with

$$p_1p_2p_3 \dots p_n \in R.$$

Example 2 We consider the regularly controlled grammar

$$G_2 = (\{S, A, X\}, \{a\}, S, \{p_1, p_2, p_3, p_4, p_5\}, R, F)$$

with appearance checking where

$p_1 = S \to AA,\ p_2 = S \to X,\ p_3 = A \to S,\ p_4 = A \to X,\ p_5 = S \to a,$
$R = (p_1^*p_2p_3^*p_4)^*p_5^* \quad \text{and} \quad F = \{p_2, p_4\}.$

We regard the sentential form $w = S^{2^n}$ for some $n \geq 0$ and assume that we have applied a sequence of $(p_1^* p_2 p_3^* p_4)^*$. This situation holds for the axiom. We have to continue with p_5 or p_1. In the former case we have to finish by further applications of p_5. Hence we get a^{2^n}. In the latter case after some applications of p_1 we have to apply p_2 which introduces X which cannot be terminated. Thus p_2 has to be applied until all letters S are replaced. This gives $S^n \Longrightarrow^* (A^2)^n = A^{2^{n+1}}$. Now we apply p_2 in the appearance checking mode without changing the sentential form $A^{2^{n+1}}$. By the same argumentation we now have to replace all occurrences of A by p_3 and then to apply p_4 in the appearance checking mode. This yields $A^{2^{n+1}} \Longrightarrow S^{2^{n+1}}$ which is of the form as the sentential form we started with. Therefore G_2 generates the non-semilinear language

$$L(G_2) = \{a^{2^m} \mid m \geq 1\}\,.$$

We denote by λrC, λrC_{ac}, rC and rC_{ac} the families of all languages generated by regularly controlled grammars (without appearance checking), regularly controlled grammars with appearance checking, regularly controlled grammars without erasing rules (and without appearance checking) and regularly controlled grammars with appearance checking and without erasing rules, respectively.

The following theorem summarizes the relations to the language families of the Chomsky hierarchy.

Theorem 1 *i) All languages of λrC over a unary alphabet are regular.*
ii) $CF \subset rC \subset rC_{ac} \subset CS$
iii) $CF \subset rC \subseteq \lambda rC \subset \lambda rC_{ac} = RE$

Proof. Since the known proofs for statement i) use deep results from the theory of Petri nets (see [10]) we omit a proof.

$CF \subset rC$. Obviously, the context-free grammar $G = (N, T, S, P)$ (which can be assumed to have no erasing rules) and the regularly controlled grammar $G' = (N, T, S, P, P^*)$ generate the same language since the control set P^* imposes no restriction. This proves $CF \subseteq rC$. The strictness of the inclusion follows by Example 1.

$rC \subset rC_{ac}$ and $\lambda rC \subset \lambda rC_{ac}$. The inclusions hold by definition. The strictnesses follow by i) and Example 2.

$rC \subset CS$. Let $G = (N, T, S, P, R)$ be a regularly controlled grammar, and let $\mathcal{A} = (P, Z, z_0, Q, \delta)$ be a deterministic finite automaton (with input set P, set Z of states, initial state z_0, set Q of final states and transition function δ) which accepts R. Then we construct the length-increasing grammar $G' = (N \cup \{S', \$\} \cup Z, T \cup \{\S\}, S', P')$ with P consisting of all rules of the following form:
– $S' \to \$z_0S$
(initial rule which introduces a marker in the beginning and a state),
– $zx \to xz$ and $xz \to zx$ for $z \in Z$ and $x \in N \cup T$
(by these rules the state can be moved to any place in the sentential form),

– $zA \to z'w$ for $p = A \to w \in P$ and $\delta(z,p) = z'$
(this rule simulates an application of p and changes the state according to δ),
– $\$z \to §^2$ for $z \in Q$
(if a final state is reached, we can finish the derivation; all rules require a state).

By the explanation added to the rules we can only finish a derivation with a terminal word, if – besides the last rule – the sequence of productions belongs to the language R accepted by $\mathcal{A}$ and the last rule is $\$z \to §^2$. Therefore we obtain $L(G') = §^2 \cdot L(G)$. Because CS is closed under derivatives we get that $L(G)$ belongs to CS.

For the proof of the other inclusions and strictnesses we refer to [2, 3]. $\square$

The words of the (regular) control set describe the complete sequences of productions which are allowed. We now define a new type of grammars where we require only partial sequences of the derivations.

Definition 4 *i) A* *matrix grammar with appearance checking* *is specified as a quintuple* $G = (N, T, S, M, F)$ *where*

- N, T *and* S *are specified as in a context-free grammar,*
- $M = \{m_1, m_2, \dots m_n\}$, $n \geq 1$, *is a finite set of sequences* $m_i = (p_{i_1}, \dots, p_{i_{k(i)}})$, $k(i) \geq 1$, $1 \leq i \leq n$, *where any* p_{i_j}, $1 \leq i \leq n$, $1 \leq j \leq k(i)$, *is a context-free production*
- F *is a subset of all productions occurring in the elements of* M, *i.e.* $F \subseteq \{p_{i_j} : 1 \leq i \leq n, 1 \leq j \leq k(i)\}$.

ii) We say that M *is a* *matrix grammar without appearance checking* *if and only if* $F = \emptyset$.
iii) For m_i, $1 \leq i \leq n$, *and* $x, y \in V_G^*$, *we define* $x \Longrightarrow_{m_i} y$ *by*

$$x = x_0 \Longrightarrow_{p_{i_1}}^{ac} x_1 \Longrightarrow_{p_{i_2}}^{ac} x_2 \Longrightarrow_{p_{i_3}}^{ac} \dots \Longrightarrow_{p_{i_{k(i)}}}^{ac} x_{k(i)} = y$$

iv) The *language* $L(G)$ *generated by* G *(with appearance checking)* *is defined as the set of all words* $w \in T^*$ *such that there is a derivation*

$$S \Longrightarrow_{m_{j_1}} y_1 \Longrightarrow_{m_{j_2}} y_2 \Longrightarrow_{m_{j_3}} \dots \Longrightarrow_{m_{j_k}} w$$

for some $k \geq 1$, $1 \leq j_i \leq n$, $1 \leq i \leq k$.

The elements of M are called matrices.

Intuitively, the application of a matrix consist of an application of the productions of the matrix in the order given by the matrix.

Example 3 We consider the matrix grammar

$$G_3 = (\{S, A, B\}, \{a, b\}, S, \{m_0, m_1, m_2, m_3, m_4\}, \emptyset)$$

(without appearance checking) where

$$\begin{aligned}&m_0 = (S \to AB),\ m_1 = (A \to aA,\ B \to aB),\ m_2 = (A \to bA,\ B \to bB),\\ &m_3 = (A \to a,\ B \to a),\ m_4 = (A \to b,\ B \to b)\,.\end{aligned}$$

Assume we have a sentential form of the form $w = zAzB$ for a certain word $z \in \{a,b\}^*$ (any derivation has to start with an application of m_0 which yields AB of the desired form). If we apply m_1 or m_2, then we obtain $zxAzxB$ with $x = a$ or $x = b$, respectively, which are sentential form of the form we started with. If we apply m_3 or m_4, then we obtain $zxzx$ with $x = a$ or $x = b$, respectively. Therefore

$$L(G_3) = \{ww \mid w \in \{a,b\}^+\ \}\,.$$

Example 4 Let

$$G_4 = (\{S, A, A', B, C, D\}, \{a,b\}, S, \{m_0, m_0', m_1, m_2, m_3, m_4, m_5, m_5'\}, F)$$

be a matrix grammar with appearance checking where

$$\begin{aligned}&m_0 = (S \to AB),\ m_0' = (S \to AD), m_1 = (A \to A'A',\ B \to B),\\ &m_2 = (A \to E,\ B \to C),\ m_3 = (A \to a, D \to D),\ m_4 = (A \to E, D \to b),\\ &m_5 = (A' \to A,\ C \to C),\ m_6 = (A' \to T,\ C \to B),\ m_6' = (A' \to E, D \to D),\\ &F = \{A \to E, A' \to E\}\,.\end{aligned}$$

Obviously, the matrices m_2 and m_4 can only be applied to a sentential form w, if w does not contain the letter A, since we generate the trap symbol E which cannot be replaced otherwise. The same holds for m_6 and m_6' with respect to A'. Thus any derivation is essentially of the following form:

$$\begin{aligned}S \ &\Longrightarrow_{m_0} \ AB \Longrightarrow_{m_1} A'A'B \Longrightarrow_{m_2} A'A'C \Longrightarrow_{m_5} AA'C \Longrightarrow_{m_5} AAC \Longrightarrow_{m_6} AAB\\ &\Longrightarrow_{m_1} \ A'A'AB \Longrightarrow_{m_1} A'A'A'A'B \Longrightarrow_{m_2} A'A'A'A'C \Longrightarrow_{m_5}^* AAAAC \Longrightarrow_{m_6} AAAAB\\ &\Longrightarrow^* \ A^{2^{n-1}}B \Longrightarrow_{m_1}^* (A')^{2^n}B \Longrightarrow_{m_2} (A')^{2^n}C \Longrightarrow_{m_4}^* A^{2^n}C \Longrightarrow_{m_5'} A^{2^n}D\\ &\Longrightarrow_{m_3}^* \ a^{2^n}D \Longrightarrow_{m_4} a^{2^n}b\,.\end{aligned}$$

This implies $L(G_4) = \{a^{2^m}b : m \geq 0\}$.

By λM, λM_{ac}, M and M_{ac} we denote the families of languages generated by matrix grammars (without appearance checking), matrix grammars with appearance checking, matrix grammars without erasing rules (and without appearance checking) and matrix grammars with appearance checking and without erasing rules, respectively.

Theorem 2 *$M = rC$, $\lambda M = \lambda rC$, $M_{ac} = rC_{ac}$ and $\lambda M_{ac} = \lambda rC_{ac}$.*

Proof. We only prove the first statement; the proofs for the other statements can be given by modifications.

$M \subseteq rC$. Let $G = (N, T, S, M)$ be a matrix grammar as in Definition 4. Then the regularly controlled grammar $G' = (N, T, S, \{p_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq k(i)\}, M^*)$ generates $L(G)$.

$rC \subseteq M$. Let $H = (N, T, S, P, R)$ be a regularly controlled grammar and let $\mathcal{A} = (P, Z, z_0, Q, \delta)$ be a deterministic finite automaton which accepts R (by the set Q of final states). Then we construct the matrix grammar

$$H' = (N \cup \{(z, x) \mid z \in Z, x \in N \cup T\}, T, (z_0, S), M)$$

where M consists of all matrices of the form
– $(A \to w, (z, x) \to (\delta(z, p), x))$ for $p = A \to w \in P$, $x \in N \cup T$, $z \in Z$,
– $((z, A) \to (z, a_1)a_2a_3 \ldots a_n, (z, a_1) \to (\delta(z, p), a_1))$ for $p = A \to a_1a_2 \ldots a_n \in P$, $z \in Z$,
– $((z, x) \to x)$ for $z \in Q$, $x \in T$.
It is easy to prove by induction that

$$S \Longrightarrow_{p_1} w_1 \Longrightarrow_{p_2} w_2 \ldots \Longrightarrow_{p_k} x_1x_2 \ldots x_s$$

with $x_i \in N \cup T$ for $1 \leq i \leq s$ holds in H if and only if

$$(z_0, S) \Longrightarrow^* (\delta(z_0, p_1p_2 \ldots p_k), x_1)x_2x_3 \ldots x_s$$

holds in H'. Thus

$$S \Longrightarrow_{p_1p_2 \ldots p_k} x_1x_2 \ldots x_s \in L(H)$$

iff

$$(z_0, S) \Longrightarrow^* (\delta(z_0, p_1p_2 \ldots p_k), x_1)x_2 \ldots x_s \Longrightarrow x_1x_2 \ldots x_s \in L(H') \,.$$

Now $L(H) = L(H')$ follows. □

In a matrix grammar, the rules of a matrix have to be used in the order given by the matrix. We can modify the definition by allowing that all rules of a matrix have to be used but they can be applied in an arbitrary order. If we require that a matrix can only be started, if all rules of the matrix used before have already been applied, we obtain *unordered matrix grammars* which have the same generative power as matrix grammars. In the following type of grammars we can start a new matrix before finishing those which have been started earlier.

Definition 5 *i) An unordered vector grammar is a quadruple $G = (V, T, S, M)$ where N, T, M and S are defined as for matrix grammars.*

ii) The language $L(G)$ generated by G is defined as the set of all words $w \in T$ such that there is a derivation

$$S \Longrightarrow_{p-1} w_1 \Longrightarrow_{p_2} w_2 \Longrightarrow_{p_3} \ldots \Longrightarrow_{p_n} w$$

where $p_1p_2 \ldots p_n$ is a permutation of some element of M^*.

Example 5 We consider the unordered vector grammar

$$G_5 = (\{S, A, B\}, \{a, b\}, \{m_0, m_1, m_2, m_3, m_4\}, S, \emptyset)$$

with

$$m_0 = (S \to AB),\ m_1 = (A \to aA,\ B \to aB),\ m_2 = (A \to bA,\ B \to bB),$$
$$m_3 = (A \to a,\ B \to a),\ m_4 = (A \to b,\ B \to b)\,.$$

Obviously, any sentential form has the form s or $zAz'B$ or zAz' or $zz'B$ or zz' where z and z' are only generated by using rules for A and B, respectively. Since in a terminating derivation all rules of a matrix have to be used and the rules of a matrix introduce in z and z' the same letter, the number of occurrences of a in z and z' have to coincide, and the same holds for b. Furthermore, in order to terminate we use exactly one of the matrices m_3 and m_4. Hence

$$L(G_5) = \{wxw'x : x \in \{a, b\}, w \in \{a, b\}^*, w' \in Perm(w)\}\,.$$

We note that the control set $Perm(M^*)$ is not regular in general. Assume that M consists of a single matrix (p, q). Then $Perm(M^*) \cap p^+q^+$ is the non-regular set $\{p^nq^n \mid n \geq 1\}$. By the closure of regular sets under intersection we get that $Perm(M^*)$ is not regular.

By λuV and uV we denote the families of languages generated by unordered vector grammars and unordered vector grammars without erasing rules, respectively.

Without proof we mention some inclusion results for unordered vector languages.

Theorem 3 *$CF \subset uV = \lambda uV \subset M$.* □

Theorem 4 *Each language in uV is semilinear.*

Proof. For a context-free grammar $G = (N, T, S, P)$, define the context-free grammar $G' = (N, P, S, P')$ where P' consists of all rules $p' = A \to h_N(w)p$ with $p = A \to w \in P$. Obviously, if $v \in L(G')$ then $Perm(v)$ contains a sequence of productions generating a terminal word. Conversely, if v is a sequence of productions generating a terminal word, then $Perm(v)$ contains an element of $L(G')$. Therefore $L(G')$ is semilinear.

Now let $H = (N, T, S, M)$ be an unordered vector grammar. Further, let $G = (N, T, S, P)$ be the context-free grammar where P consists of all productions which occur in some matrix of M. Moreover, let G' be the context-free grammar $G' = (N, P, S, P')$ associated with G as above. Then $L(G') \cap Perm(M^*)$ is the set C of terminating derivations in H. Since the intersection of semilinear sets is semilinear, C is semilinear.

We define the linear transformation τ as follows: For $x = (x_1, x_2, \ldots x_n) \in \Psi_P(C)$, we set $\tau(x) = \sum_{i=1}^{n} x_i \Psi_T(w_i)$ where $p_i = A_i \to w_i$ is the ith rule of P. It is easy to see that $\tau(\Psi_P(C)) = \Psi_T(L(H))$. Since linear transformations preserve the semilinearity, $L(H)$ is semilinear. □

2. Control by Computed Sequences of Productions

In the preceding section the allowed derivations were given in the grammar. We now give some grammars where the derivation is accompanied by a computation which selects the allowed derivations.

Definition 6 *i) A* <u>*programmed grammar*</u> *is a quadruple* $G = (N, T, S, P)$ *where* N, T *and* S *are specified as in a context-free grammar and* P *is a finite set of triples* $r = (p, \sigma, \varphi)$ *where* p *is a context-free productions and* σ *and* φ *are subsets of* P.

ii) If $r = (p, \sigma, \emptyset)$ *holds for any* $r \in P$, *then we say that* G *is a programmed grammar without appearance checking. Otherwise* G *is a programmed grammar with appearance checking.*

iii) The <u>*language* $L(G)$ *generated by* G</u> *is defined as the set of all words* $w \in T^*$ *such that there is a derivation*

$$S = w_0 \Longrightarrow_{r_1} w_1 \Longrightarrow_{r_2} w_2 \Longrightarrow_{r_3} \ldots \Longrightarrow_{r_k} w_k = w,$$

$k \geq 1$ *and, for* $r_i = (A_i \to v_i, \sigma_i, \varphi_i)$, *one of the following conditions hold:*

$$w_{i-1} = w'_{i-1} A_i w''_{i-1},\ w_i = w'_{i-1} v_i w''_{i-1} \text{ for some } w'_{i-1}, w''_{i-1} \in V_G \text{ and } r_{i+1} \in \sigma_i$$

or

$$A_i \text{ does not occur in } w_{i-1},\ w_{i-1} = w_i \text{ and } r_{i+1} \in \varphi_i.$$

If $r = (p, \sigma, \varphi)$, then σ and φ are called the <u>success field</u> and <u>failure field</u> of r, respectively.

In a programmed grammar after applying a production p we "compute" (choose) the next production which has to be taken from its success field; if the left hand side of p does not occur in the sentential form, we apply p in the appearance checking mode and continue with a rule from its failure field.

Example 6 We consider the programmed grammar

$$G_6 = (\{S, A, B\}, \{a, b\}, S, \{r_0, r_1, r_2, \ldots, r_8\})$$

with

$$\begin{aligned}
&r_0 = (S \to AB, \{r_1, r_3, r_5, r_7\}, \emptyset),\ r_1 = (A \to aA, \{r_2\}, \emptyset),\\
&r_2 = (B \to aB, \{r_1, r_3, r_5, r_7\}, \emptyset),\ r_3 = (A \to bA, \{r_4\}, \emptyset),\\
&r_4 = (B \to bB, \{r_1, r_3, r_5, r_7\}, \emptyset),\ r_5 = (A \to a, \{r_6\}, \emptyset),\\
&r_6 = (B \to a, \emptyset, \emptyset),\ r_7 = (A \to b, \{r_8\}, \emptyset),\ r_8 = (B \to b, \emptyset, \emptyset).
\end{aligned}$$

G_6 is a grammar without appearance checking since all failure fields are empty.

Assume that we have a sentential form $zAzB$ with $z \in \{a,b\}^*$ and we have to apply a rule from the set $Q = \{r_1, r_3, r_5, r_7\}$ (any derivation starts with an application of r_0 which yields the sentential form AB and we have to continue with a rule of Q). If we apply r_1 or r_3, we have to continue with r_2 or r_4, get $zaAzaB$ or $zbAzbB$, respectively, and the next production has to be taken from Q, again. If we apply r_5 or r_7, the next production has to be r_6 or r_7, yielding $zaza$ or $zbzb$, respectively, and the derivation stops. Therefore

$$L(G_6) = \{ww \mid w \in \{a,b\}^+ \}\,.$$

Example 7 Let

$$G_7 = (\{S,A\}, \{a\}, \{r_1, r_2, r_3\}, S)$$

be a programmed grammar with

$$r_1 = (S \to AA, \{r_1\}, \{r_2\}),\ \ r_2 = (A \to S, \{r_2\}, \{r_1, r_3\}),\ \ r_3 = (S \to a, \{r_3\}, \emptyset)\,.$$

By definition r_1 and r_3 have to applied as long as an S occurs in the sentential and then we have to continue with r_2 or to stop, respectively. r_2 has to be applied as long as A occurs and then we have to continue with r_1. Thus any derivation is of the form

$$S \Longrightarrow_{r_1} AA \Longrightarrow^*_{r_2} SS \Longrightarrow^*_{r_1} A^4 \Longrightarrow^*_{r_2} S^4 \Longrightarrow^* S^{2^n} \Longrightarrow^*_{r_3} a^{2^n}\,.$$

This implies

$$L(G_7) = \{a^{2^m} : m \geq 0\}\,.$$

By λPr and λPr_{ac} we denote the families of languages generated by programmed grammars (without appearance checking) and programmed grammars with appearance checking, respectively. We omit the λ if we restrict to families of languages generated by grammars without erasing rules.

Theorem 5 $Pr = M,\ \ \lambda Pr = \lambda M,\ \ Pr_{ac} = M_{ac},\ \ \lambda Pr_{ac} = \lambda M_{ac}$

Proof. We only prove the first statement; the proofs for the other statements can be given by modifications.

$M \subseteq Pr$. Let $G = (N, T, S, M)$ be a matrix grammar as in Definition 4. Then the programmed grammar

$$G' = (N \cup \{S'\}, T, S', \{r\} \cup \{r_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq k(i)\},)$$

with

$$\begin{aligned} r &= (S' \to S, \{r_{t,1} \mid 1 \leq t \leq n\}, \emptyset)\,, \\ r_{i,j} &= (p_{i,j}, \{r_{i,j+1}\}, \emptyset) \quad \text{for} \quad 1 \leq i \leq n,\ j < k(i)\,, \\ r_{i,k(i)} &= (p_{i,k(i)}, \{r_{t,1} \mid 1 \leq t \leq n\}, \emptyset) \quad \text{for} \quad 1 \leq i \leq n. \end{aligned}$$

It is easy to see that – besides the first rule – we have to use the rules in the programmed grammar in the same order as in the matrix grammar. Hence $L(G) = L(G')$.

$Pr \subseteq M$. Let $H = (N, T, S, P)$ be a programmed grammar. Then we construct the matrix grammar $H' = (N \cup \{S'\} \cup \{(r, x) \mid r \in P, x \in N \cup T\}, T, S', M)$ where M consists of all matrices of the form
– $(S' \to (r, S)$ with $r \in P$
– $(A \to w, (r, x) \to (r', x))$ for $r = (A \to w, \sigma(r), \emptyset) \in P$, $r' \in sigma(r)$ and $x \in X$,
– $((r, A) \to (r, a_1)a_2a_3 \dots a_n, (r, a_1) \to (r', a_1))$ for $r = (A \to a_1a_2 \dots a_n, \sigma(r), \emptyset) \in P$ and $r' \in \sigma(r)$,
– $((r, x) \to x)$ for $r \in P$ and $x \in T$.
It is easy to prove by induction that

$$S \Longrightarrow_{r_1} x_1w_1 \Longrightarrow_{r_2} x_2w_2 \Longrightarrow \dots \Longrightarrow_{r_k} x_kw_k$$

with $x_i \in N \cup T$, $w_i \in (N \cup T)^*$ for $1 \leq i \leq k$ holds in H if and only if

$$S' \Longrightarrow (r_1, S) \Longrightarrow^* (r_2, x)w_1 \Longrightarrow^* \dots \Longrightarrow^* (r_k, x_k)w_k \Longrightarrow x_kw_k$$

holds in H'. Thus $L(H) = L(H')$ follows. □

We now define a type of grammars where with each sentential form an element of a monoid is associated, which is computed during the derivation. Then we accept only such derivations where the element associated with the terminal word is the neutral element of the monoid.

Definition 7 *i) A valence grammar over a monoid is a quintuple $G = (N, T, S, P, M)$ where*
– N, T and S are specified as in a context-free grammar,
– $(M, \circ)$ is a monoid with neutral element e,
– P is a finite set of pairs $r = (p, m)$ where p is a context-free production and $m \in M$.

ii) For $x, y \in V_G^*$, $k, l \in M$, we say that (x, k) directly derives (y, l), written as $x \Longrightarrow y$, iff there is a pair $r = (A \to w, m) \in P$ such that
– $x = x'Ax''$ and $y = x'wx''$ for some $x', x'' \in V_G^*$

and
– $l = k \circ m$.

iii) The language $L(G)$ generated by G is defined as

$$L(G) = \{w \mid w \in T^*,\ (S, e) \Longrightarrow^* (w, e)\}.$$

A valence grammar is called additive or multiplicative if M is the monoid $(\mathbf{Z}, +)$ of integers or $(\mathbf{Q}, \cdot)$ of rational numbers, respectively.

Example 8 We consider the multiplicative grammar

$$G_8 = (\{S, A, B\}, \{a, b\}, S, \{r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}, (\mathbf{Q}, \cdot))$$

where

$$\begin{aligned}
&r_0 = (S \to AB, 1),\\
&r_1 = (A \to aA, 2),\ r_2 = (B \to aB, 1/2),\ r_3 = (A \to bA, 3),\ r_4 = (B \to bB, 1/3),\\
&r_5 = (A \to a, 2),\ r_6 = (B \to a, 1/2),\ r_7 = (A \to b, 3),\ r_8 = (B \to b, 1/3)
\end{aligned}$$

Obviously, any sentential form generated by G_8 has the form

$$(S,1) \text{ or } (z_1Az_2B, 2^{\#_a(z_1)}3^{\#_b(z_1)}(\frac{1}{2})^{\#_a(z_2)}(\frac{1}{3})^{\#_b(z_2)}) \text{ or } (z_1z_2, 2^{\#_a(z_1)}3^{\#_b(z_1)}(\frac{1}{2})^{\#_a(z_2)}(\frac{1}{3})^{\#_b(z_2)})\,.$$

Since the words of $L(G)$ have to be associated with 1 we number of occurrences of a (and b) have to be the same in z_1 and z_2. Thus

$$L(G_8) = \{w_1w_2 \mid w_1 \in \{a,b\}^+,\ w_2 \in Perm(w_1)\}\,.$$

Example 9 Let

$$G_9 = (\{S,A,B\}, \{a,b\}, S, \{r_0', r_1', r_2', r_3', r_4', r_5', r_6', r_7', r_8'\}, (\mathbf{Z}, +))$$

be an additive valence grammar with

$$\begin{aligned}
&r_0' = (S \to AB, 0),\\
&r_1' = (A \to aA, 2),\ r_2' = (B \to aB, -2),\ r_3' = (A \to bA, 3),\ r_4' = (B \to bB, -3),\\
&r_5' = (A \to a, 2),\ r_6' = (B \to a, -2),\ r_7' = (A \to b, 3),\ r_8' = (B \to b, -3)\,.
\end{aligned}$$

Analogously to Example 8 we can see that

$$L(G_9) = \{w_1w_2 \mid w_1, w_2 \in \{a,b\}^+, 2\#_a(w_1) + 3\#_b(w_1) = 2\#_a(w_2) + 3\#_b(w_2)\}\,.$$

It is easy to prove by standard methods that $L(G_9)$ is not context-free.

By λaV, aV, λmV, mV we denote the families of languages generated by additive valence grammars, additive valence grammars without erasing rules, multiplicative valence grammars and multiplicative valence grammars without erasing rules, respectively.

The following theorem presents the relations between families of valence languages and unordered vector languages.

Theorem 6 *$CF \subset aV = \lambda aV \subset mV = \lambda mV = uV$.*

Proof. A context-free grammar can be interpreted as a additive valence grammar where each production is associated with 0. This implies the first inclusion, and its strictness follows by Example 9.

We omit the proofs of the other relations and refer to [2, 7]. □

2. Control by Context Conditions

In this section we consider some grammars where the applicability of a rule depends on the current sentential form. With any rule we associate some restrictions for words (sentential forms) which have to be satisfied in order to apply the rule. The first restriction is the belonging to a regular language associated with the rule.

Definition 8 *i) A <u>conditional grammar</u> is a quadruple $G = (N, T, S, P)$ where*
– N, T and S are specified as in a context-free grammar, and
– P is a finite set of pairs $r = (p, R)$ where p is a context-free production and R is a regular set over V_G.

ii) For $x, y \in V_G^*$, we say that x directly derives y, written as $x \Longrightarrow y$, iff there is a pair $r = (A \to w, R) \in P$ such that $x = x'Ax''$ and $y = x'wx''$ for some $x', x'' \in V_G^*$ and $x \in R$.

iii) The <u>language $L(G)$ generated by G</u> is defined as $L(G) = \{w : w \in T^*,\ S \Longrightarrow^* w\}$.

Example 10 We consider the conditional grammar

$$G_{10} = (\{S, A, B, A', B'\}, \{a, b\}, \{r_0, r_1, \dots r_8\}, S)$$

with

$$\begin{aligned}
&r_0 = (S \to AB, S),\ r_1 = (A \to aA', V^*BV^*),\ r_2 = (A \to bA', V^*BV^*),\\
&r_3 = (B \to aB, V^*aA'V^*),\ r_4 = (B \to bB', V^*bA'V^*),\ r_5 = (A' \to A, V^*B'V^*),\\
&r_6 = (A' \to \lambda, V^*B'V^*),\ r_7 = (B' \to B, V^*AV^*),\ r_8 = (B' \to \lambda, T^*B'T^*)\,.
\end{aligned}$$

We consider $zAzB$ with $z \in \{a, b\}^*$ (any derivation starts with an application of r_0 which gives AB of this form). The only applicable rules are r_1 and r_2 since the rules for B require the presence of A' in the sentential form.

In the former case we obtain $zaA'zB$ which only allows the application of r_3 since the rules for A' require an occurrence of B' and r_4 requires a b before A'. Thus we get $zaA'zaB'$. Now we can continue with r_5 or r_6 which gives $zaAzaB'$ or $zazaB'$ and has to be followed by r_7 and r_8, respectively. Hence we obtain $zaAzaB$, which means that we can iterate the process, or the terminal word $zaza$.

Analogously, if we apply r_2, we get $zbAzbB$ or $zbzb$.

Thus $L(G_{10}) = \{ww \mid w \in \{a, b\}^+\ \}$.

The following types of grammar will be obtained by restrictions to special regular sets.

Definition 9 *i) A <u>semi-conditional grammar</u> is a quadruple $G = (N, T, S, P)$ where*
— N, T and S are specified as in a context-free grammar, and
— P is a finite set of triples $r = (p, R, Q)$ where p is a context-free production and R and Q are disjoint finite sets of words over V_G.

ii) For $x, y \in V_G^*$, we say that x directly derives y, written as $x \Longrightarrow y$, iff there is a triple $r = (A \to w, R, Q) \in P$ such that
— $x = x'Ax''$ and $y = x'wx''$ for some $x', x'' \in V_G^*$,
— any word of R is a subword of x, and no word of Q is a subword of x.

iii) The language $L(G)$ generated by G is defined as $L(G) = \{w : w \in T^*,\ S \Longrightarrow^* w\}$.

R and Q are called the permitted context and forbidden context associated with r (or p), respectively.

Example 11 We consider the semi-conditional grammar

$$G_{11} = (\{S, S', S'', A\}, \{a\}, \{r_1, r_2, r_3, r_4, r_5\}, S)$$

with

$$r_1 = (S \to S'S', \emptyset, \{S'', A\}),\ r_2 = (S' \to S'', \emptyset, \{S\}),\ r_3 = (S'' \to S, \emptyset, \{S'\}),$$
$$r_4 = (S \to A, \emptyset, \{S'\}),\ r_5 = (A \to a, \emptyset, \{S\})\,.$$

We consider S^{2^n} (the axiom S has this form). The only applicable rules are r_1 and r_4.

In the latter case we get S^rAS^s with $r + s = 2^n - 1$. Now the only applicable rule is r_4, again, since A is in the forbidden context of r_1 and S is in the forbidden context of r_5. Thus we have to replace all occurrences of S by A, which gives A^{2^n}. Now, we can only apply r_5 to all occurrences of a and get a^{2^n}. In the former case, by analogous arguments we get a derivation

$$S^{2^n} \Longrightarrow_{r_1}^* (S'S')^{2^n} = (S')^{2^{n+1}} \Longrightarrow_{r_2}^* (S'')^{2^{n+1}} \Longrightarrow_{r_3}^* S^{2^{n+1}}$$

such that we can iterate the process. Hence $L(G_{11}) = \{a^{2^n} : n \geq 0\}$.

Any semi-conditional grammar can be interpreted as a conditional grammar. Instead of using $(A \to w, R, Q)$ we have to take $(A \to w, R')$ where

$$R' = \bigcap_{w \in R} (N \cup T)^*\{w\}(N \cup T)^* \cap ((N \cup T)^* \setminus (N \cup T)^*Q(N \cup T)^*)\,.$$

Obviously, in both cases the rule $A \to w$ can be applied to the same words.

We now make a further restriction to words of length 1 in the permitting and forbidden contexts.

Definition 10 *A random context grammar is a semi-conditional grammar where the permitting and forbidden contexts of all productions are subsets of the set of nonterminals.*

A permitting (forbidden, respectively) random context grammar is a random context grammar where all forbidden (permitting, respectively) contexts are empty.

G_{11} of Example 11 is a forbidding random context grammar.

By λC, λsC, λRC, λpRC and λfRC we denote the families of languages generated by conditional grammars, semi-conditional grammars, random context grammars, permitting random context grammars and forbidden random context grammars, respectively. We omit the λ if we restrict to families of languages generated by grammars without erasing rules.

Theorem 7 *$\lambda C = \lambda sC = RE$ and $C = sC = CS$*

Proof. $C \subseteq CS$. Let $G = (N, T, S, P)$ be a conditional grammar. We construct the length increasing grammar $G' = (N', T \cup \{\$, \S\}, S', P')$. For $p = (A \to w, R_p) \in P$, let $\mathcal{A}_p = (N \cup T, Z_p, z_{0,p}, Q_p, \delta_p)$ be the finite deterministic automaton accepting R_p. We assume $Z_p \cap Z_q = \emptyset$ for $p \neq q$. We set

$$N' = N \cup \{S', S''\} \cup \bigcup_{p \in P} Z_p \cup \{S_p\},,$$

and define P as the set of all rules of the form
(1) $S' \to \$z_{0,p}S\S$ for $p \in P$
(2) $zx \to xz'$ for $z \in Z_p$, $x \in N \cup T$, $z' = \delta_p(z, x)$ and $z\S \to S_p\S$ for $z \in Q_p$,
(3) $xS_p \to S_px$ for $p \in P$, $x \in N \cup T$ and $S_pA \to S''w$ for $p = (A \to w, R)$,
(4) $xS'' \to S''x$ for $p \in P$, $x \in N \cup T$,
(5) $\$S'' \to \$z_{0,p}$ for $p \in P$ and (8) $\$S'' \to \$\$$.
(1) is the initial step. By the rules of the form (2) we check whether the sentential form belongs to R_p and switch to S_p in the affirmative case. By rules of the form (3), we move S_p to the left and replace some occurrence of the left hand side of p by its right hand side and introduce S''. Thus we have simulated the application of p in $G.S''$ is moved to the left marker by rules of the form (4). By a rule of the form (5) we restart the simulation process or we stop the derivation process. Thus we get $L(G') = \$\$L(G)\S$. Since length-increasing grammars generate context-sensitive languages and CS is closed under derivatives, we obtain $L(G) \in CS$.

$CS \subseteq C$. Let $L \in CS$. Then $L = L(H)$ for some length-increasing $H = (N, T, S, P)$ in Kuroda normal form, i.e. all rules of P are of the form $A \to BC$ or $AB \to CD$ or $A \to a$ with $A, B, C, D \in N$ and $a \in T$. We construct the conditional grammar $H' = (N', T, S, P')$ where P' contains all rules of P of the forms $A \to BC$ and $A \to a$ and all rules

$$(A \to A_p, (N \cup T)^*),\ (B \to B_p, (N \cup T)^* A_p B (N \cup T)^*),$$
$$(A_p \to C, (N \cup T)^* A_p B_p (N \cup T)^*),\ (B_p \to D, (N \cup T)^* C B_p (N \cup T)^*)$$

for any rule $p = AB \to CD$ (which have to be applied in this order and give $wABw' \Longrightarrow^* wCDw'$, hence simulating the application of p) and N' contains the letters of N and all symbols A_p and B_p for $p \in P$. It is easy to see that $L(H) = L(H')$.

For a proof of the remaining statements we refer to [2, 3] $\square$

Theorem 8 *i)* $CF \subset pRC \subset RC = M_{ac} \subset \lambda RC = \lambda M_{ac}$.
ii) $pRC \subseteq \lambda pRC \subset \lambda RC = \lambda M_{ac}$.
iii) $RC \subseteq M$
iv) $\lambda pRC \subseteq \lambda M$

Proof. We only prove $RC \subset M_{ac}$; for proofs of the other statements we refer to [2, 3].

Let $G = (N, T, S, P)$ be a random context grammar. Then we construct the matrix grammar $G' = (N \cup \{E\}, T, S, M, F)$ where M and F are defined as follows: For any rule

$$p = (A \to w, \{A_1, A_2, \ldots, A_r\}, \{B_1, B_2, \ldots, B_s\})$$

we associate the matrix

$$m_p = (A_1 \to A_1, A_2 \to A_2, \ldots, A_r \to A_r, B_1 \to E, B_2 \to E, \ldots, B_s \to E, A \to w).$$

Then M consists of all matrices m_p for $p \in P$ and F consists of all rules with right hand side E. If m_p is applied to w in a terminating derivation, then the letters $A_1, A_2, \ldots, A_r$ have to occur in w and $B_1, B_2, \ldots, B_s$ cannot occur in w since the application of a rule $B_i \to E$ introduces E which cannot be replaced. Thus $w \Longrightarrow_p w'$ in G if and only if $w \Longrightarrow_{m_p} w'$ in G'. Hence $L(G) = L(G')$. □

Relations for the forbidden random context grammars will be given in Section 4. **3.**

Further Regulations

We now introduce a type of grammars where we impose a partial order on the set of productions, and we can only apply a production if there is no greater applicable production.

Definition 11 *i) An ordered grammar is a quadruple* $G = (N, T, S, P)$ *where*
– N, T *and* S *are specified as in a context-free grammar and*
– P *is a finite partially ordered set of context-free production.*

ii) For $x, y \in V_G$*, we say that* x directly derives y*, written as* $x \Longrightarrow y$*, if and only if there is a production* $p = A \to w \in P$ *such that* $x = x'Ax''$, $y = x'wx''$ *and there is no production* $q = B \to v \in P$ *such that* $p \prec q$ *and* B *occurs in* x.

iii) The language $L(G)$ *generated by* G *is defined as* $L(G) = \{w : w \in T^*, S \Longrightarrow^* w\}$.

Example 12 We consider the ordered grammar $G_{12} = (\{S, S', S'', A, Z\}, \{a\}, P, S)$ where the partially ordered set of productions is given by the following graph

$$\begin{tikzcd}
S'' \to Z & A \to Z & S' \to Z & S \to Z & \\
S \to S'S' \arrow[u] \arrow[ur] & S'' \to S \arrow[u] \arrow[ur] & S \to A \arrow[u] & S' \to S'' \arrow[u] & A \to a \arrow[ul]
\end{tikzcd}$$

(p is smaller than q if there is a directed edge from p to q). By definition, the rule $S \to S'S'$ is only applicable to a sentential form which does not contain the nonterminals S'' and A, $S'' \to S$ can only be applied to sentential form without an occurrence of A and S', $S \to A$ is only applicable if S' does not occur, and $S' \to S''$ and $A \to a$ can only be applied if S is not present. These are exactly the requirements mentioned in the semi-conditional grammar of Example 11. Thus G_{12} and G_{11} allow the same derivations and hence they generate the same language. Hence $L(G_{12}) = \{a^{2^n} : n \geq 0\}$.

By λO and O we denote the families of languages generated by ordered grammars and ordered grammars without erasing rules, respectively.

Theorem 9 *$CF \subset O = fRC \subseteq \lambda O = \lambda fRC \subset \lambda RC$ and $O \subset RC$*

Proof. Obviously, a context-free grammar can be considered as an ordered grammar where all rules are incomparable with each other. Thus the first inclusion holds. Its strictness follows from Example 12.

$fRC \subseteq O$ follows by the construction presented in Example 12. For any forbidden random context grammar $G = (N, T, S, P)$ we construct the ordered grammar $G' = (N \cup \{Z\}, T, S, P')$ where P' contains all rules $A \to Z$ with $A \in N$ and all rules $B \to w$ with $(B \to w, \emptyset, Q) \in P$. Moreover, $A \to Z$ is greater than $B \to w$, if A is contained in the forbidden context Q of $B \to w$.

$O \subseteq fRC$. We take the same rules and add to $A \to w$ as forbidden context all left hand sides of productions greater than $A \to w$.

By definition, $fRC \subseteq \lambda fRC \subseteq \lambda RC$ and $fRC \subseteq RC$. The strictnesses are shown in [4] and [5]. □

Now we consider a type of grammars where with any nonterminal in a sentential form we associate (partially) its derivation.

Definition 12 *i) An <u>indexed grammar</u> is a quintuple $G = (N, T, S, I, P)$ where – N, T and S are specified as in a context-free grammar,*
– I is a finite set of finite sets of productions of the form $A \to w$ with $A \in N$ and $w \in V_G^*$, and
– P is a finite set of productions of the form $A \to \alpha$ with $A \in N$ and $\alpha \in (NI^* \cup T)^*$.

ii) For $x, y \in (NI^* \cup T)^*$, we say that x directly derives y, written as $x \Longrightarrow y$, if either
– $x = x_1 A\beta x_2$ for some $x_1, x_2 \in (NI^* \cup T)^*$, $A \in N$, $\beta \in I^*$,
$A \to X_1\beta_1 X_2\beta_2 \dots X_k\beta_k \in P$, $y = x_1 X_1\gamma_1 X_2\gamma_2 \dots X_k\gamma_k x_2$
with $\gamma_i = \beta_i\beta$ for $X_i \in N$ and $\gamma_i = \lambda$ for $X_i \in T$, $1 \leq i \leq k$,
or
– $x = x_1 Ai\beta x_2$ for some $x_1, x_2 \in (NI^* \cup T)^*$, $A \in N$, $i \in I$, $\beta \in I^*$,
$A \to X_1 X_2 \dots X_k \in i$, $y = x_1 X_1\gamma_1 X_2\gamma_2 \dots X_k\gamma_k x_2$
with $\gamma_i = \beta$ for $X_i \in N$ and $\gamma_i = \lambda$ for $X_i \in T$, $1 \leq i \leq k$.

iii) The <u>language $L(G)$ generated by G</u> is defined as $L(G) = \{w : w \in T^*,\ S \Longrightarrow^* w\}$

Example 13 We consider the index grammar $G_{13} = (\{S, A, B\}, \{a, b\}, S, \{f_a, f_b, h\}, P)$ with

$$f_a = \{B \to Ba\},\ f_b = \{B \to Bb\},\ h = \{B \to \lambda\},$$
$$P = \{S \to Ah, A \to aAf_a, A \to bAf_b, A \to B\}\,.$$

Any derivation has the form

$$\begin{aligned} S &\Longrightarrow Ah \Longrightarrow x_1Af_{x_1}h \Longrightarrow x_1x_2Af_{x_2}f_{x_1}h \Longrightarrow^* x_1x_2\dots x_kAf_{x_k}f_{x_{k-1}}\dots f_{x_1}h \\ &\Longrightarrow x_1x_2\dots x_kBf_{x_k}f_{x_{k-1}}\dots f_{x_1}h \Longrightarrow x_1x_2\dots x_kBf_{x_{k-1}}f_{x_{k-2}}\dots f_{x_1}hx_k \\ &\Longrightarrow x_1x_2\dots x_kBf_{x_{k-2}}f_{x_{k-3}}\dots f_{x_1}hx_{k-1}x_k \Longrightarrow^* x_1x_2\dots x_kBhx_1x_2\dots x_k \\ &\Longrightarrow x_1x_2\dots x_kx_1x_2\dots x_k \end{aligned}$$

which shows that
$$L(G_{13}) = \{ww \mid w \in \{a, b\}^*\}\,.$$

By λI and I we denote the families of languages generated by indexed grammars and indexed grammars without erasing rules, respectively.

Without proof we present the following theorem, for a proof we refer to [1].

Theorem 10 *$CF \subset I = \lambda I \subseteq CS$.*

Proof. The first inclusion holds since a context-free grammar can be considered as an index grammar with an empty set I and a production set $P \subset N \times (N \cup T)^*$. The strictness follows from Example 13.

For the other relations we refer to [1]. □

The following theorems summarize the relation between the language families introduced in this and the preceding sections.

Theorem 11 *i) The following equalities are valid:*
$RE = \lambda M_{ac} = \lambda rC_{ac} = \lambda Pr_{ac} = \lambda RC = \lambda C = \lambda sC$ and $CS = C = sC$,
$\lambda M = \lambda rC = \lambda Pr$ and $M_{ac} = rC_{ac} = Pr_{ac} = RC$, and $M = rC = P$,
$mV = \lambda mV = uV = \lambda uV$ and $aV = \lambda aV$,
$\lambda O = \lambda fRC$ and $O = fRC$,
$I = \lambda I$.
ii) The inclusions presented in Figure 1 hold. □

5. Closure and Decidability Properties

In the preceding section we have defined some grammars with a control mechanism for the application of productions, and we have compared their generative power. In this section we add the closure properties of the language families introduced above and discuss their decidability properties.

$$\begin{tikzcd}
 & & RE & & \\
 & & CS \arrow[u] & & \\
\lambda O \arrow[uurr] & I \arrow[ur] & M_{ac} \arrow[u] & \lambda M \arrow[uul] & \\
 & & M \arrow[u] \arrow[ur, dash] \arrow[dr, dash] & \lambda pRC) \arrow[u, dash] & mV \arrow[ul] \\
O \arrow[uu] \arrow[uurr] & & & pRC \arrow[u, dash] & aV \arrow[u] \\
 & & CF \arrow[ull] \arrow[uuul] \arrow[ur] \arrow[urr] & &
\end{tikzcd}$$

Figure 1: If two families are connected by a line (an arrow), then the upper family includes (includes properly) the lower family; if two families are not connected then they are not necessarily incomparable.

Theorem 12 *The table of Figure 2 gives the closure properties of the families I, λO, O, λpRC, pRC, M_{ac}, λM, M, mV and aV.* □

We now present some undecidability results.

Theorem 13 *Let X be a family of grammars generating one of the following families*

$$\{M_{ac}, M, pRC, O, \lambda M, \lambda pRC, \lambda O, I, mV, aV\}$$

of languages. Then the equivalence problem (decide whether or not two given grammars of X generate the same language) and the context-freeness problem (decide whether or not a grammar of X generates a context-free language) are undecidable.

Proof. The statement for the equivalence problem follows from the known fact that the equivalence of context-free grammars is undecidable. The proof for the context-freeness problem follows along the lines of the proof of Theorem 1.3.6 in [2]. □

Theorem 14 *The table of Figure 3 presents the decidability status of the membership, finiteness and emptiness problem for grammar families generating I, λO, O, M_{ac}, λM, M, pRC, mV and aV.*

Proof. We only prove the decidability and NP-hardness of the emptiness problem for matrix grammars without appearance checking and unordered matrix grammars, respectively. For the other proofs, we refer to [3] and its references.

operation	M_{ac}	λM	M	mV	aV	I	λO	O	λpRC	pRC
union	+	+	+	+	+	+	+	+	+	+
intersection	?	-	-	-	-	-	-	-	-	-
complementation	?	-	-	-	-	-	-	-	-	-
intersection by reg. sets	+	+	+	+	+	+	+	+	+	+
concatenation	+	+	+	+	-	+	+	+	+	+
Kleene-closure	+	?	?	-	-	+	+	+	+	+
λ-free morphisms	+	+	+	+	+	+	+	+	+	+
(arbitrary) morphisms	-	+	-	+	+	+	+	?	+	?
inverse morphisms	+	+	+	+	+	+	+	+	+	+
λ-free gsm-mappings	+	+	+	+	+	+	+	+	+	+
gsm-mappings	-	+	-	+	+	+	+	?	+	?
quotient by regular sets	-	+	-	+	+	+	+	+	+	?
quotient by letters	+	+	+	+	+	+	+	+	+	+

Figure 2: If the family X is closed under the operation $\circ$, then we write a symbol + in the intersection of the corresponding row and column. A symbol - is given, if X is not closed under $\circ$. A question mark denotes an open problem.

An n-dimensional vector addition system is a couple (x_0, K) where $x_0 \in \mathbf{N}^n$ and K is a finite subset of $\mathbf{Z}^n$.

A vector $y \in \mathbf{N}^n$ is called reachable within (x_0, K) if and only if there are vectors $v_1, v_2, \ldots, v_t \in K$, $t \geq 1$, such that

$$x_0 + \sum_{i=1}^{j} v_i \in \mathbf{N}^n \text{ for } 1 \leq j \leq t \quad \text{and} \quad x_0 + \sum_{i=1}^{t} v_i = y.$$

It is known (see [12]) that the reachability problem (given an n-dimensional vector addition system (x_0, K) and a vector $y \in \mathbf{N}^n$, decide whether or not y is reachable within (x_0, K)) is decidable (in exponential space).

Let $G = (N, T, S, M)$ be a matrix grammar without appearance checking. Without loss of generality we can assume that G is of the form presented in the proof of $rC \subseteq M$ (see Theorem 2). With G we associate a vector addition (x_0, K) as follows. We set $x_0 = \Psi_N(S)$. Further, let K be the set of all vectors $v_m = \Psi_N(h_N(w_1 w_2)) - \Psi_N(A_1 A_2)$ with $m = (A_1 \rightarrow w_1, A_2 \rightarrow w_2) \in M$. If we obtain w' from w by application of m, then $\Psi_N(h_N(w')) = \Psi_N(h_N(w)) + v_m$. On the other hand, if $\Psi_N(h_N(w)) + v_m \in \mathbf{N}^n$, then we can apply m to w. Thus $L(G)$ contains a word if and only if $(0, 0, \ldots, 0)$ is reachable in (x_0, K).

The 3-partition problem (given a multiset $\{t_1, t_2, \ldots, t_{3m}\}$ of integers and an integer t, decide whether or not there is partition $\{Q_1, Q_2, \ldots, Q_m\}$ of $\{t_1, t_2, \ldots, t_{3m}\}$ such that $\#(Q_i) = 3$ and $\sum_{s \in Q_i} s = t$ for $1 \leq i \leq m$) is NP-complete.

grammar family	membership problem	emptiness problem	finiteness problem
I	NP-complete	$+$	$+$
λO	?	?	?
O	$+$, NP-hard	?	?
M_{ac}	$+$, NP-hard	-	-
λM	$+$	$+$, NP-hard	$+$, NP-hard
M	$+$	$+$, NP-hard	$+$, NP-hard
RC	$+$	$+$, NP-hard	$+$, NP-hard
uV	$\in$LOGCFL	$+$, NP-hard	$+$, NP-hard
aV	DTIME(n^4)	$+$	$+$

Figure 3: The symbol + deotes that the problem is decidable for the grammar family; the symbol - denotes undecidability; the symbol ? denotes an open problem; in some (decidable) cases a remark on the complexity of the problem is added.

With such a problem we associate the unordered vector grammar

$$G = (\{S, A_1, A_2, \ldots, A_{3m}\}, \{a_1, a_2, \ldots, a_{3m}\}, S, P)$$

with

$$P = \{S \to A_1 A_2 \ldots A_{3m}\} \cup \{(A_i \to a_i, A_j \to a_j, A_k \to a_k) \mid (i,j,k) \in U\}$$

where

$$U = \{(i,j,k) \mid t_i + t_j + t_k = t\} .$$

Obviously, $S \Longrightarrow^* a_1 a_2 \ldots a_{3m}$ if and only if a partition $Q_1, Q_2, \ldots, Q_m$ exists. $\square$

6. Two Measures of Complexities

For a language, we are interested to have a concise description. This implies the search for "small" grammars, where "small" can be understood as a small length of the word representing the grammar. The number of nonterminals or the number of productions are related measures of syntactic complexity. We here restrict to the number of nonterminals; for other measures we refer to [2].

Definition 13 *i) For a grammar G, $Var(G)$ denotes the cardinality of its set of nonterminals.*

ii) Let X be a family of languages generated by grammars of type Y. For a language $L \in X$, we set

$$Var_X(L) = \min\{Var(G) : G \text{ is of type } Y,\ L(G) = L\}.$$

Obviously, $Var(G)$ can immediately be seen from the grammar. However, we note that Gruska has shown that there is no algorithm to determine $Var_{CF}(L)$ for a context-free language L.

The following theorem shows that the description of context-free languages by grammars with regulation can be much more efficient than those by context-free grammars.

Theorem 15 *There is a sequence of regular languages L_n, $n \geq 1$, such that*

$$Var_{CF}(L_n) = n+1,\ Var_{rC}(L_n) = 1,\ Var_M(L_n) \leq 3,\ Var_{Pr}(L_n) = 1,\ Var_{pRC}(L_n) \leq 8.$$

Proof. We consider the language

$$L_n = \bigcup_{i_1}^{n} \{a^i b\}^* \{b\}^n \,.$$

L_n can be generated by the regularly controlled grammar

$$(\{S\}, \{a,b\}, S, \{r\} \cup \{r_i \mid 1 \leq i \leq n-1\}, \bigcup_{i=1} \{r_i\}^* \{r\})$$

with

$$r = S \to b^n \quad \text{and} \quad r_i = A \to a^i bA \text{ for } 1 \leq i \leq n \,.$$

Thus $Var_{rC}(L_n) = 1$ is shown.

It is left to the reader to give a matrix grammar, a programmed grammar and a random context grammar (without erasing rules) and three, one and eight nonterminals, respectively.

The context-free grammar

$$(\{S\} \cup \bigcup_{i=1}^{n} \{A_i\}, \{a,b\}, s, \bigcup_{i=1}^{n} \{S \to A_i, A_i \to a^i bA_i, A_i \to b^n\})$$

generates L_n, and it can be shown that this grammar is minimal with respect to the number of nonterminals (see [2], Example 4.1.3). □

We now present a theorem saying that any recursively enumerable language has a succint description by matrix and programmed grammars whereas this does not hold for random context grammars. For a proof we refer to [8], [6] and [2].

Theorem 16 *i) For any recursively enumerable language L, we have $Var_{\lambda M_{ac}}(L) \leq 3$ and $Var_{\lambda Pr_{ac}}(L) \leq 3$.*

ii) $Var_{\lambda M_{ac}}(\{a^n b^n c^m d^m e^p f^p \mid n,m,p \geq 1\}) = 3$

iii) There is a sequence of recursively enumerable languages L_n, $n \geq 1$, such that $f(n) \leq Var_{\lambda RC_{ac}}(L_n) \leq [\log_2 n] + 3$ for $n \geq 1$ where f is an unbounded function from **N** *into* **N**. □

We now introduce a further measure of complexity. However, it cannot immediately be seen from the grammar; one has to calculated it from the derivations in the grammar.

Definition 14 *i) Let G be a grammar, and let $D = S = w_0 \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \ldots \Longrightarrow w_n = w$ be a derivation of w in G. Then we set*

$$\begin{aligned} Ind(G,w,D) &= \max\{\#_N(w_i) \mid 0 \leq 1 \leq n\}\,, \\ Ind(G,w) &= \min\{Ind(G,w,D) \mid D \textit{ is a derivation of } w \textit{ in } G\}\,, \\ Ind(G) &= \sup\{Ind(G,w) \mid w \in L(G)\}\,. \end{aligned}$$

ii) Let X be a family of languages generated by grammars of type Y. For a language $L \in X$, we set

$$\begin{aligned} Ind(L,X) &= \min\{Ind(G) \mid G \textit{ is of type } Y,\ L = L(G)\}\,, \\ X_{fin} &= \{L \mid L \in X, Ind(L,X) < \infty\}\,. \end{aligned}$$

If we impose the finite index restriction the hierarchy of the language families (see Theorem 11 is essentially changed; most of the families coincide as can be seen from the following theorem (for the proof we refer to the constructions given in the preceding sections and to [2]).

Theorem 17 *i) All the following language families are equal to M_{fin}: Pr_{fin}, $(Pr_{ac})_{fin}$, λPr_{fin}, $(\lambda Pr_{ac})_{fin}$, rC_{fin}, $(rC_{ac})_{fin}$, λrC_{fin}, $(\lambda rC_{ac})_{fin}$, λM_{fin}, $(M_{ac})_{fin}$, $(\lambda M_{ac})_{fin}$, RC_{fin}, λRC_{fin},*
ii) $O_{fin} \subseteq M_{fin} \subseteq C_{fin}$
iii) $pRC_{fin} \subseteq M_{fin} \subset M$
iv) $aV_{fin} \subset uV_{fin} \subseteq M_{fin}$ □

Theorem 18 *Each language in M_{fin} is semilinear.* □

For a proof Theorem 18 we refer to [2].
By Theorem 18, the Parikh images of finite index matrix, programmed, regular control etc. grammars coincide with that of regular languages.

References

[1] A. Aho, Indexed grammars. An extension of context-free grammars. *J. Assoc. Comp. Mach.* **15** (1968) 647–671.

[2] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, Heidelberg, 1989.

[3] J. Dassow, Gh. Păun, A. Salomaa, Grammars with controlled derivations. In [13], Volume 2, Chapter 3, 101–154.

[4] H. Fernau, A predicate for separating language classes. *Bulletin EATCS* **58** (1995) 96–97.

[5] H. Fernau, On grammars and language families. *Fundamenta Informaticae* **25** (1996) 17–34.

[6] H. Fernau, Nonterminal complexity of programmed grammars. In: M. Margenstern and Y. Rogozhin (eds.), Machines, Computations, and Universality, Proc. 3rd MCU, LNCS 2076 (2001) 202–213.

[7] H. Fernau, R. Stiebe, Sequential grammars and automata with valences. *Theor. Comp. Sci.* **276** (2001) 377–405.

[8] R. Freund, Gh. Păun, On the number of non-terminal symbols in graph-controlled, programmed and matrix grammars.In: M. Margenstern and Y. Rogozhin (eds.), Machines, Computations, and Universality, Proc. 3rd MCU, LNCS 2076 (2001) 214–225.

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

[10] D. Hauschildt, M. Jantzen, Petri net algorithms in the theory of matrix grammars. *Acta Informatica* **31** (1994) 719–728.

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

[12] E.W. Mayr, An algorithm for the general Petri net reachability problem. In: *Proc. 13th Symp. Theory of Computation*, 1981, 238–246.

[13] G. Rozenberg, A. Salomaa, *Handbook of Formal Languages I – III*, Springer-Verlag, Berlin, Heidelberg, 1997.

[14] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.