CONTEXT-FREE GRAMMARS WITH GRAPH CONTROLLED TABLES

by

Grzegorz Rozenberg

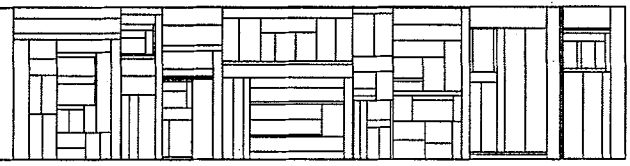
&

Arto Salomaa

DAIMI PB-43

January 1975

Institute of Mathematics University of Aarhus
DEPARTMENT OF COMPUTER SCIENCE
Ny Munkegade - 8000 Aarhus C - Denmark
Phone 06-12 83 55



1. Introduction

One of the quite vivid areas of research in formal language theory is a search for proper kinds of control which, when applied to context-free grammars, yield the family of context-sensitive languages or some large enough subfamily. (see, e.g., [2], [4], [7], [8], [9].) Context-free grammars have the advantage of being very useful as a language generating device but have the disadvantage that their generative capacity is not too strong.

Perhaps the most obvious control one can introduce on a set of context-free productions is a graph specifying their succession in rewriting. Modifications of this idea are matrix grammars, programmed grammars, time-varying grammars, and grammars with a regular control language, cf. [9]. It has been shown in [8] and [4] that proper direction control added to matrix grammars or to programmed grammars (without erasing) yields indeed all context-sensitive languages. These results are to be contrasted with the fact that the largest family of context-free programmed languages without erasing (P_{ac} in the notation of [9]), as well as the corresponding family with the left most interpretation on the application of productions, are properly contained in the family of context-sensitive languages.

In this paper, we re-examine the idea of graph control from the very beginning. We prove that if one imposes the control over sets of productions rather than over single productions, then one gets indeed (under the leftmost interpretation but without erasing) all context-sensitive languages. The result holds true for both variations of graph control corresponding to programmed grammars

of simplest kinds: those with empty failure fields and those with unconditional transfer. The idea of using sets of productions rather than single productions corresponds to the notion of a "table" borrowed from the theory of L systems, cf. [3] and [5].

A brief outline of the contents of this paper follows. In Section 2, we investigate traditional programmed grammars and present some results concerning unconditional transfer and leftmost application of the productions. Graph control of two different types for sets of productions is introduced in Section 3. In Section 4, it is shown that both of these types of control yield all context-sensitive languages. Section 5 contains some corollaries and concluding remarks.

2. Unconditional transfer and leftmost interpretation

We expect the reader to be familiar with formal language theory in the extent of [9]. As regards programmed grammars, we use the notation of [9].

(1) $$P\ ,\ P^{\lambda}\ ,\ P_{ac}\ ,\ P^{\lambda}_{ac}$$

for the four language families obtained by allowing or excluding λ-productions and/or appearance checking. In [9], several equivalent characterizations for the families (1) (e.g., in terms of matrix grammars) are given. It is well-known that

(2) $$P^{\lambda}_{ac} = L_0 \text{ and } L_2 \subset P \subseteq P_{ac} \subset L_1\ ,$$

where L_0, L_1 and L_2 denote the families of recursively enumerable, context-sensitive and context-free languages, respectively, whereas it is an open problem whether or not the inclusion $P \subseteq P_{ac}$ is proper.

Also the position of the family P^{λ}, apart from inclusions obvious by definitions, remains open.

We now consider some modifications of the families (1). The families obtained by the leftmost interpretation (contrasted to the free interpretation in the families (1)) on the application of productions are denoted by adding (left) after the name of the family. Thus, P_{ac} (left) is the family generated by λ-free programmed grammars such that each production $A \to x$ has to be applied to the leftmost occurrence of the nonterminal A. It has been shown in [2] that

(3) $$P_{ac} \subseteq P_{ac}\text{ (left)},$$

whereas it is an open problem whether or not the inclusion is proper. The family P_{ac} (left) is still properly contained in the family of context-sensitive languages.

We denote by P_{ut} (resp. P_{ut} (left)) the family of languages generated by λ-free unconditional transfer programmed grammars (resp. operating under leftmost interpretation), i.e., programmed grammars where each production has identical success and failure fields. It is well-known that

$$L_2 \subset P_{ut}\text{ (left)} \subset P_{ac}$$

but the mutual relation between the families P and P_{ut} is open. The emptiness problem is decidable for P_{ut}, undecidable for P_{ac}, and open for P. Finally, we denote by R_{fc} the family of languages generated by λ-free context-free grammars with a regular control language operating under full checking, i.e., appearance checking is possible for every production. The family R_{fc}(left) is defined

similarly. Full checking was introduced in [7]. Our first result gives a new characterization for the family P_{ut}.

Theorem 1. $P_{ut} = R_{fc}$.

Proof. The inclusion of the left side in the right side is easy to establish by the technique of [9, p. 177]. In fact, the situation here is even somewhat simpler than in the quoted reference because we do not have to introduce two copies of the productions.

To prove the reverse inclusion $R_{fc} \subseteq P_{ut}$, we consider an arbitrary language L generated by a λ-free context-free grammar $G = (V_N, V_T, X_0, F)$ with a regular control language operating under full checking. Assume, furthermore, that the control language is accepted by the finite deterministic automaton with state set S, initial state s_0, final state set S_1, transition function δ, and input alphabet equal to the alphabet of labels for the productions in F. An unconditional transfer programmed grammar G_1 for L will now be defined. The nonterminal alphabet of G_1 is obtained by adding to V_N the letters

$$Y \text{ and } [\alpha, s], \quad \alpha \in V_N \cup V_T, \quad s \in S.$$

The terminal alphabet of G_1 is V_T, and the initial symbol $[X_0, s_0]$. For a nonempty word x over $V_N \cup V_T$ and $s \in S$, we denote by x^s the word obtained from x by replacing the last letter α with the letter $[\alpha, s]$. For a production $A \to x$ in F labeled by t, we let $A^s \to x^{\delta(s,t)}$ labeled by t^s, and $A^s \to x$ labeled by $t^s(\text{final})$ be productions of G_1. The production set of G_1 consists of the productions appearing in the following matrices, where $\beta(s, \alpha)$ denotes the

sequence consisting of all productions of the form

$$[\alpha^1, s^1] \to Y \quad , \quad s^1 \neq s \quad , \quad \alpha^1 \neq \alpha$$

in some order:

(4) $\quad [\beta(s,\alpha) \ , \ t \ , \ [\alpha, s] \to [\alpha, \delta(s,t)] \ , \ \alpha \notin A \ ,$

$(4)^1$ $\quad [\beta(s,\alpha) \ , \ [\alpha, s] \to \alpha \ , \ \alpha \to [\alpha, s]]$

(5) $\quad [t^S]$

(6) $\quad [\beta(s,\alpha), \ t \ , \ [\alpha, s] \to \alpha]$ if $\alpha \notin A$ and $\delta(s,t) \in S_1$,

(7) $\quad [t^S(\text{final})]$ if $\delta(s,t) \in S_1$.

(Throughout this paper, Y stands for a "garbage" letter which can never be eliminated once it is introduced.) The go-to fields are defined in the following way. From the last productions of the matrices (4), $(4)^1$ and (5) it is possible to go to the first production of any matrix. The go-to fields of the last productions of (6) and (7) are empty. The go-to field of any production in (4), $(4)^1$ or (6) different from the last one consists of the next production in the same matrix. It is easy to see that L is generated by G_1 and this completes the proof.

The proof of the next theorem is similar and is, therefore, omitted.

Theorem 2. $P_{ut}(\text{left}) = R_{fc}(\text{left})$.

The results of Theorems 1 and 2 hold true also for the corresponding families with erasing productions. Our next theorem gives a binary normal form for grammars of languages in P_{ac} .

Theorem 3. Every language in P_{ac} is generated by a grammar, all of whose productions are of one of the two forms

(8) $A \to x$, $x \in V_N^2 \cup V_N \cup V_T$,

(9) $A \to Y$, $Y \in V_N$,

where the failure fields of productions (8) and success fields of productions (9) are empty.

Proof. Given a programmed grammar for a language in P_{ac}, we first eliminate terminals except from productions of the form $A \to a$ by the technique of [9, p. 19]. We then apply the technique of [9, p. 177] to get a grammar satisfying the requirements, except that x in (8) might belong to V_N^i, for some $i > 2$. Finally, the reduction technique of [9, p. 56] is applied, which proves the theorem.

It is to be noted that we have not been able to eliminate from (8) productions of the form $A \to B$, where B is a nonterminal. Such an elimination might be very difficult for programmed grammars.

Our last theorem in this section proves the rather surprising result that the inclusion corresponding to (3) does not hold true for unconditional transfer programmed grammars. (Note that all inclusions in this paper are effective in the sense that, given a device for a language in the smaller family, we can effectively produce a device for the same language in the larger family. Such an effective inclusion is meant also in the statement of the next theorem.)

Theorem 4. The family P_{ut} is not included in the family $P_{ut}(\text{left})$.

Proof. Assuming the contrary, we show how to solve the emptiness problem for the family P_{ac}. This is a contradiction because every recursively enumerable language is a homomorphic image of a language in P_{ac}.

Given a grammar for a language L in P_{ac}, we first transform it to an equivalent grammar G satisfying the requirements of Theorem 3. We now construct an unconditional transfer programmed grammar G_1 (operating under free interpretation) as follows. The only production for the initial letter X_0^1 of G_1 is $X_0^1 \to BX_0$, where X_0 is the initial letter of G and B is a new nonterminal. The productions (9) are taken to be productions of G_1, too, with their success field made identical to their failure field. The productions (8) are replaced by the matrices

(10) $$[B \to B_A,\ A \to B_A,\ B_A \to x,\ B_A \to B],$$

where B_A is a new nonterminal and the go-to fields are defined in the obvious fashion. (If (8) is in the field of some production, then $B \to B_A$ is added to this field. The productions in (10) have to be applied in the order indicated. The common success and failure field of the production $B_A \to B$ equals the success field of (8).) Finally the production $B \to d$, where d is a new terminal, is added. The go-to field of this production is empty, and this production is added to the go-to field of (9) and to that of the last production in (10). It is now easy to verify that L is nonempty iff d occurs as an initial subword of a word in $L(G_1)$. However by [2], this property is decidable for languages in P_{ut}(left). This completes the proof.

In our estimation, the open problems mentioned in this section, in particular the strictness of the inclusions in (2) and (3), are difficult ones. As regards the open problem of determining the relation between the families $P(left)$ and $P_{ut}(left)$, our results in Section 4 show that these families coincide with the family of context-sensitive languages when control is introduced for sets of productions rather than for single productions.

3. Graph controlled tables

A context-free grammar with graph controlled tables is an ordered sixtuple

$$(11) \qquad G = (V_N, V_T, X_0, F, \{F_1, \ldots, F_n\}, \varphi) ,$$

where $G_1 = (V_N, V_T, X_0, F)$ is a labeled context-free grammar, i.e., each production in the set F is provided with a label and there may be several copies of the same production with different labels, $n \geq 1$ and $F_i \subseteq F$, for $i = 1, \ldots, n$, and φ is a directed graph whose nodes are the sets F_i, $i = 1, \ldots, n$. We denote $V = V_N \cup V_T$, and define yield relations of several types. All relations are defined in the Cartesian prduct

$$V^* \times \{F_1, \ldots, F_n\} .$$

The relations are defined for a fixed grammar (11) and, thus, G could be added as an index to the name of the relation.

By definition, $(x, F_i) \Rightarrow (y, F_j)$ iff both of the following conditions are satisfied: (i) For some x_1, x_2, A and α, $x = x_1 A x_2$, $y = x_1 \alpha x_2$,

and the production $A \to \alpha$ is in the set F_i ; (ii) There is an edge from F_i to F_j in φ .

By definition, $(x, F_i) \overset{left}{\Rightarrow} (y, F_j)$ iff conditions (i) and (ii) above hold and, furthermore, x_1 does not contain any letters appearing on the left sides of the productions in F_i .

By definition, $(x, F_i) \Rightarrow_{ut} (y, F_j)$ iff either $(x, F_i) \Rightarrow (y, F_j)$, or else condition (ii) is satisfied, x does not contain any letters appearing on the left sides of the productions in F_i , and $x = y$.

Finally, by definition $(x, F_i) \overset{left}{\Rightarrow}_{ut} (y, F_j)$ iff either $(x, F_i) \overset{left}{\Rightarrow} (y, F_j)$, or else condition (ii) is satisfied, x does not contain any letters appearing on the left sides of the productions in F_i , and $x = y$.

For any of the four relations δ thus defined, we denote by δ^* its reflexive transitive closure.

We define now $GC_{\alpha}^{\lambda}(\beta)$, where α is either missing or equal to ut and (β) is either missing or equal to (left) , to be the family of languages of the form

$$\{x \in V_T^* \mid (X_0, F_i) \delta^* (x, F_j) \text{ , for some } i \text{ and } j\} \text{ ,}$$

for some grammar (11) and relation δ such that ut and left appear simultaneuosly in the name of the relation and in the name of the family. By omitting λ from the name of the family we indicate that only such grammars (11) are considered, where all productions in F are λ-free. We have, thus, defined the following eight language families:

(12) GC^{λ} , GC_{ut}^{λ} , $GC^{\lambda}(left)$, $GC_{ut}^{\lambda}(left)$,
GC , GC_{ut} , $GC(left)$, $GC_{ut}(left)$,

Note that the situation in analoguos to the one considered in the previous section and we have, in fact, defined variations of the families P and P_{ut}.

4. Generation of context-sensitive languages

In this section we show that both of the families GC(left) and GC_{ut}(left) equal the family L_1 of context-sensitive languages. Graph control is perhaps the simplest control device so far introduced which yields, operating on context-free core productions, the family of context-sensitive languages.

Theorem 5. GC(left) = L_1.

Proof. The inclusion of the left side in the right side is obvious, either by the workspace theorem of [9] or by a simulation by a linear bounded automaton.

To prove the reverse inclusion, we consider an arbitrary context-sensitive language L. (Throughout this paper, we assume that the empty word is not contained in a context-sensitive language.) By a result due to Kuroda, cf. [6] and [1], L is generated by a grammar G_1 with productions of the forms

(13) $A \to BC$,

(14) $AB \to CD$,

(15) $A \to a$,

where capital letters denote nonterminals and a is a terminal.

Furthermore, the following assumptions can be made without less of generality. Every word possesses a derivation, where all applications of productions (13) precede all applications of productions (14) which, in turn, precede all applications of productions (15) and, finally, productions (13) are applied always to the leftmost letter only. (The letter assumption is due to the fact that it suffices to assume that productions (13) are used to generate the language TU^*, for some nonterminals T and U.)

We now define a context-free grammar (11) with graph controlled tables which, in the sense of relation $\overset{left}{\Rightarrow}$, generates L. The set of nonterminals is obtained by adding to the set of nonterminals A of G_1 their "primed versions" A' and A'', as well as a new garbage nonterminal Y. The productions and tables will now be defined.

All productions (13) (resp. (15)) constitute a table denoted by F_E (resp. F_T). All productions $A \to A'$ (resp. $A' \to A$), where A ranges over nonterminals of G_1, constitute a table denoted by F_{prime} (resp. $F_{nonprime}$). For each of the productions (14), we introduce three tables $F_1^{(i)}$, $F_2^{(i)}$, $F_3^{(i)}$, for $i = 1, \ldots, k$ where k is the number of productions (14). The table $F_1^{(i)}$ consists of the production $A \to C'$ alone. The table $F_2^{(i)}$ consists of the production $B \to D''$ and of the productions $X \to Y$, where X ranges over all nonterminals of G_1 different from B. The table $F_3^{(i)}$ consists of the production $D'' \to D$ and of the productions $X' \to Y$, where X ranges over all nonterminals of G_1. (We may assume that all of the tables $F_j^{(i)}$ are distinct by introducing several copies of the productions whenever necessary.)

Finally, the graph φ will be defined. From F_E there is an edge to F_E, F_{prime}, F_T, and each $F_1^{(i)}$. From F_T there is an edge to F_T only. From F_{prime} there is an edge to itself and to each $F_1^{(i)}$. From $F_1^{(i)}$ there is an edge to $F_2^{(i)}$. From $F_2^{(i)}$ there is an edge to $F_{nonprime}$. From $F_{nonprime}$ there is an edge to itself and to each $F_3^{(i)}$. From $F_3^{(i)}$ there is an edge to F_T, F_{prime} and to each $F_1^{(j)}$.

This completes the definition of the grammar (11). We leave to the reader to verification of the fact that the construction has desired effect.

Theorem 6. $GC_{ut}(left) = L_1$.

Proof. Again, the inclusion of the left side in the right side is immediate. The proof of the reverse inclusion is obtained from the preceding proof by the following modifications in the constructions of the grammar (11). The starting configuration is $X_0\#$, where $\#$ is a new nonterminal. The production $\# \to b$, where b is a terminal, is added to the terminal table F_T. The production $\# \to Y$ is added to each of the tables $F_1^{(i)}$ and $F_2^{(i)}$. (This is to make sure that these tables are not applied in the appearence checking sense.) Now the modified grammar (11) generates in the sense of the relation $\overset{left}{\Rightarrow}_{ut}$ the language Lb. We, thus, have the following result: for each context-sensitive language L and terminal symbol b, the language Lb is in the family $GC_{ut}(left)$. The proof is now concluded by the argument in [9, p. 152].

5. Corollaries and concluding remarks

Our next theorem follows immediately by Theorems 5 and 6 and [9, p. 90].

Theorem 7. $GC^{\lambda}(\text{left}) = GC^{\lambda}_{ut}(\text{left}) = L_0$.

As regards the remaining families introduced in (12), we are able to state the following results.

Theorem 8. $GC^{\lambda} = P^{\lambda}$, $GC = P$, $P_{ut} \subseteq GC_{ut} \subseteq P_{ac}$.

Proof. In the first place it is immediate that any P-family is contained in the corresponding GC-family because P-grammars can be considered as grammars with tables where each table consists of one production only. The reverse inclusions needed for the two equations follow because if we have no appearance checking and no left control, we can simulate any table by single productions in the obvious way. Finally, the last inclusion in the statement is established by the following construction. Given a GC_{ut}-grammar, we construct a P_{ac}-grammar by splitting all tables into single productions. Any one of these productions can be entered whenever the table could be entered. The success fields are obtained directly from the original graph, whereas the failure fields lead to a sequence of productions, where each nonterminal appearing on the left side in the table is mapped into garbage, after which the continuation is as in the original graph. This completes the proof.

It remains an open problem whether or not the inclusions in Theorem 8 are proper. In particular, as regards the first inclusion, we have not been able to settle the question whether or not the additional appearance checking mechanism for the whole table increases

the generative capacity with respect to P_{ut}. As regards the remaining family GC^{λ}_{ut}, we can of course prove the inclusions of Theorem 8 for the P-families with λ. It seems likely that the much stronger result

$$GC^{\lambda}_{ut} \subset L_1$$

(modulo λ) holds true. The proof of this involves problems similar to those still open for the family P^{λ}.

In this paper, we have generalized the notion of a programmed grammar in a most natural way. Surprisingly enough, this yielded the family of context-sensitive languages (or recursively enumerable languages if erasing is allowed) in two different fashions, corresponding to the simplest classes of languages generated by programmed grammars.

We would like to point out the duality between our results and those of [8] concerning matrix grammars with the leftmost restriction. A matrix grammar can be viewed as a collection of sets of productions (tables) with no control (graph) on the tables themselves but a control (linear order) within each table. In a grammar with graph controlled tables, there is no control within a table but there is a control (graph) on the tables themselves. In both ways one gets the same families of languages.

Acknowledgement. The authors are grateful to Mogens Nielsen, Martti Penttonen and Sven Skyum for useful discussions.

References

[1] M. Penttonen, One-sided and two-sided context in formal grammars. Information and Control 25 (1974) 371-392.

[2] D. Rosenkrantz, Programmed grammars and classes of formal languages. J. Assoc. Comput. Mach. 16 (1969) 107-131.

[3] G. Rozenberg, TOL systems and languages. Information and Control 23 (1973) 357-381.

[4] G. Rozenberg, Direction controlled programmed grammars. Acta Informatica 1 (1972) 242-252.

[5] G. Rozenberg and A. Salomaa (ed.), L Systems. Springer Lecture Notes for Computer Science, Vol. 15, (1974).

[6] A. Salomaa, Theory of Automata. Pergamon Press, Oxford (1969).

[7] A. Salomaa, On grammars with restricted use of productions. Ann. Acad. Sci. Fennicae, Ser. AI 454 (1969).

[8] A. Salomaa, Matrix grammars with a leftmost restriction. Information and Control 20 (1972) 143-149.

[9] A. Salomaa, Formal Languages. Academic Press, New York (1973).