J. Dassow, B. Truthe (Eds.): Colloquium on the Occasion of the 50th Birthday of Victor Mitrana.
Otto von Guericke Universität Magdeburg, Germany, June 27, 2008. Proceedings, pages 29 – 36.

On the Complexity of the Control Language in Tree Controlled Grammars

RALF STIEBE

Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik
PSF 4120, D-39016 Magdeburg, Germany

stiebe@iws.cs.uni-magdeburg.de

Abstract: It is shown that any context-sensitive language can be generated by a tree controlled grammar whose control language is accepted by a deterministic finite automaton with at most 5 states.

Keywords: Tree controlled grammar, regular control, state complexity.

1. Introduction

Regulated rewriting is a common means to increase the generative power of context-free grammars. A context-free core grammar is combined with a mechanism which controls the derivation process. For a thorough introduction to regulated rewriting, see [2, 1]. In tree controlled grammars, introduced by Čulik and Maurer [5], the structure of the derivation trees is restricted as all words belonging to a level of the derivation tree have to be in a given regular language. It was shown by Păun [4] that tree controlled grammars with λ-free context-free core grammars are equivalent to context-sensitive grammars.

Recently, Dassow and Truthe [3] have investigated the generative power of tree controlled grammars when restricting the control language to subfamilies of the regular languages. In particular, they discussed the generative power with respect to the state complexity of the control language. The problem whether the hierarchy defined by the state complexity collapses was left open. We will settle this problem by showing that a state complexity of 5 for the control language is sufficient to obtain the full generative power of tree controlled grammars.

2. Definitions and Basic Notations

We assume that the reader is familiar with the basic concepts of formal language theory. The families of regular, context-free, context-sensitive, recursively enumerable languages are denoted by *REG*, *CF*, *CS*, *RE*, respectively.

For a derivation tree $\mathcal{T}$ of height k for a derivation in a context-free grammar and a number $0 \leq j \leq k$, the *word of level* j is given by the nodes of depth j read from left to right.

Definition 1. *A* tree controlled grammar *is a tuple*

$$G = (N, T, P, S, R),$$

where $G' = (N, T, P, S)$ *is a context-free grammar and* $R \subseteq (N \cup T)^*$ *is a regular language.*

The language $L(G)$ *consists of all words* $w \in T^*$ *generated by* G' *with a derivation tree whose words of all levels (except the last one) are in* R*.*

Let $\mathcal{L}$ be a subfamily of *REG*. By $\mathcal{TC}_\lambda(\mathcal{L})$ ($\mathcal{TC}(\mathcal{L})$, respectively) we denote the family of languages generated by tree controlled grammars with control languages from $\mathcal{L}$ and arbitrary context-free core grammars (λ-free context-free core grammars, respectively). It is known that $\mathcal{TC}_\lambda(\mathit{REG}) = \mathit{RE}$ and $\mathcal{TC}(\mathit{REG}) = \mathit{CS}$ [4].

For $n \geq 1$, let REG_n be the family of languages that are accepted by deterministic finite automata with at most n states. Dassow and Truthe studied the families $\mathcal{TC}(\mathit{REG}_n)$ and obtained the following results, where *E0L* and *ET0L* denote the families generated by E0L and ET0L systems.

Theorem 2 [3].

1. $\mathcal{TC}(\mathit{REG}_1) \subseteq \mathcal{TC}(\mathit{REG}_2) \subseteq \mathcal{TC}(\mathit{REG}_3) \subseteq \cdots \subseteq \mathcal{TC}(\mathit{REG}) = \mathit{CS}$.
2. $\mathit{E0L} = \mathcal{TC}(\mathit{REG}_1) \subset \mathcal{TC}(\mathit{REG}_2)$.
3. $\mathit{ET0L} \subset \mathcal{TC}(\mathit{REG}_4)$.

The principal proof technique for our result will be the simulation of *queue automata* by tree controlled grammars. Intuitively, a queue automaton consists of a finite control with a queue as storage. In a step of the automaton, the first symbol of the queue is removed and a sequence of symbols is appended to the end of the queue.

Definition 3. *A* queue automaton *is a tuple* $\mathcal{A} = (Z, \Sigma, \Gamma, \delta, z_0, F)$*, where* Z *is the finite set of states,* Σ *is the input alphabet,* Γ *is the tape alphabet with* $\Sigma \subseteq \Gamma$*,* $\delta \subset Z \times \Gamma \times Z \times \Gamma^+$ *is the finite transition relation,* $z_0 \in Z$ *is the initial state,* $F \subseteq Z$ *is the set of accepting states.*

The automaton $\mathcal{A}$ *is called a* linearly bounded *queue automaton if* $\delta \subseteq Z \times \Gamma \times Z \times \Gamma$*.*

A configuration of $\mathcal{A}$ *is given by a pair* (z, w)*, where* $z \in Z$*,* $w \in \Gamma^+$*. The successor relation* $\vdash$ *on the set of configurations is defined as* $(z, aw) \vdash (z', wx)$ *iff* $(z, a, z', x) \in \delta$*.*

The language accepted by $\mathcal{A}$*,* $L(\mathcal{A})$*, is defined as*

$$L(\mathcal{A}) = \{w \in \Sigma^+ : (z_0, w) \vdash^* (z_f, y), \text{ for some } z_f \in F, y \in \Gamma^*\}.$$

It is well-known that the language family accepted by queue automata is equal to the family of recursively enumerable languages. The proof is usually performed by constructing an equivalent queue automaton from a given Turing machine, and vice versa. These constructions preserve linear boundedness. Hence, the language family accepted by linearly bounded queue automata equals the family of context-sensitive languages. Moreover, the following technical result on queue automata can be easily shown analogously to similar results for Turing machines (we leave the proof to the reader).

Lemma 4. *Any recursively enumerable (context-sensitive) language can be accepted by a (linearly bounded) queue automaton* $\mathcal{A} = (Z, \Sigma, \Gamma, \delta, z_0, \{q\})$*, such that*

- *all reachable accepting configurations of* $\mathcal{A}$ *have the form* $(z_f, \square^n), n \geq 1$*, for a special symbol* $\square \in \Gamma \setminus \Sigma$ *(called the blank symbol);*
- $\delta \cap \{q\} \times \Gamma \times Z \times \Gamma^+ = \emptyset$ *(the accepting state* q *has no successor);*
- $\delta \cap Z \times \Gamma \times \{z_0\} \times \Gamma^+ = \emptyset$ *(the initial state* z_0 *has no predecessor).*

3. The Result

The idea of the construction is to rewind the accepting computation of a linearly bounded queue automaton by means of a tree controlled grammar. We will first give a simple construction where the size of the deterministic finite automaton for the control language depends on the size of the tape alphabet of the queue automaton. Later, this construction will be refined to limit the number of states by 5.

For a Cartesian product $X_1 \times X_2 \times \cdots \times X_n$, let $\mathrm{pr}_i : X_1 \times X_2 \times \cdots \times X_n \to X_i$ denote the projection on the i-th component, i. e., the mapping

$$\mathrm{pr}_i : X_1 \times X_2 \times X_n \to X_i$$

with

$$\mathrm{pr}_i(x_1, x_2, \ldots, x_n) = x_i.$$

Lemma 5. *For any linearly bounded queue automaton* $\mathcal{A}$*, there is a tree controlled grammar* G *such that* $L(G) = L(\mathcal{A})$*.*

Proof. Let $\mathcal{A} = (Z, \Sigma, \Gamma, \delta, z_0, \{q\})$ be in the normal form as in Lemma 4 with the blank symbol $\square$.

The tree controlled grammar G is obtained as $G = (N, \Sigma, P, S, R)$, where

$$
\begin{aligned}
N &= N_1 \cup N_2, \\
&\quad N_1 = \Gamma \times \Gamma, \\
&\quad N_2 = \Gamma \times \Gamma \times Z, \\
P &= \{p_1\} \cup P_2 \cup P_3 \cup P_4, \\
&\quad p_1 = (\square, \square, q) \to (\square, \square, q)(\square, \square), \\
&\quad P_2 = \{(a, x) \to (y, a) : a, x, y \in \Gamma\}, \\
&\quad P_3 = \{(b, x, z') \to (y, a, z) : x, y \in \Gamma, (z, a, z', b) \in \delta\}, \\
&\quad P_4 = \{(a, b) \to b, (a, b, z_0) \to b : a, b \in \Sigma\}, \\
S &= (\square, \square, q), \\
R &= \{A_1 A_2 \cdots A_n : n \geq 1, A_1 \in N_2, A_i \in N_1 \text{ for } 2 \leq i \leq n, \\
&\qquad \text{pr}_1(A_1) = \text{pr}_2(A_n), \text{pr}_1(A_i) = \text{pr}_2(A_{i-1}) \text{ for } 2 \leq i \leq n\}.
\end{aligned}
$$

A word in R can be seen as the encoding of a configuration of $\mathcal{A}$. More specifically, a configuration $(z, a_1 a_2 a_3 \cdots a_{n-1} a_n)$ is encoded by

$$(a_n, a_1, z)(a_1, a_2)(a_2, a_3) \ldots (a_{n-1}, a_n) \in R.$$

We now consider the tree of a successful derivation in G in detail. As noted above, all level words (except the last one) are encodings of configurations of $\mathcal{A}$. On the root level we find the word $(\square, \square, q)$, i. e., the encoding of the accepting configuration of length 1. Now suppose that some level contains a word $(\square, \square, q)(\square, \square)^{j-1}$, encoding the accepting configuration of length j. If the first symbol is replaced using rule p_1, the next level must have the form $(\square, \square, q)(\square, \square)(x_1, \square) \cdots (x_{j-1}, \square)$, as the remaining symbols are replaced using rules from P_2. The control language requires that $x_i = \square$, $1 \leq i \leq j-1$, and thus the next level word is $(\square, \square, q)(\square, \square)^j$, encoding the accepting configuration of length $j+1$.

Next, consider a level encoding a non-initial configuration $(z', a_1 a_2 \cdots a_n)$ where $z' \neq z_0$, i. e., with the word $(a_n, a_1, z')(a_1, a_2)(a_2, a_3) \cdots (a_{n-1}, a_n)$. The first symbol has to be rewritten using a rule from P_3, the remaining symbols are rewritten using P_2, giving a word of the form $(x_n, a_0, z)(x_1, a_1)(x_2, a_2) \cdots (x_{n-1}, a_{n-1})$, where $(z, a_0, z', a_n) \in \delta$. In view of the control language R, $x_i = a_{i-1}$ has to hold, for $1 \leq i \leq n$. Hence, the next level word describes a configuration $(z, a_0 a_1 a_2 \cdots a_{n-1})$ with $(z, a_0, z', a_n) \in \delta$, i. e., a predecessor configuration. On the other hand, for any predecessor configuration, the encoding word can be obtained at the next level by choosing for the replacement of the first symbol that rule from P_3 which corresponds to the appropriate transition and for the replacement of the other symbols the appropriate rules from P_2.

Finally, consider a level encoding a configuration $(z_0, a_1 a_2 \cdots a_n)$, i. e., with the word $(a_n, a_1, z_0)(a_1, a_2)(a_2, a_3) \cdots (a_{n-1}, a_n)$. The only possibility to rewrite the first symbol is to use the rule $(a_n, a_1, z_0) \to a_1$ if $a_n, a_1 \in \Sigma$. Hence the next level of the derivation tree is the final. The remaining symbols have to be rewritten using rules of P_4, implying

that $a_1, a_2, \ldots, a_n \in \Sigma$ and giving the word $a_1 a_2 \cdots a_n$ as the next level and as the yield of the derivation.

Consequently, a terminal word is generated by G iff it is accepted by $\mathcal{A}$. □

A deterministic finite automaton accepting the control language R in the above proof requires $|\Gamma|^2 + 2$ states, as it must store the second component of the current symbol for comparison with the next symbol and the first component of the first symbol for comparison with the last symbol; moreover two separate initial and failure states are needed. To construct a tree controlled grammar with a control language with a fixed number of states, we modify the grammar as follows. The symbols of the queue automaton are encoded by a bit vector of length $k = \lceil \log_2 |\Gamma| \rceil$. A reverse computation step of $\mathcal{A}$ is simulated in k derivation levels of the tree controlled grammar. In each sub-step, one bit is passed from a symbol to its right neighbour. The details of the construction will be given in the proof of the following theorem.

Theorem 6. $\mathcal{TC}(REG_5) = CS$.

Proof. Let $\mathcal{A} = (Z, \Sigma, \Gamma, \delta, z_0, \{q\})$ be a linearly bounded queue automaton as in the proof of Lemma 5 with the blank symbol $\square \in \Gamma$. Let $k = \lceil \log_2 |\Gamma| \rceil$ and let $\varphi : \Gamma \to \{0,1\}^k$ be an encoding of Γ with $\varphi(\square) = (0, 0, \ldots, 0)$.

The tree controlled grammar G is obtained as $G = (N, \Sigma, P, S, R)$, where

$$
\begin{aligned}
N &= N_1 \cup N_2, \\
&\quad N_1 = \{0,1\}^{k+1}, \\
&\quad N_2 = \{0,1\}^{k+1} \times Z \times \{1, 2 \ldots, k\}, \\
P &= \{p_1\} \cup P_2 \cup P_3 \cup P_4 \cup P_5, \\
&\quad p_1 = (0^{k+1}, q, 1) \to (0^{k+1}, q, 1) 0^{k+1}, \\
&\quad P_2 = \{(a_1, \ldots, a_k, a_{k+1}) \to (y, a_1, \ldots, a_k) : a_1, \ldots, a_{k+1}, y \in \{0,1\}\}, \\
&\quad P_3 = \{(a_1, \ldots, a_k, a_{k+1}, z, i) \to (y, a_1, \ldots, a_k, z, i+1) : \\
&\qquad\qquad a_1, \ldots, a_{k+1}, y \in \{0,1\}, z \in Z, 1 \le i < k\}, \\
&\quad P_4 = \{(\varphi(b), x, z', k) \to (y, \varphi(a), z, 1) : x, y \in \{0,1\}, (z, a, z', b) \in \delta\}, \\
&\quad P_5 = \{(y, \varphi(b)) \to b, (y, \varphi(b), z_0, 1) \to b : b \in \Sigma, y \in \{0,1\}\}, \\
S &= (0^{k+1}, q, 1), \\
R &= \{A_1 A_2 \cdots A_n : n \ge 1, A_1 \in N_2, A_i \in N_1 \text{ for } 2 \le i \le n, \\
&\qquad\qquad \mathrm{pr}_1(A_1) = \mathrm{pr}_{k+1}(A_n), \mathrm{pr}_1(A_i) = \mathrm{pr}_{k+1}(A_{i-1}) \text{ for } 2 \le i \le n\}.
\end{aligned}
$$

We set $N_{2,i} = \{0,1\}^{k+1} \times Z \times \{i\}$, for $1 \le i \le k$. A word from R encodes a configuration of the queue automaton as follows. A configuration $(z, a_1 a_2 a_3 \cdots a_{n-1} a_n)$ is encoded by

$$(\mathrm{pr}_k(\varphi(a_n)), \varphi(a_1), z, 1)(\mathrm{pr}_k(\varphi(a_1)), \varphi(a_2))(\mathrm{pr}_k(\varphi(a_2)), \varphi(a_3)) \ldots (\mathrm{pr}_k(\varphi(a_{n-1})), \varphi(a_n)).$$

Similar to the proof of Lemma 5, we will now discuss the successful derivation trees in G. On the root level, we find the word $S = (0^{k+1}, q, 1)$, which encodes the accepting configuration of length 1. If the word of some level encodes the accepting configuration of length j and rule p_1 is applied to the first symbol, then the next level encodes the accepting configuration of length $j+1$.

Now consider a level word $\alpha_1 = A_1 A_2 \cdots A_n$ encoding a configuration. The symbols have the forms

$$A_1 = (a_{n,k}, a_{1,1}, a_{1,2} \ldots, a_{1,k}, z', 1),$$
$$A_i = (a_{i-1,k}, a_{i,1}, a_{i,2}, \ldots, a_{i,k}), \text{ for } 2 \leq i \leq n.$$

As α_1 encodes a configuration of the queue automaton, $(a_{i,1}, a_{i,2}, \ldots, a_{i,k}) = \varphi(x_i)$ has to hold for appropriate $x_i \in \Gamma$, $1 \leq i \leq n$. The symbol A_1 has to be rewritten using a rule from P_3 (with the exception of $z' = z_0$, discussed below), which implies that the remaining symbols are replaced using rules of P_2. In view of R, the next level has to be labelled $\alpha_2 = A_1^2 A_2^2 \cdots A_n^2$ with

$$A_1^2 = (a_{n,k-1}, a_{n,k}, a_{1,1}, a_{1,2} \ldots, a_{1,k-1}, z', 2),$$
$$A_i^2 = (a_{i-1,k-1}, a_{i-1,k}, a_{i,1}, a_{i,2}, \ldots, a_{i,k-1}), \text{ for } 2 \leq i \leq n.$$

By analogous arguments for words in $N_{2,j} N_1^*$, $2 \leq j < k$, we obtain after $k-1$ levels the word $\alpha_k = A_1^k A_2^k \cdots A_n^k \in R$ with

$$A_1^k = (a_{n,1}, a_{n,2}, \ldots, a_{n,k}, a_{1,1}, z', k),$$
$$A_i^k = (a_{i-1,1}, a_{i-1,2}, \ldots, a_{i-1,k}, a_{i,1}), \text{ for } 2 \leq i \leq n.$$

On the next level, A_1^k is replaced using a rule from P_4 and the remaining symbols using a rule from P_2. One obtains a word $\alpha_{k+1} = A_1^{k+1} A_2^{k+1} \cdots A_n^{k+1} \in R$ with

$$A_1^{k+1} = (a_{n-1,k}, b_{1,1}, \ldots, b_{1,k}, z, 1),$$
$$A_2^{k+1} = (b_{1,k}, a_{1,1}, \ldots, a_{1,k}),$$
$$A_i^{k+1} = (a_{i-2,k}, a_{i-1,1}, a_{i-1,2}, \ldots, a_{i-1,k}) = A_{i-1}, \text{ for } 3 \leq i \leq n,$$

where $(b_{1,1}, \ldots, b_{1,k}) = \varphi(x_0)$, $(z, x_0, z', x_n) \in \delta$. Hence, the configuration encoded by α_{k+1} is a predecessor of that encoded by α_1. On the other hand, the encoding of any predecessor configuration can be reached by choosing the appropriate rules.

Finally, if and only if a level word describes an initial configuration of $\mathcal{A}$, the input word can be reached as terminal word on the next level by using the rules of P_5.

The control language R can be accepted by a deterministic finite automaton with six states. However, note that the rules of G imply that any derivable sentential form over N is a word from $N_2 N_1^*$. Instead of R, we can use any regular language R' such

that $R' \cap N_2 N_1^* = R$. Such a language is the one accepted by the deterministic finite automaton

$$\mathcal{M} = (\{z_{00}, z_{01}, z_{10}, z_{11}, \mathit{fail}\}, N \cup \Sigma, f, z_{00}, \{z_{00}, z_{11}\})$$

with the transition function f defined as

$$\begin{aligned} f(z_{ab}, (b, a_1, \ldots, a_k)) &= z_{aa_k}, \text{ for } a, b, a_1, \ldots, a_k \in \{0,1\}; \\ f(z_{ab}, (a_0, a_1, \ldots, a_k, z, i)) &= z_{a_0 a_k}, \text{ for } a, b, a_0, a_1, \ldots, a_k \in \{0,1\}, z \in Z, i \in \{1, \ldots, k\}; \\ f(z, A) &= \mathit{fail}, \text{ in all other cases.} \end{aligned}$$

Obviously, $\mathcal{M}$ accepts only words over N. When receiving an input from $N_2 N_1^*$, $\mathcal{M}$ works as follows. A state of the form z_{ab} is meant to store two bits: the first bit of the symbol from N_2 is a, while the last bit of the currently read symbol is b. If the first bit of the next symbol is unequal to the stored one, the input is rejected. Finally, $\mathcal{M}$ accepts iff it reaches a state z_{aa}, thus if additionally the last bit of the last symbol is equal to the first of the first one. □

Corollary 7. $\mathcal{TC}_\lambda(\mathit{REG}_5) = \mathit{RE}$.

Proof. Note that any recursively enumerable language L' can be expressed as $h(L)$ for appropriate homomorphism h and context-sensitive language L. In the construction of the tree controlled grammar for L as in the proof of Theorem 6, one has just to change the rules in P_5 by replacing on the right-hand sides the letters from Σ by their images under h. □

References

[1] J. DASSOW, GH. PĂUN, and A. SALOMAA, Grammars with controlled derivations. In: G. ROZENBERG and A. SALOMAA (eds.), *Handbook of Formal Languages, Volume II*, Springer-Verlag, Berlin, 1997, 101–154.

[2] J. DASSOW and GH. PĂUN, *Regulated Rewriting in Formal Language Theory*, EATCS Monographs in Theoretical Computer Science 18, Springer-Verlag, Berlin, 1989.

[3] J. DASSOW and B. TRUTHE, On Two Hierarchies of Subregularly Tree Controlled Languages. In: C. CÂMPEANU and G. PIGHIZZINI (eds.), *10th International Workshop on Descriptional Complexity of Formal Systems, DCFS 2008, Charlottetown, Prince Edward Island, Canada, July 16–18, 2008, Proceedings*. University of Prince Edward Island, 2008, 145–156.

[4] GH. PĂUN, On the generative capacity of conditional grammars. *Information and Control* **43** (1979), 178–186.

[5] K. ČULIK and H. A. MAURER, Tree controlled grammars. *Computing* **19** (1977), 129–139.