ON VERTICAL RESTRICTIONS OF GRAMMATICAL DERIVATIONS

Ing. Tomáš KOPEČEK, Doctoral Degree Programme (1)
Dept. of Information Systems, FIT, BUT
E-mail: kopecek@fit.vutbr.cz

Supervised by: Dr. Alexander Meduna

ABSTRACT

Traditional formal language theory is interested in some versions of grammars which are restricted in some ways. Most of them have restrictions in the form, which applies to current or following sentence. The others use some regulation on rewriting mechanism.

We introduce new view — vertical restrictions. We could imagine derivation process as a table of derivations. When we set some boundaries going across the table we can see something like vertical splitting. In these terms we examine some restrictions and power of resulting grammars.

1 INTRODUCTION

We can imagine derivation process (in any type of grammar) as a table 1.

$$\begin{array}{lllllll} S & \Rightarrow & x_{11} & x_{12} & \dots & x_{1m} \\ & \Rightarrow & x_{21} & x_{22} & \dots & x_{2m} \\ & \vdots & & & & \\ & \Rightarrow & x_{n1} & x_{n2} & \dots & x_{nm} \end{array}$$

Table 1: Derivation process

We can see, that in each step we have split derived sentence into m strings. As we also know, most of the strings will be empty. In the advanced stages of rewriting there will be more and more non-empty strings. The final line will have all strings non-empty.

Based on this approach we define new grammar, which will be almost context-free. Context-sensitive productions will be limited to form $ab \rightarrow cd$ and application of these rules can be applied only on boundary. So, if $x_{11} = klma$ and $x_{12} = bpq$, we could apply previous rule on this boundary.

Further we restricts number of use of context-sensitive productions on each boundary. We can show, that under these restrictions is the power grammar (which looks like context-sensitive) limited to family of context-free languages.

2 DEFINITIONS

We assume that the reader is familiar with the language theory (see [1]).

Let V be an alphabet. The cardinality of V is denoted by $\#_V$. V^* represents the free monoid generated by V under the operation of concatenation. The unit of V is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$; algebraically, V^+ is thus the free semigroup generated by V under the operation of concatenation. For every symbol $X \in V, \#_X w$ denotes the number of occurrences of X in w.

A *context-free grammar* is a quadruple, $G = (V,T,P,S)$, where V is an alphabet, $T \subseteq V$, and $S \in V - T$. P is a finite set of productions of the form $A \to x$, where $A \in V - T$ and $x \in V^*$. If $A \to x \in P$ and $u,v \in V^*$, then $uAv \Rightarrow_G uxv$ in G. Let $\Rightarrow_G^*$ denote the transitive-reflexive closure of $\Rightarrow_G$. The *language of* G, $L(G)$, is defined as $L(G) = \{y : S \Rightarrow_G^* y, y \in T^*\}$.

An *extended finite automaton* is an n-tuple $M = (Q,\Sigma,R,s,F)$, where Q is a finite set of inner automaton states, Σ is an finite input alphabet, R is a finite set of transition rules of the form $pxy \to qy, p,q \in Q, x,y \in \Sigma^*$, s is a starting state of M, $s \in Q$ and $F \subseteq Q$ is a set of final states. Configuration of such automaton will be denoted by $\chi = qx, q \in Q, x \in \Sigma^*$, where x means the input string. Automaton makes a computation step (transition) from configuration $qxy, q \in Q, x,y \in \Sigma^*$ to $py, p \in Q$ denoted by $qxy \vdash_M py$ if exist transition rule $qx \to p \in R$. $\vdash_G^*$ denotes transitive and reflexive closure of relation $\vdash_M$. Language accepted by M is $L(M) = \{x : sx \vdash_M^* f, x \in \Sigma^*, f \in F\}$.

3 RESULTS

We define new grammar $G = (V,T,P,S)$ derived from Kuroda normal form. Its sets V,T and symbol S have the same meaning as in context-free grammars. Set P contains productions in these forms:

1. $AB \to CD, A,B,C,D \in V - T$
2. $A \to BC, A,B,C \in V - T$
3. $A \to a, A \in V, a \in T$
4. $A \to \varepsilon, A \in V$

It is a definition of Kuroda normal form. Now we attach some limitations to this grammar.

If G makes $xABy \Rightarrow xCDy$ by using $AB \to CD$, we say that $xABy$ is rewritten between xA and By in a context way.

Derivation process could be described by table 1. This derivation have to respect these rules:

1. $m \geq 1$

2. $|x_{ij}| \leq k$
3. for all $n = 2, \ldots, m$ there exists $x_{rn} \in V^+$ such that for all $q = 1, \ldots, n$ and $o = n+1, \ldots, m$, holds $x_{q_o} = \varepsilon$
4. for each $d = 1, \ldots, m-1$, there exist no more that l substrings of the form $x_{cd}x_{cd+1}$, where $1 \leq c \leq n$, such that $x_{cd}x_{cd+1}$ is rewritten between x_{cd} and x_{cd+1} in a context way.

We can see, that this grammar is very similar to Kuroda normal form. More precisely, there are only added restrictions on a number of context rewritings in one "column".

So, we can expect, that family of languages generated by this grammar will be equal to family of recursively enumerable languages.

Claim 1. *Grammar G generates precisely the family of regular languages.*

Proof. We prove claim 1 by simulating G with extended finite automaton. We now show construction of such automaton which accept language if and only if final state is reached and input tape is empty.

Construction 2. Construct $M = (Q, T, R, s, \{f\})$ in the following way:

- $Q = \{f\} \cup \{\langle A, u, \alpha, X, v\rangle : \langle A, u, \alpha, X, v\rangle \neq f$
  $A \in V, X \in T \cup \{\varepsilon\}, \alpha \in V^*, |\alpha| \leq k, u, v \in P^*_{CS}, |u|, |v| \leq l\}$
- $s = \langle S, \varepsilon, \varepsilon, \varepsilon, \varepsilon\rangle$
- R is constructed by performing following steps:
  1. For every $x \in T^*$ with $|x| \leq k$ add rules to R in this form:
     $\langle X, u, \varepsilon, \varepsilon, \varepsilon, \rangle x \to \langle X, u, x, \varepsilon, \varepsilon\rangle$.
  2. For every $a \in T \cup \{\varepsilon\}$ and rule $A \to a$ in G add these rules to R:
     $\langle X, u, \alpha a\beta, Y, v\rangle \to \langle X, u, \alpha A\beta, Y, v\rangle$
  3. For every $A \in N$ add these rules to R:
     $\langle X, \varepsilon, \varepsilon, Y, v\rangle \to \langle Y, v, \varepsilon, \varepsilon, \varepsilon\rangle$
  4. For every $AB \to CD$ in G add these rules into R:
     (a) $\langle X, u, \beta C, \varepsilon, v\rangle \to \langle X, u, \alpha A, \varepsilon, vAB \to CD\rangle$
     (b) $\langle X, AB \to CDu, D\beta, X, v\rangle \to \langle X, u, B\alpha, X, v\rangle$
  5. For every $A \to BC$ in G add these rules into R:
     (a) $\langle X, u, \alpha BC\beta, Y, v\rangle \to \langle X, u, \alpha A\beta, Y, v\rangle$
     (b) $\langle A, \varepsilon, B, \varepsilon, v\rangle \to \langle C, v, \varepsilon, \varepsilon, \varepsilon\rangle$
  6. Add $\langle A, \varepsilon, A, \varepsilon, \varepsilon\rangle \to f$ to R.

We should observe, that number of states and transition rules could be very large but definitely finite. So we have an correct extended automaton by the definition.

We could do another step which can possibly reduce number of states. After generating R we remove unreachable states by some well-known alogrithm.

Non-formal description of each component of automaton state is:

1. This part remembers symbol on the top of derivation tree. Clearly, the symbol to which M should reduce its input.

2. Here are remembered uses of context-sensitive rules from previous column. All these should be eliminated by their use in proper time.

3. This is a rewritten string according to actual column. In the start of the simulation it is equal to an input string. In the end ti should be equal to first component.

4. Here is saved new head of sub-tree for next column. In the next pass it will become the first component.

5. Here are saved all uses of context-sensitive rules. In the next pass it will become the second component.

Automaton will work in such way: It will read input matching one column. This input will be reduced to symbol S. If this is all input, then the computation ends in state f. Otherwise (possibly, because there were used context-sensitive rules), it will read input for next column and continue computation, until all input is read and final state is reached. Automaton has to check context-sensitive rewritings and remember their positions, so they can be completed in next column (same checking has to be done for rewritings of the form $A \to BC$ crossing the borders of columns. As we see further, only case for this, is when C becomes the head of next column derivation sub-tree. It have to be checked by other mechanism (5b). Use of such rule triggers the move to the following column.

Example 3. We have this grammar with vertical restrictions: $G = (V,T,P,S)$

$$V = \{S,A,B,C,D\}$$
$$T = \{a,b,c,d\}P = \{S \to AB, A \to S, B \to S, AB \to CD, C \to c, D \to d\}$$

which generates language $L(G) = \{(cd)^n, n \geq 1\}$

Shortest successful derivation in this grammar looks as $S \to AB \to CD \to cD \to cd$. This derivation can be simulated by automaton obtained from Construction 2 as:

$$\begin{array}{lll} & \langle S,\varepsilon,\varepsilon,\varepsilon,\varepsilon\rangle\, cd & \\ \vdash_M & \langle S,\varepsilon,c,\varepsilon,\varepsilon\rangle\, d & [1] \\ \vdash_M & \langle S,\varepsilon,C,\varepsilon,\varepsilon\rangle\, d & [2] \\ \vdash_M & \langle S,\varepsilon,A,\varepsilon,AB \to CD\rangle\, d & [4a] \\ \vdash_M & \langle B,AB \to CD,\varepsilon,\varepsilon\rangle\, d & [5b] \\ \vdash_M & \langle B,AB \to CD,d,\varepsilon,\varepsilon\rangle & [1] \\ \vdash_M & \langle B,AB \to CD,D,\varepsilon,\varepsilon\rangle & [2] \\ \vdash_M & \langle B,\varepsilon,B,\varepsilon,\varepsilon\rangle & [4b] \\ \vdash_M & f & [6] \end{array}$$

Furthermore, we could see, that $L(G)$ is regular, so it can be described by simpler (and regular) grammar, e. g. $G' = (\{S,A\},\{c,d\},\{S \to Acd, A \to cd, A \to \varepsilon\},S), L(G) = L(M) = L(G')$, without any restrictions.

Corollary 4. *Q accepts every $x \in L(G)$, so $L(G) \subseteq L(Q)$.*

Proof. Any derivation step in G could be expressed as $x \Rightarrow_G y$. Strings x and y could be described as $x = |\alpha_1|\alpha_2|\ldots|\alpha_m|$ and $y = |\alpha_1|\alpha_2|\ldots|\beta_i|\beta_{i+1}|\ldots|\alpha_m|$ (see Table 1).

Without loss of generality we examine only cases of strings $|\alpha_i|\alpha_{i+1}|, |\beta_i|\beta i+1|$, so:

1. $\alpha_{i+1} = \beta_{i+1}$
   First two strings are exactly same, so derivation step occurs in the others two. So there exist derivation $\alpha_i \to \beta_i$. This derivation have to be done by one of following rules:

   (a) $A \to BC \in P$, then $\alpha_i = \gamma_1 A \gamma_2$ and $\beta_i = \gamma_1 BC\gamma_2$. This kind of step is simulated by this transition rule in M: $\langle X, u, \gamma_1 BC\gamma_2, Y, v\rangle \to \langle X, u, \gamma_1 A\gamma_2, Y, v\rangle$ which was created by point 5a in Construction 2.

   (b) $A \to a \in P$. This is the same as preceding case. Only string BC is substituted by a and we use point 2.

2. $\alpha_i = \beta_i$
   All steps from preceding point are repeated. (No matter what string are equal, more important is that changing of sentence is inside of section.)

3. $\alpha_i \neq \beta_i, \alpha_{i+1} \neq \beta_{i+1}$
   Rewriting occurs across borders of sections. It could happen only in two following cases:

   (a) $AB \to CD$. In this case strings look like these: $\alpha_i = \gamma_1 A, \alpha_{i+1} = B\gamma_2$ and $\beta_i = \gamma_1 C, \beta_{i+1} = D\gamma_2$. For this situation we have created transition rules 4a and 4b. First one handle this situation in segment i and second one in the following segment. Cooperation of these rules is handled by fourth (respectively second) member of state.

   (b) $A \to BC$. In this case strings look like these: $\alpha_i = A, \alpha_{i+1} = \gamma_2$ and $\beta_i = B, \beta_{i+1} = C\gamma_2$. For this case we have rule 5b.

In case all input was read and automaton has followed grammar G we have to be in some state of $\langle A, \varepsilon, A, \varepsilon\rangle$. From this state we can transit to state f by point 6 of Construction 2. □

Corollary 5. *G generates all strings accepted by Q, so $L(Q) \subseteq L(G)$.*

Proof. Because of limited space we have to leave this part of proof on reader. □

From Corollaries 4, 5 and equivalency of regular languages and extended finite automatons, we can see, that Claim 1 holds. ■

REFERENCES

[1] Meduna, A.: Automata and Languages: Theory and Applications, Springer, London, 2000