AN INFINITE HIERARCHY OF LANGUAGE FAMILIES RESULTING FROM *N*-LIMITED PROGRAMMED GRAMMARS

Petr Zemek
Bachelor Degree Programme (3), FIT BUT
E-mail: xzemek02@stud.fit.vutbr.cz

Supervised by: Alexander Meduna
E-mail: meduna@fit.vutbr.cz

ABSTRACT

This paper establishes an equivalence between n-limited state grammars and n-limited programmed grammars. This equivalence results into an infinite hierarchy of language families resulting from n-limited programmed grammars, which can be considered in syntactical analysis, when writing a parser based on programmed grammars.

1 INTRODUCTION

There are situations when one needs to parse some noncontext-free language, but does not want to use parsers based on context-sensitive grammars (or even unrestricted grammars), because they are too complex for this purpose. However, standard parsers based on context-free grammars are not enough powerful. In such situations, parsers based on programmed grammars can be considered, but if we restrict derivations to only leftmost, we loose the advantage of generative power of these grammars [3]. The goal of this paper is to show a way how to reduce the effect of this restriction by studying n-limited (n-leftmost) derivations in programmed grammars.

2 PRELIMINARIES AND DEFINITIONS

This paper assumes that the reader is familiar with the formal language theory (see [1]). For a set Q, $|Q|$ denotes the cardinality of Q. For an alphabet V, V^* represents the free monoid generated by V under the operation of concatenation. The identity of V^* is denoted by ε. Set $V^+ = V^* - \{\varepsilon\}$ is thus the free semigroup generated by V under the operation of concatenation.

A *context-free grammar* (see [1]) is a quadruple, $G = (N, T, P, S)$, where N is an alphabet of nonterminals, T is an alphabet of terminals, $V = N \cup T$ be the total alphabet, $S \in N$ and P is a finite set of productions of the form $r\colon A \to x$, where $A \in N$, $x \in V^*$ and r is a label of this rule. If $r\colon A \to x$ is a rule in P, $u = u_1 A u_2$ and $v = u_1 x u_2$, where $u_i \in V^*$, for all $1 \leq i \leq 2$, then $u \Rightarrow v\ [r]$ in G, or, simply, $u \Rightarrow v$. Let $\Rightarrow^n$ denote the n-fold product of $\Rightarrow$, where $n \geq 0$.

Furthermore, let $\Rightarrow^*$ denote the transitive-reflexive closure of $\Rightarrow$. The language of G is defined as $L(G) = \{w\colon S \Rightarrow^* w, w \in T^*\}$. $Lab(p)$ denotes the label of rule p and for set of rules P, $lab(P)$ denotes the set of all labels of rules from P.

A *state grammar* (see [2]) is a sextuple, $G = (N,T,W,P,S,p_0)$, where N, T and S are defined as in a context-free grammar, $V = N \cup T$ be the total alphabet, W is a finite set of states, $p_0 \in W$ is the starting state and $P \subseteq W \times N \times W \times V^+$ is a finite relation. An element $(p,A,q,v) \in P$ is called a state production (abbreviated production or rule) and is usually written as $(p,A) \to (q,v)$. A nonterminal A is said to be applicable under a state p if $(p,A) \to (q,v)$ is in P for some $p,q \in Q$ and $v \in V^+$. Given a state grammar $G = (N,T,W,P,S,p_0)$, let $\Rightarrow$ be a relation on $W \times V^+$ defined as follows: Let p be in W and $w = xAy$ be in V^+. If this A is the leftmost occurence of applicable nonterminal in w under p and $(p,A) \to (q,v)$ is in P, then we write $(p,xAy) \Rightarrow (q,xvy)$. For α and β in $W \times V^+$, write $\alpha \Rightarrow^* \beta$ if either $\alpha = \beta$ or there exists $\alpha_0,\ldots,\alpha_r$ such that $\alpha_0 = \alpha$, $\alpha_r = \beta$, and $\alpha_i \Rightarrow \alpha_{i+1}$ for each i. The sequence $\alpha_0,\ldots,\alpha_r$ is called a derivation (of length r) and is denoted by $\alpha_0 \Rightarrow \cdots \Rightarrow \alpha_r$. The language of G is defined as $L(G) = \{w\colon w \in T^+, (p_0,S) \Rightarrow^* (q,w) \text{ for some } q \in W\}$.

A *programmed grammar* (see [3]) is a quadruple, $G = (N,T,P,S)$, where N, T and S are defined as in a context-free grammar, $V = N \cup T$ be the total alphabet and P is a finite set of rules of the form $(r\colon A \to v, \sigma(r))$, where $r\colon A \to v$, $A \in N$, $v \in V^+$ is a context free rule labeled by r and $\sigma(r)$ is a set of rule labels associated with this rule. After a derivation step, symbolically denoted by $\Rightarrow$, according to a rule of this form in an ordinary context-free way, in the next step a rule labeled by a label from $\sigma(r)$ has to be applied. In the standard manner, we define $\Rightarrow^m$, where $m \geq 0$, $\Rightarrow^+$, and $\Rightarrow^*$. The language of G is defined as $L(G) = \{w\colon S \Rightarrow^* w, w \in T^+\}$.

Let $G = (N_G,T_G,P_G,S_G)$ be a programmed grammar and $H = (N_H,T_H,W_H,P_H,S_H,p_0)$ be a state grammar, respectively, and let n be a positive integer. An *n-limited (n-leftmost) derivation* (see [2]) is a derivation $\alpha_0 \overset{j(1)}{\Rightarrow} \alpha_1 \ldots \overset{j(r)}{\Rightarrow} \alpha_r$ such that $j(i) \leq n$ for each i, where $j(i)$ means that the j-th nonterminal (from the left) is rewritten. In this case we write $\alpha_0 \,{}_n\!\Rightarrow^* \alpha_r$ instead of $\alpha_0 \Rightarrow^* \alpha_r$ in order to indicate that it is realized by an n-limited derivation. The language generated by G this way is defined as $L_{n-lim}(G) = \{w\colon S_G \,{}_n\!\Rightarrow^* w, w \in T_G^+\}$. The language generated by H this way is defined as $L_{n-lim}(H) = \{w\colon w \in T_H^+, (p_0,S_H) \,{}_n\!\Rightarrow^* (q,w) \text{ for some } q \in W_H\}$. By $\mathscr{L}_{n-lim}(SG)$ and $\mathscr{L}_{n-lim}(PG)$ we denote the class of all languages generated by n-limited state and programmed grammars, respectively.

3 RESULTS

Lemma 1. *For every $n \geq 1$, $\mathscr{L}_{n-lim}(SG) \subseteq \mathscr{L}_{n-lim}(PG)$.*

Proof. Let $H = (N,T,W,P,S,p_0)$ be an n-limited state grammar. We construct an n-limited programmed grammar $G = (N \cup \{S_G\},T,P_G,S_G)$, where $S_G \notin N$ is a new nonterminal and P_G is constructed by performing the following steps:

1. Enumerate all rules in P with labels $[i]$, $i \in \{1,2,\ldots,|P|\}$.
2. Add a new rule $(r_0\colon S_G \to S, \{r_i\colon [i]\,(p_0,S) \to (q,v) \in P, q \in W, v \in V^+\})$ to P_G.

3. For $i \in \{1,2,\ldots,|P|\}$:

   Add a new rule to P_G of the form $(r_i\colon A \to v, \sigma(r_i))$, $[i]\,(p,A) \to (q,v) \in P$, $\sigma(r_i) = \{r_j\colon [j]\,(q,B) \to (n,u) \in P\}$, $p,q,n \in W$, $A,B \in N$, $u,v \in V^+$ for some positive integers $j \in \{1,2,\ldots,|P|\}$. □

Lemma 2. *For every $n \geq 1$, $\mathscr{L}_{n-lim}(PG) \subseteq \mathscr{L}_{n-lim}(SG)$.*

Proof. Let $G = (N,T,P,S)$ be an n-limited programmed grammar. We construct an n-limited state grammar $H = (N \cup S_H, T, W, P_H, S_H, \langle \rho \rangle)$, where $S_H \notin N$ is a new nonterminal, ρ is a new symbol and P_H and W are constructed by performing the following steps:

1. For each $(r\colon S \to v, \sigma(r)) \in P$, $r \in lab(P)$, $v \in V^+$, $\sigma(r) \subseteq lab(P)$, add $(\langle \rho \rangle, S_H) \to (\langle r \rangle, S)$ to P_H, where $\langle \rho \rangle$ is a new state in W.
2. For each $(r\colon A \to v, \sigma(r)) \in P$, $r \in lab(P)$, $A \in N$, $v \in V^+$, $\sigma(r) \subseteq lab(P)$:

   If $\sigma(r) = \emptyset$, then add $(\langle r \rangle, A) \to (\langle \phi \rangle, v)$ to P_H, where $\langle r \rangle$ and $\langle \phi \rangle$ are new states in W. Otherwise, for each $s \in \sigma(r)$ add $(\langle r \rangle, A) \to (\langle s \rangle, v)$ to P_H, where $\langle r \rangle$ and $\langle s \rangle$ are new states in W. □

Theorem 1. *For every $n \geq 1$, $\mathscr{L}_{n-lim}(PG) = \mathscr{L}_{n-lim}(SG)$.*

Proof. The result directly follows from Lemmas 1 and 2. □

Theorem 2. *For every $n \geq 1$, $\mathscr{L}_{n-lim}(PG) \subset \mathscr{L}_{(n+1)-lim}(PG)$.*

Proof. Recall that $\mathscr{L}_{n-lim}(SG) \subset \mathscr{L}_{(n+1)-lim}(SG)$ (see [2]) and $\mathscr{L}_{n-lim}(PG) = \mathscr{L}_{n-lim}(SG)$ (see Theorem 1) for every $n \geq 1$. Thus, Theorem 2 holds. □

4 CONCLUSION

In this paper an equivalence between n-limited state grammars and n-limited programmed grammars was established. Rigorous proofs were omitted due to limited space, so only constructions were given. This equivalence results into an infinite hierarchy of language families resulting from n-limited programmed grammars. One can use this result when writing a parser for some noncontext-free language based on programmed grammars which uses only n-leftmost derivations, so it can be more effecient than using universal (unrestricted) derivations.

REFERENCES

[1] Alexander Meduna, Automata and Languages: Theory and Applications, Springer, London, 2000. ISBN 1-85233-074-0.

[2] Takumi Kasai, An hierarchy between context-free and context-sensitive languages, Journal of Computer and System Sciences, 1970.

[3] Jürgen Dassow and Gheorghe Păun, Regulated Rewriting in Formal Language Theory, Springer, New York, 1989, ISBN 38751-414-7.