On State Grammars With Final States

Petr Zemek, izemek@fit.vutbr.cz

November 4, 2010

Abstract

A state grammar is a combination of a context-free grammar and a finite automaton. However, unlike finite automata, state grammars have no final states. The present note shows that by introducing final states to state grammars we do not affect their generative power.

1 Introduction

A state grammar, as is was introduced by Kasai in [2], is a combination of a context-free grammar and a finite automaton. In a single derivation step, a nonterminal is rewritten to a string and the current state is changed. Moreover, the leftmost nonterminal which can be rewritten has to be rewritten. Also, recall that Kasai did not consider the presence of erasing rules—that is, rules with the empty string on their right hand sides. This was considered later by Horváth and Meduna in [1].

In finite automata, the input string is accepted if and only if the automaton reads the whole input and ends in a final state. However, in the original definition of a state grammar, no final states are present. Indeed, a string of terminals is generated by a state grammar no matter what state the grammar ends up in. Therefore, it is a natural question what happens if we include final states to the definition of a state grammar. The present paper shows that by introducing final states to state grammars we do not affect their generative power.

2 Preliminaries and Definitions

This paper assumes that the reader is familiar with the theory of formal languages (see [4, 3]). For an alphabet, V, V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. For a word, $w \in V^*$, $alph(w)$ denotes the set of symbols occurring in w.

A *state grammar* (see [1, 2]) is a sextuple, $G = (N, T, Q, P, S, s)$, where N and T are two disjoint alphabets, Q is a finite non-empty set, $P \subseteq N \times Q \times V^* \times Q$ is finite, $S \in N$, and $s \in Q$. Set $V = N \cup T$. V, N, T, Q, P, S, and s are called the *total alphabet*, the alphabet of *terminals*, the alphabet of *nonterminals*, the set of *states*, the set of *rules*, the *start nonterminal*, and the *start state*, respectively. Instead of $(A, p, x, q) \in P$, we write $(A, p) \rightarrow (x, q) \in P$ throughout the rest of the paper. If every $(A, p) \rightarrow (x, q) \in P$ implies $x \neq \varepsilon$, then G is *ε-free*. The relation of a *direct derivation*, symbolically denoted by $\Rightarrow$, is defined as follows: if $(A, p) \rightarrow (x, q) \in P$, $u = (rAs, p)$, $v = (rxs, q)$, where $r, s \in V^*$, and for every $(B, p) \rightarrow (y, t) \in P$, $B \notin alph(r)$, then $u \Rightarrow v$ in G. The *language generated by* G, denoted by $L(G)$, is defined as $L(G) = \{w \in T^* \mid (S, s) \Rightarrow^* (w, q)$, for some $q \in Q\}$, where $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$.

A *state grammar with final states* is a septuple, $H = (N, T, Q, P, S, s, F)$, where N, T, Q, P, S, s, $\Rightarrow$, and $\Rightarrow^*$ are defined as in state grammars and $F \subseteq Q$ is the set of *final states*. By analogy with ε-free state grammars, we define *ε-free state grammars with final states*. The *language generated by* H, denoted by $L(H)$, is defined as $L(H) = \{w \in T^* \mid (S, s) \Rightarrow^* (w, f)$, for some $f \in F\}$.

By **CS** and **RE**, we denote the families of context-sensitive and recursively enumerable languages, respectively. $\mathbf{ST}^{\varepsilon}$ and $\mathbf{ST}$ denote the families of languages generated by state grammars and by ε-free state grammars, respectively. $\mathbf{ST}_F^{\varepsilon}$ and $\mathbf{ST}_F$ denote the families of languages generated by state grammars with final states and by ε-free state grammars with final states, respectively.

3 Main Result

Recall that $\mathbf{ST}^{\varepsilon} = \mathbf{RE}$ (see Theorem 1 in [1]) and $\mathbf{ST} = \mathbf{CS}$ (see Theorem 2 in [2]). We show that also $\mathbf{ST}_F^{\varepsilon} = \mathbf{RE}$ and $\mathbf{ST}_F = \mathbf{CS}$.

Lemma 1. $\boldsymbol{ST}^{\varepsilon} \subseteq \boldsymbol{ST}_F^{\varepsilon}$ *and* $\boldsymbol{ST} \subseteq \boldsymbol{ST}_F$

Proof. Let $G = (N, T, Q, P, S, s)$ be any state grammar. The state grammar $H = (N, T, Q, P, S, s, Q)$ clearly satisfies $L(H) = L(G)$ and if G is ε-free, then so is H. □

Lemma 2. $\boldsymbol{ST}_F^{\varepsilon} \subseteq \boldsymbol{RE}$

Proof. This inclusion can be obtained by standard simulations by a Turing machine. □

Lemma 3. $\boldsymbol{ST}_F \subseteq \boldsymbol{CS}$

Proof. As ε-free state grammars cannot shorten their sentential forms, their derivations can be simulated by linear bounded automata. Hence, the lemma holds. □

Theorem 1. *$\boldsymbol{ST}^{\varepsilon} = \boldsymbol{ST}_F^{\varepsilon} = \boldsymbol{RE}$ and $\boldsymbol{ST} = \boldsymbol{ST}_F = \boldsymbol{CS}$*

Proof. $\mathbf{ST}^{\varepsilon} = \mathbf{ST}_F^{\varepsilon} = \mathbf{RE}$ follows from Lemma 1, Lemma 2, and from the fact that $\mathbf{ST}^{\varepsilon} = \mathbf{RE}$. $\mathbf{ST} = \mathbf{ST}_F = \mathbf{CS}$ follows from Lemma 1, Lemma 3, and from the fact that $\mathbf{ST} = \mathbf{CS}$. □

References

[1] G. Horváth and A. Meduna. On state grammars. *Acta Cybernetica*, 1988(8):237–245, 1988.

[2] T. Kasai. An hierarchy between context-free and context-sensitive languages. *Journal of Computer and System Sciences*, 4:492–508, 1970.

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

[4] A. Salomaa. *Formal Languages.* Academic Press, 1973.