15-453 Formal Languages, Automata, and Computation

Some Notes on Regular Grammars

Frank Pfenning

Lecture 9
February 7, 2000

In this notes we describe restrictions to context-free grammars which ensure that the generated languages are regular. We first discuss *strictly right-linear grammars* and show that they correspond directly to non-deterministic finite automata (NFAs). We then generalize to *right-linear grammars* which does not change the set of accepted languages, but allows more concise specifications. The dual *left-linear grammars* also generate the same languages. A grammar is *regular* if it is either right-linear or left-linear.

1 Strictly Right-Linear Grammars

We first review the definition of a *context-free grammar.*

Definition 1 (Context-Free Grammar)
A context-free grammar *G is specified by a tuple (V, Σ, R, S) where*

V	*is a finite set of* variables *(also called* non-terminals*),*
Σ	*is a finite alphabet,*
$R \subseteq V \times (V \cup \Sigma)^*$	*is the set of* rules,
$S \in V$	*is the* start variable.

We write rules (A, w) as $A \to w$. A derivation step *has the form $uAv \Rightarrow uwv$ where $A \in V$ and $A \to w \in R$. A* derivation *has the form*

$$u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_n$$

which we abbreviate $u_1 \stackrel{*}{\Longrightarrow} u_n$. Finally, we define the language generated by G

$$\mathcal{L}(G) = \{w \in \Sigma^* \mid S \stackrel{*}{\Longrightarrow} w\}.$$

For a strictly right-linear grammar, we severely restrict the form of the rules.

Definition 2 (Strictly Right-Linear Grammar) *A* strictly right-linear grammar *is a context-free grammar G where each rule has one of the following forms:*

$$A \to xB \quad \text{for } x \in \Sigma_\epsilon$$
$$A \to \epsilon$$

Any derivation of a word w from a start symbol S in a strictly right-linear grammar has the form

$$S \Rightarrow x_1 A_1 \Rightarrow x_1 x_2 A_2 \Rightarrow \cdots \Rightarrow x_1 \ldots x_n A_n \Rightarrow x_1 \ldots x_n$$

where $w = x_1 \ldots x_n$. Note that some of the x_i may be ϵ. The form of these derivations suggests the connection to non-deterministic finite automata.

2 Strictly Right-Linear Grammars Generate Regular Languages

Lemma 1 *If G is a strictly right-linear grammar, then $\mathcal{L}(G)$ is regular.*

Proof: Given a strictly right-linear grammar $G = (V, \Sigma, R, S)$, we construct a non-deterministic finite automaton $N = (V, \Sigma, \delta, S, F)$ such that $\mathcal{L}(G) = \mathcal{L}(N)$. The set of states is simply the set of non-terminals of G, with the start state corresponding to the start variable. We further define

$$\begin{array}{lcl} \delta(A, x) & = & \{B \mid A \to xB \in R\}, \\ F & = & \{C \mid C \to \epsilon \in R\}. \end{array}$$

It remains to show that $\mathcal{L}(G) = \mathcal{L}(N)$. As usual, we do this in two parts, $\mathcal{L}(G) \subseteq \mathcal{L}(N)$ and $\mathcal{L}(N) \subseteq \mathcal{L}(G)$. We write $A \xrightarrow{w}_N B$ for computations of the NFA N.

1. $\mathcal{L}(G) \subseteq \mathcal{L}(N)$. We show by induction on the structure of the derivation:

   If $A \xrightarrow{*} w$ then either
   (a) $w = uB$ and $A \xrightarrow{u}_N B$ for some $u \in \Sigma^*$, or
   (b) $w = u$ and $A \xrightarrow{u}_N C$ for some $u \in \Sigma^*$ and final state C.

   Base Case: $A \xrightarrow{*} A$. Then $A \xrightarrow{\epsilon}_N A$ by definition of $\Longrightarrow_N$.

   Induction Step, Case 1: $A \Rightarrow xA_1 \xrightarrow{*} xw_1$ for $x \in \Sigma_\epsilon$ and $w = xw_1$. By induction hypothesis on $A_1 \xrightarrow{*} w_1$, we have two subcases to consider:
   (a) $A_1 \xrightarrow{*} vB$ and $A_1 \xrightarrow{v}_N B$. Then $u = xv$ satisfies the claim, since $A \Rightarrow xvB$ and $A \xrightarrow{x}_N A_1 \xrightarrow{v}_N B$.
   (b) $A_1 \xrightarrow{*} v$ and $A_1 \xrightarrow{u}_N C$ for some final state C. Then $u = xv$ satisfies the claim, since $A \Rightarrow xv$ and $A \xrightarrow{x}_N A_1 \xrightarrow{v}_N C$ where C is a final state.

   Induction Step, Case 2: $A \Rightarrow \epsilon$. Then $A \in F$ and $A \xrightarrow{\epsilon}_N A$.

2. $\mathcal{L}(N) \subseteq \mathcal{L}(G)$. We show by induction on the structure of the computation:

   If $A \xrightarrow{w}_N B$ then $A \xrightarrow{*} wB$.

   From this the claim follows by using S for A and a final state C for B.

   Base Case: $A \xrightarrow{\epsilon}_N A$. Then $A \xrightarrow{*} A$ by definition of $\xrightarrow{*}$.

   Induction Step: $A \xrightarrow{x}_N A_1 \xrightarrow{u}_N B$ where $w = xu$ and $x \in \Sigma_\epsilon$. By induction hypothesis, $A_1 \xrightarrow{*} uB$ so $A \Rightarrow xA_1 \xrightarrow{*} xuB$.

□

Lemma 2 *If L is a regular language, then there is a strictly right-linear grammar G such that $\mathcal{L}(G) = L$.*

Proof: If L is regular, then there is an NFA $N = (Q, \Sigma, \delta, q_0, F)$ recognizing L. We construct a strictly right-linear grammar $G = (Q, \Sigma, R, q_0)$ where

$$R = \{q \to xr \mid r \in \delta(q, x)\} \cup \{q \to \epsilon \mid q \in F\}.$$

We leave the proof that $\mathcal{L}(N) = \mathcal{L}(G)$ to the reader. It follows by straightforward inductions in both directions. $\square$

Theorem 1 *A language L is regular if and only if there is a strictly right-linear grammar G such that $\mathcal{L}(G) = L$.*

Proof: By the two preceding lemmas. $\square$

3 Right-Linear Grammars

Right-linear grammars allow more general right-hand sides for rules, but they can easily be translated to strictly right-linear grammars.

Definition 3 (Right-Linear Grammar)
A right-linear grammar *is a context-free grammar $G = (V, \Sigma, R, S)$ where each rule in R has one of the following forms:*

$$\begin{array}{ll} A \to wB & \textit{for } w \in \Sigma_\epsilon^* \\ A \to w & \textit{for } w \in \Sigma_\epsilon^* \end{array}$$

Clearly, any strictly right-linear grammar is right-linear. In addition, strictly right-linear grammars can easily simulate right-linear grammars.

Lemma 3 *For any right-linear grammar G there exists a strictly right-linear grammar H such that $\mathcal{L}(G) = \mathcal{L}(H)$.*

Proof: Let $G = (V, \Sigma, R, S)$ be a right-linear grammar. We construct a strictly right-linear grammar by transforming each rule

$$A \to a_1 \ldots a_n B$$

where $n \geq 2$ into the set of rules

$$\begin{array}{lcl} A & \to & a_1 A_1 \\ A_1 & \to & a_2 A_2 \\ & \ldots & \\ A_n & \to & B \end{array}$$

where $A_1, \ldots, A_n$ are new non-terminals. Similarly, each rule

$$A \to a_1 \ldots a_n$$

for $n \geq 1$ is translated into

$$\begin{array}{lcl} A & \to & a_1 A_1 \\ A_1 & \to & a_2 A_2 \\ & \ldots & \\ A_n & \to & \epsilon \end{array}$$

where $A_1, \ldots, A_n$ are new non-terminals.

Then $H = (V', \Sigma, R', S)$, where V' contains V and all new non-terminals introduced in the transformation above, and R' contains the rules of R satisfying the strict right-linearity condition and all rules resulting from the above transformation.

It is now easy to see that $\mathcal{L}(G) = \mathcal{L}(H)$. We leave the details of an inductive proof to the reader. $\square$

Theorem 2 *A language L is regular if and only if there is a right-linear grammar G with $\mathcal{L}(G) = L$.*

Proof: By the preceding lemma and Theorem 1. $\square$

4 Regular Grammars

Regular grammars are grammars that are either right-linear or left-linear as defined below.

Definition 4 (Left-Linear Grammars)
A left-linear grammar *is a context-free grammar $G = (V, \Sigma, R, S)$ where each rule in R has one of the following forms:*

$$\begin{array}{ll} A \rightarrow Bw & \textit{for } w \in \Sigma_\epsilon^* \\ A \rightarrow w & \textit{for } w \in \Sigma_\epsilon^* \end{array}$$

It is easy to show that left-linear grammars also define regular languages. This leads to the theorem that a language L is regular if and only if it is generated by a regular grammar.

Note, however, that if we are allowed to mix left-linear and right-linear rules in a single grammar, the result will not necessarily generate a regular language. For example, a grammar with the rules

$$\begin{array}{lcl} S & \rightarrow & 0A \\ S & \rightarrow & 1B \\ S & \rightarrow & \epsilon \\ A & \rightarrow & S0 \\ B & \rightarrow & S1 \end{array}$$

generates the language $L = \{ww^R \mid w \in \{0,1\}^*\}$ which is not regular.