Finite Automata Theory and Formal Languages

TMV027/DIT321– LP4 2018

Lecture 13

Ana Bove

May 7th 2018

Recap: Context-Free Grammars

- Equivalence between recursive inference, (leftmost/rightmost) derivations and parse trees;
- Ambiguous grammars;
- Inherent ambiguity;
- Proofs about grammars and languages.

Overview of Today's Lecture

- Simplification of CFL;
- Chomsky normal form for CFL.

Contributes to the following learning outcome:

- Explain and manipulate the diff. concepts in automata theory and formal lang;
- Simplify automata and context-free grammars;
- Differentiate and manipulate formal descriptions of lang, automata and grammars.

And guest lecture by *Martin Fabian* on *Application of Formal Verification to the Lane Change Module of an Autonomous Vehicle*.

Generating, Reachable, Useful and Useless Symbols

Let $G = (V, T, \mathcal{R}, S)$ be a CFG.
Let $X \in V \cup T$ and let $\alpha, \beta \in (V \cup T)^*$.

Definition: X is *reachable* if $S \Rightarrow^* \alpha X \beta$.
(This is similar to accessible states in FA.)

Definition: X is *generating* if $X \Rightarrow^* w$ for some $w \in T^*$.

Definition: The symbol X is *useful* if $S \Rightarrow^* \alpha X \beta \Rightarrow^* w$ for some $w \in T^*$.
Note: A symbol that is useful should be generating and reachable.

Definition: X is *useless* iff it is not useful.

We shall simplify the grammars by eliminating useless symbols.

Computing the Generating Symbols

Let $G = (V, T, \mathcal{R}, S)$ be a CFG.

The following recursive procedure computes the generating symbols of G:

Base Case: All elements of T are generating;

Recursive Step: If a production $A \to \alpha$ is such that all symbols of α are known to be generating, then A is also generating.
Observe that α could be ϵ.

The recursive step must be applied until no new symbols are found generating.

Theorem: *The procedure above finds all and only the generating symbols of a grammar.*

Proof: See Theorem 7.4 in the book.

Example: Generating Symbols

Consider the grammar over $\{a\}$ given by the rules:

$$\begin{array}{lcl} S & \to & aS \mid W \mid U \\ W & \to & aW \\ U & \to & a \\ V & \to & aa \end{array}$$

a is generating.

U and V are generating since $U \to a$ and $V \to aa$.

S is generating since $S \to U$.

No other symbol is found generating so W is not generating.

After eliminating the non-generating symbols and their productions we get

$$S \to aS \mid U \qquad U \to a \qquad V \to aa$$

Computing the Reachable Symbols

Let $G = (V, T, \mathcal{R}, S)$ be a CFG.

The following recursive procedure computes the reachable symbols of G:

Base Case: The start variable S is reachable;

Recursive Step: If A is reachable and we have a production $A \to \alpha$ then all symbols in α are reachable.

The recursive step must be applied until no new symbols are found reachable.

Theorem: *The procedure above finds all and only the reachable symbols of a grammar.*

Proof: See Theorem 7.6 in the book.

Example: Reachable Symbols

Consider the grammar given by the rules:

$$\begin{array}{ll} S \to aB \mid BC & C \to b \\ A \to aA \mid c \mid aDb & D \to B \\ B \to DB \mid C & \end{array}$$

S is reachable.

Hence a, B and C are reachable.

Then b and D are reachable.

No other symbol are found reachable so A and c are not reachable.

After eliminating the non-reachable symbols and their productions we get

$$\begin{array}{ll} S \to aB \mid BC & C \to b \\ B \to DB \mid C & D \to B \end{array}$$

Eliminating Useless Symbols

It is important in which order we check generating and reachable symbols!

Example: Consider the following grammar

$$S \to AB \mid a \qquad\qquad A \to b$$

If we first check for generating symbols and then for reachability we get

$$S \to a$$

If we first check for reachability and then for generating we get

$$S \to a \qquad\qquad A \to b$$

Eliminating Useless Symbols

Theorem: *Let $G = (V, T, \mathcal{R}, S)$ be a CFG and let $\mathcal{L}(G) \neq \emptyset$.*
Let $G' = (V', T', \mathcal{R}', S)$ be constructed as follows:

1. *First, eliminate all non-generating symbols and all productions involving one or more of those symbols;*
2. *Then, eliminate all non-reachable symbols and all productions involving one or more of those symbols.*

Then G' has no useless symbols and $\mathcal{L}(G) = \mathcal{L}(G')$.

Proof: See Theorem 7.2 in the book.

Example: Eliminating Useless Symbols

Consider the grammar given by the rules:

$$\begin{array}{lllllll} S & \to & gAe \mid aYB \mid CY & \quad & A & \to & bBY \mid ooC \\ B & \to & dd \mid D & & C & \to & jVB \mid gl \\ D & \to & n & & U & \to & kW \\ V & \to & baXXX \mid oV & & W & \to & c \\ X & \to & fV & & Y & \to & Yhm \end{array}$$

After eliminating non-generating symbols:

$$\begin{array}{lllllll} S & \to & gAe & \quad & A & \to & ooC \\ B & \to & dd \mid D & & C & \to & gl \\ D & \to & n & & U & \to & kW \\ & & & & W & \to & c \end{array}$$

After eliminating non-reachable symbols:

$$S \to gAe \qquad A \to ooC \qquad C \to gl$$

What is the language generated by the grammar?

Nullable Variables

Definition: A variable A is *nullable* if $A \Rightarrow^* \epsilon$.

Note: Observe that only variables are nullable!

Let $G = (V, T, \mathcal{R}, S)$ be a CFG.

The following recursive procedure computes the nullable variables of G:

Base Case: If $A \to \epsilon$ is a production then A is nullable;

Recursive Step: If $B \to X_1X_2 \ldots X_k$ is a production and all the X_i are nullable then B is also nullable.

The recursive step must be applied until no new symbols are found nullable.

Theorem: *The procedure above finds all and only the nullable variables of a grammar.*

Proof: See Theorem 7.7 in the book.

Eliminating ϵ-Productions

Definition: An *ϵ-production* is a production of the form $A \rightarrow \epsilon$.

Let $G = (V, T, \mathcal{R}, S)$ be a CFG.

The following procedure eliminates the ϵ-production of G:

1. Determine all nullable variables of G;
2. Build $\mathcal{P}$ with all the productions of $\mathcal{R}$ plus a rule $A \rightarrow \alpha\beta$ whenever we have $A \rightarrow \alpha B \beta$ and B is nullable.
   Note: If $A \rightarrow X_1 X_2 \ldots X_k$ and all X_i are nullable, we do not include the case where all the X_i are absent;
3. Construct $G' = (V, T, \mathcal{R}', S)$ where $\mathcal{R}'$ contains all the productions in $\mathcal{P}$ except for the ϵ-productions.

Theorem: *The grammar G' constructed from the grammar G as above is such that $\mathcal{L}(G') = \mathcal{L}(G) - \{\epsilon\}$.*

Proof: See Theorem 7.9 in the book.

Example: Eliminating ϵ-Productions

Example: Consider the grammar given by the rules:

$$S \rightarrow aSb \mid SS \mid \epsilon$$

By eliminating ϵ-productions we obtain

$$S \rightarrow ab \mid aSb \mid S \mid SS$$

Example: Consider the grammar given by the rules:

$$S \rightarrow AB \qquad A \rightarrow aAA \mid \epsilon \qquad B \rightarrow bBB \mid \epsilon$$

By eliminating ϵ-productions we obtain

$$S \rightarrow A \mid B \mid AB \qquad A \rightarrow a \mid aA \mid aAA \qquad B \rightarrow b \mid bB \mid bBB$$

Eliminating Unit Productions

Definition: A *unit production* is a production of the form $A \rightarrow B$.
(This is similar to ϵ-transitions in a ϵ-NFA.)

Let $G = (V, T, \mathcal{R}, S)$ be a CFG.

The following procedure eliminates the unit production of G:

1. Build $\mathcal{P}$ with all the productions of $\mathcal{R}$ plus a rule $A \rightarrow \alpha$ whenever we have $A \rightarrow B$ and $B \rightarrow \alpha$;
   Observe that this step might introduce new unit productions that must be expanded!
2. Construct $G' = (V, T, \mathcal{R}', S)$ where $\mathcal{R}'$ contains all the productions in $\mathcal{P}$ except for the unit production.

Theorem: *The grammar G' constructed from the grammar G as above is such that $\mathcal{L}(G') = \mathcal{L}(G)$.*

Proof: See Theorem 7.13 in the book.

Example: Eliminating Unit Productions

Consider the grammar given by the rules:

$$\begin{array}{lll} S & \rightarrow & CBh \mid D \\ B & \rightarrow & Sf \mid ggg \\ D & \rightarrow & E \mid SABC \end{array} \qquad \begin{array}{lll} A & \rightarrow & aaC \\ C & \rightarrow & cA \mid d \mid C \\ E & \rightarrow & be \end{array}$$

By eliminating unit productions we obtain:

$$\begin{array}{lll} S & \rightarrow & CBh \mid be \mid SABC \\ B & \rightarrow & Sf \mid ggg \\ D & \rightarrow & be \mid SABC \end{array} \qquad \begin{array}{lll} A & \rightarrow & aaC \\ C & \rightarrow & cA \mid d \\ E & \rightarrow & be \end{array}$$

Simplification of a Grammar

Theorem: *Let $G = (V, T, \mathcal{R}, S)$ be a CFG whose language contains at least one string other than ϵ. If we construct G' by*

1. *First, eliminating ϵ-productions;*
2. *Then, eliminating unit productions;*
3. *Finally, eliminating useless symbols;*

using the procedures shown before then $\mathcal{L}(G') = \mathcal{L}(G) - \{\epsilon\}$.

In addition, G' contains no ϵ-productions, no unit productions and no useless symbols.

Proof: See Theorem 7.14 in the book.

Note: It is important to apply the steps in this order!

Chomsky Normal Form

Definition: A CFG is in *Chomsky Normal Form* (CNF) if G has no useless symbols and all the productions are of the form $A \to BC$ or $A \to a$.

Note: Observe that a CFG that is in CNF has no unit or ϵ-productions!

Theorem: *For any CFG G whose language contains at least one string other than ϵ, there is a CFG G' that is in Chomsky Normal Form and such that $\mathcal{L}(G') = \mathcal{L}(G) - \{\epsilon\}$.*

Proof: See Theorem 7.16 in the book.

Constructing a Chomsky Normal Form

Let us assume G has no ϵ- or unit productions and no useless symbols.

Then every production is of the form $A \rightarrow a$ or $A \rightarrow X_1 X_2 \ldots X_k$ for $k > 1$.

If X_i is a terminal introduce a new variable A_i and a new rule $A_i \rightarrow X_i$
(if no such rule exists for X_i with a variable that has no other rules).

Use A_i in place of X_i in any rule whose body has length > 1.

Now, all rules are of the form $B \rightarrow b$ or $B \rightarrow C_1 C_2 \ldots C_k$ with all C_j variables.

Introduce $k - 2$ new variables and break each rule $B \rightarrow C_1 C_2 \ldots C_k$ as

$$B \rightarrow C_1 D_1 \quad D_1 \rightarrow C_2 D_2 \quad \cdots \quad D_{k-2} \rightarrow C_{k-1} C_k$$

Example: Chomsky Normal Form

Example: Consider the grammar given by the rules:

$$S \rightarrow aSb \mid SS \mid ab$$

We first obtain

$$S \rightarrow ASB \mid SS \mid AB \qquad A \rightarrow a \qquad B \rightarrow b$$

Then we build a grammar in Chomsky Normal Form

$$\begin{array}{lll} S & \rightarrow & AC \mid SS \mid AB \\ C & \rightarrow & SB \end{array} \quad \begin{array}{lll} A & \rightarrow & a \\ B & \rightarrow & b \end{array}$$

Example: Observe however that

$$S \rightarrow aa \mid a$$

is NOT equivalent to

$$S \rightarrow SS \mid a$$

Instead we need to build

$$S \rightarrow AA \mid a \qquad A \rightarrow a$$

Overview of Next Lecture

Sections 7.2–7.4, and notes on *Pumping lemma*:

- Regular grammars;
- Chomsky hierarchy;
- Pumping lemma for CFL;
- Closure properties of CFL;
- Decision properties of CFL.

Overview of next Week

Mon 14	Tue 15	Wed 16	Thu 17	Fri 18
	10-12 EA Exercise		**10-12 ES61 Individual help**	
Lec 13-15 HB3 CFL.			**Lec 13-15 HB3** PDA. TM.	
15-17 EA Exercise		**15-17 EL41 Consultation**		

Assignment 6: CFL.
Deadline: Sunday May 20th 23:59.