COMS W3261
Computer Science Theory
Lecture 9
Equivalence of CFG's and PDA's

Alexander Roth

2014 – 09 – 29

Outline

- From a CFG to a PDA.
- From a PDA to a CFG
- Eliminating useless symbols from a CFG
- Eliminating ϵ-productions
- Eliminating unit productions
- Chomsky normal form

1 From a CFG to an equivalent PDA

- Given a CFG G, we can construct a PDA P such that $N(P) = L(G)$.
- The PDA will simulate leftmost derivations of G.
- Algorithm to construct a PDA for a CFG
  - Input: a CFG $G = (V, T, Q, S)$.
  - Output: a PDA P such that $N(P) = L(G)$.
  - Method: Let $P = (\{q\}, T, V \cup T, \delta, q, S)$ where
    1. $\delta(q, \epsilon, A) = \{(q, \beta) | A \to \beta \text{ is in } Q\}$ for each nonterminal A in V.
    2. $\delta(q, a, a) = \{(q, \epsilon)\}$ for each terminal a in T.
- For a given input string w, the PDA simulates a leftmost derivation for w in G.

- We can prove that $N(P) = L(G)$ by showing that w is in $N(P)$ iff w is in $L(G)$:
  - If part: If w is in $L(G)$, then there is a leftmost derivation

    $$S = Y_1 \Rightarrow Y_2 \Rightarrow \cdots \Rightarrow Y_n = w$$

    We show by induction on i that P simulates this leftmost derivation by the sequence of moves

    $$(q, w, S) \overset{*}{\vdash} (q, y_i, \alpha_i)$$

    such that if $\gamma_i = x_i\alpha_i$, then $x_i y_i = w$.
  - Only-if part: If $(q, x, A) \overset{*}{\vdash} (q, \epsilon, \epsilon)$, then $A \overset{*}{\Rightarrow} x$.
    We can prove this statement by induction on the number of moves made by P.

2 From a PDA to an equivalent CFG

- Given a PDA P, we can construct a CFG G such that $L(G) = N(P)$.
- The basic idea of the proof is to generate the strings that cause P to go from state q to state p, popping a symbol X off the stack, by a nonterminal of the form $[qXp]$.
- Algorithm to construct a CFG for a PDA
  - Input: a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$.
  - Output: a CFG $G = (V, \Sigma, R, S)$ such that $L(G) = N(P)$.
  - Method:
    1. Let the nonterminal S be the start symbol of G. The other nonterminals in V will be symbols of the form $[pXq]$ where p and q are states in Q, and X is a stack symbol in Γ.
    2. The set of productions R is constructed as follows:
       * For all states p, R has the production $S \to [q_0Z_0p]$.
       * If $\delta(q, a, X)$ contains $(r, Y_1Y_2 \ldots Y_k)$, then R has the productions

         $$[qXr_k] \to a[rY_1r_1][r_1Y_2r_2] \ldots [r_{k-1}Y_kr_k]$$

         for all lists of states $r_1, r_2, \ldots, r_k$.
  - We can prove that $[qXp] \overset{*}{\Rightarrow} w$ iff $(q, w, X) \overset{*}{\vdash} (p, \epsilon, \epsilon)$.
  - From this, we have $[q_0Z_0p] \overset{*}{\Rightarrow} w$ iff $(q_0, w, Z_0) \overset{*}{\vdash} (p, \epsilon, \epsilon)$, so we can conclude $L(G) = N(P)$.

3 Eliminating Useless Symbols from a CFG

- A symbol X is *useful* for a CFG if there is a derivation of the form $S \stackrel{*}{\Rightarrow} \alpha X \beta \stackrel{*}{\Rightarrow} w$ for some strings of terminals w.
- If X is not useful, then we say X is *useless*.
- To be useful, a symbol X needs to be
  1. *generating*; that is, X needs to be able to drive some string of terminals.
  2. *reachable*; that is, there needs to be a derivation of the form $S \stackrel{*}{\Rightarrow} \alpha X \beta$ where α and β are strings of nonterminals and terminals.
- To eliminate useless symbols from a grammar, we
  1. identify the non-generating symbols and eliminate all productions containing one or more of these symbols, and then
  2. eliminate all productions containing symbols that are not reachable from the start symbol.
- In the grammar

$$\begin{aligned} S &\rightarrow AB \quad | \quad a \\ A &\rightarrow b \end{aligned}$$

  S, A, a, and b are generating. B is not generating.
  Eliminating the productions containing the non-generating symbols, we get

$$\begin{aligned} S &\rightarrow a \\ A &\rightarrow b \end{aligned}$$

  Now we see A is not reachable from S, so we can eliminate the second production to get

$$S \rightarrow a$$

- The generating symbols can be computed inductively bottom-up from the set of terminal symbols.
- The reachable symbols can be computed inductively starting from S.

4 Eliminating ϵ-productions from a CFG

- If a language L has a CFG, then $L - \{\epsilon\}$ has a CFG without any ϵ-productions.
- A nonterminal A in a grammar is *nullable* if $A \stackrel{*}{\Rightarrow} \epsilon$.

- The nullable nonterminals can be determined iteratively.
- We can eliminate all ϵ-productions in a grammar as follows:
  - Eliminate all productions with ϵ bodies.
  - Suppose $A \to X_1 X_2 \ldots X_k$ is a production and m of the kX_i's are nullable. Then add the 2^m versions of this production where the nullable X_i's are present or absent. (But if all symbols are nullable, do not add an ϵ-production.)
- Let us eliminate the ϵ-productions from the grammar G

$$\begin{aligned} S &\to AB \\ A &\to aAA \mid \epsilon \\ B &\to bBB \mid \epsilon \end{aligned}$$

S, A, and B are nullable.
For the production $S \to AB$ we add the productions $S \to A \mid B$
For the production $A \to aAA$ we add the productions $A \to aA \mid a$
For the production $B \to bBB$ we add the productions $B \to bB \mid b$
The resulting grammar H with no ϵ-production is

$$\begin{aligned} S &\to AB \mid A \mid B \\ A &\to aAA \mid aA \mid a \\ B &\to bBB \mid bB \mid b \end{aligned}$$

We can prove that $L(H) = L(G) - \{\epsilon\}$.

5 Eliminating Unit Productions from a CFG

- A *unit* production is one of the form $A \to B$ where both A and B are nonterminals.
- Let us assume we are given a grammar G with no ϵ-productions.
- From G we can create an equivalent grammar H with no unit productions as follows.
  - Define (A, B) to be a unit pair if $A \overset{*}{\Rightarrow} B$ in G.
  - We can inductively construct all unit pairs for G.
  - For each unit pair (A, B) in G, we add to H the productions $A \to \alpha$ where $B \to \alpha$ is a nonunit production of G.

- Consider the standard grammar G for arithmetic expressions:

$$
\begin{aligned}
E &\to E + T \mid T \\
T &\to T * F \mid F \\
F &\to (E) \mid a
\end{aligned}
$$

The unit pairs are (E, E), (E, T), (E, F), (T, T), (T, F), (F, F).
The equivalent grammar H with no unit productions is:

$$
\begin{aligned}
E &\to E + T \mid T * F \mid (E) \mid a \\
T &\to T * F \mid (E) \mid a \\
F &\to (E) \mid a
\end{aligned}
$$

6 Putting a CFG into Chomsky Normal Form

- A grammar G is in Chomsky Normal Form if each production in G is one of two forms:

  1. $A \to BC$ where A, B, and C are nonterminals, or
  2. $A \to a$ where a is a terminal.

- We will further assume G has no useless symbols.

- Every context-free language without ϵ can be generated by a Chomsky Normal Form grammar.

- Let us assume we have a CFG G with no useless symbols, ϵ- productions, or unit productions. We can transform G into an equivalent Chomsky Normal Form grammar as follows:

  – Arrange that all bodies of length two or more consist only of nonterminals.

  – Replace bodies of length three or more with a cascade of fproductions, each with a body of two nonterminals.

- Applying these two transformations to the grammar H above, we get:

$$\begin{aligned}
E &\to EA \,|\, TB \,|\, LC \,|a \\
A &\to PT \\
P &\to + \\
B &\to MF \\
M &\to * \\
L &\to (\\
C &\to ER \\
R &\to) \\
T &\to TB \,|\, LC \,|\, a \\
F &\to LC \,|\, a
\end{aligned}$$