COMS W3261
Computer Science Theory
Lecture 8
Pushdown Automata

Alexander Roth

2014 – 09 – 29

Outline

1. Examples of context-free grammars
2. Pushdown automata (PDA)
3. Instantaneous descriptions of PDA's
4. The language of a PDA
5. Deterministic PDA
6. From a CFG to an equivalent PDA
7. From a PDA to an equivalent CFG

1 Examples of Context-Free Grammars

1. Even-length palindromes: $S \rightarrow aSa \,|\, bSb \,|\, \epsilon$
2. Odd-length palindromes: $S \rightarrow aSa \,|\, bSb \,|\, a \,|\, b$
3. Palindromes with a center marker: $S \rightarrow aSA \,|\, bSb \,|\, c$
4. Prefix notation: $E \rightarrow +EE \,|\, *EE \,|\, a$
5. Postfix notation: $E \rightarrow EE+ \,|\, EE* \,|\, a$
6. Balanced parentheses: $S \rightarrow (S)S \,|\, \epsilon$
7. Arithmetic expressions over id's and num's (ambiguous):

$$E \rightarrow E + E \,|\, E * E \,|\, (E) \,|\, \texttt{id} \,|\, \texttt{num}$$

8. Arithmetic expressions over `id`'s and `num`'s (unambiguous):

$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (E) \mid \texttt{id} \mid \texttt{num}$$

9. Regular expressions over $\{a, b\}$ (ambiguous):

$$R \to R + R \mid RR \mid R^* \mid (R) \mid a \mid b \mid \epsilon \mid \varphi$$

10. If-then, if-then-else statements (ambiguous):

$$S \to \text{if}\, c \,\text{then}\, S$$
$$S \to \text{if}\, c \,\text{then}\, S \,\text{else}\, S$$
$$S \to \text{other\_stat}$$

Class Notes

Membership Problem
Given a CFG G and a string w is w in $L(G)$?
Parsing Problem
Given G and w, if w is in $L(G)$, produce a parse tree form.

2 Pushdown Automata

- A pushdown automaton is an ϵ-NFA with a pushdown stack (last-in, first-out stack).
- Pushdown automata are to context-free languages as finite automata are to regular languages: that is to say, pushdown automata define exactly the context-free languages.
- There are seven components to a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$:
  1. Q is a finite set of states.
  2. Σ is a finite set of input symbols (the input alphabet).
  3. Γ is a finite set of stack symbols (the stack alphabet).
  4. δ is a transition function from $(Q \times (\Sigma \cup \{\epsilon\} \cup \Gamma)$ to subsets of $(Q \times \Gamma^*)$.
     - Suppose $\delta(q, a, X)$ contains (p, γ). Then whenever P is in state q, looking at the input symbol a with X on top of the stack, P may go into state q, move to the next input symbol, and replace X on top of the stack by the string γ.
     - The second component, a, may be ϵ in which case P makes the move without looking at the input symbol and does not move to the next input symbol.

- Note that if P is nondeterministic, there may be more than one pair in $\delta(q, a, X)$. If P is nondeterministic, there may be a pair in $\delta(q, a, X)$ where a is a symbol in Σ and also a pair in $\delta(q, \epsilon, X)$.

5. q_0 is the start state.
6. Z_0 is the start stack symbol.
7. F is the set of final (accepting states).

Class Notes

ϵ means we pop from the stack. `abcba`, where we have $x =$ `ab` and $x^R =$ `ba`, and `c` is the center marker.

3 Instantaneous Descriptions of PDA's

- We can represent a configure of the PDA P above by a triple (q, w, γ) where:
  - q is the state of the finite-state control.
  - w is the string of remaining input symbols.
  - γ is the string of symbols on the stack. If $\gamma = XYZ$, then X is on top of the stack.
- Suppose $\delta(q, a, X)$ contains (p, α). Then to represent a single move of P using this transition we write

$$(q, aw, X\beta) \vdash (p, w, \alpha\beta)$$

  for all strings w in Σ^* and β in Γ^*. Note that a may be ϵ.

4 The Language of a PDA

- A PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ can define a language two ways.
- Acceptance by final state: P can accept an input string w by reading all of it during a sequence of moves and entering a final state at the end.
  - Formally, we define $L(P)$, the language accepted by P by final state, to be the set of input strings w such that P can go from its initial ID (q_0, w, Z_0) in a sequence of zero or more moves to an accepting ID of the form (q, ϵ, α) where q is a final state and α is any stack string (perhaps empty).
- Acceptance by empty (null) stack: P can accept an input string by reading all of it and emptying its stack.

- Formally, we define $N(P)$, the language accepted by P by empty stack, to be the set of input strings w such that P can go from its initial ID (q_0, w, Z_0) in a sequence of zero or more moves to an accepting ID of the form (q, ϵ, ϵ) for any state q.
- Note that the final state of a PDA accepting by empty stack are irrelevant.

- These two modes of acceptance are equivalent. That is, a language L has a PDA that accepts it by final state iff L has a PDA that accepts it by empty stack.

5 Deterministic Pushdown Automata

- A PDA is deterministic (DPDA) if there is never a choice for a next move in any instantaneous description.
- More precisely, a PDA $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is deterministic if:
  1. $\delta(q, a, X)$ has at most one member of any q in Q, a in $\Sigma \cup \{\epsilon\}$ and X in Γ.
  2. If $\delta(q, a, X)$ is nonempty for some a in Σ, then $\delta(q, \epsilon, X)$ must be empty.
- A language that can be recognized by a DPDA is called a deterministic context-free language.
- A DPDA can recognize $\{ wcw^R \mid w$ is any string of a's and b's $\}$.
- A PDA can recognize $\{ ww^R | w$ is any string of a's and b's $\}$, but no DPDA can recognize this language.
- Thus, unlike finite automata, pushdown automata with their nondeterminism are strictly more powerful than deterministic pushdown automata.
- Note that, if L is a regular language, then L can be recognized by a DPDA.
- Since a DPDA can recognize the non-regular language $\{ wcw^R | w$ is any string of a's and b's $\}$, DPDA are strictly more powerful than finite automata.

Class Notes

A PDA is deterministic if it has at least one next node.

6 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 can simulate 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

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

    Suppose the sentential form $\gamma_i = x_i \alpha_i$ where x_i is a prefix of w and α_i is a sequence of input and stack symbols for $1 \le i \le n$. We can show by induction on i that if P simulates this leftmost derivation by the sequence of moves

    $$(q, w, S) |-^* (q, y_i, \alpha_i),$$

    then $x_i y_i = w$.
    This shows that if $S \overset{*}{\Rightarrow} w$, then $(q, w, S) |-^* (q, \epsilon, \epsilon)$. Thus, $L(G)$ is contained in $N(P)$.
  - Only-if part: if $(q, x, A) |-^* (q, \epsilon, \epsilon)$ then $A \overset{*}{\Rightarrow} x$.
    We can prove this statement by induction on the number of moves made by P.
    This shows that if $(q, x, A) |-^* (q, \epsilon, \epsilon)$, then $S \overset{*}{\Rightarrow} w$. Thus, $N(P)$ is contained in $L(G)$.
- We can now conclude $N(P) = L(G)$. Thus, every context-free language can be recognized by a PDA.

From a PDA to an equivalent CFG

- We now show that every language recognized by a PDA can be generated by a context-free grammar.

- 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 $S \to [q_0 Z_0 p]$.
       * If $\delta(q, aX)$ contains $(r, Y_1 Y_2 \ldots Y_k)$, 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 $[qXp] \stackrel{*}{\Rightarrow} w$ iff $(q, w, X) |-^* (p, \epsilon, \epsilon)$.
  - From this, we have $[q_0 Z_0 p] \stackrel{*}{\Rightarrow} w$ iff $(q_0, w, Z_0) |-^* (p, \epsilon, \epsilon)$, so we can conclude $L(G) = N(P)$.

- Sections 6 and 7 allow us to conclude that family of languages generated by context-free grammars is the same as the family of languages recognized by pushdown automata.

- In summary, the regular languages are a proper subset of deterministic CFL's which are a proper subsets of all CFL's.