Lecture 7: Chomsky Normal Form & PDA's

CSCI 81
Spring, 2012
Kim Bruce

Normal Forms

• Because of \( \varepsilon \)-productions, can be hard to determine if \( w \) in \( L \).
  • Parsers recognize terms of language and build abstract syntax tree (thinned down parse tree)
  • Normal forms can make it easier.
  • Chomsky and Greibach Normal Forms
    • Do only Chomsky

Chomsky Normal Form (CNF)

• Require all productions to be of form \( A \rightarrow BC \), \( A \rightarrow a \), or \( S \rightarrow \varepsilon \), where \( S \) is start symbol, & neither \( B \) nor \( C \) can be \( S \).

• Advantages:
  • No \( \varepsilon \)-productions (except from start)
  • Parse trees are all binary
  • To see if \( w \) in \( L \) try all derivations of length \( < 2|w| \)
  • Efficient parsing algorithms (CYK)
    • but watch blowup in size!

Converting grammar to CNF

• Add new start symbol \( S_0 \) and rule \( S_0 \rightarrow S \)

• Eliminate \( \varepsilon \)-productions from all vars except \( S_0 \)
  • If \( A \rightarrow \varepsilon \) is a rule, then drop it and for all rules of the form \( B \rightarrow w \), add all rules of the form \( B \rightarrow w' \) where \( w' \) formed by dropping one or more \( A \)'s from right side of \( w \).
  • Note: Don’t add \( B \rightarrow \varepsilon \) if it has already been dropped.
Converting grammar to CNF

- Eliminate unit productions (size of right is 1)
  - If $A \rightarrow B$ is a rule, then drop it, and for each production $B \rightarrow w$, add $A \rightarrow w$.
  - Note: If $A \rightarrow w$ is a unit production that was already eliminated, then don't add it back.

- Eliminate long right sides:
  - If $A \rightarrow W_1 \ldots W_n$ where each $W_i \in V$, replace by
    - $A \rightarrow W_1 X_1, X_1 \rightarrow W_2 X_2, \ldots, X_{n-2} \rightarrow W_{n-1} W_n$ where $X_i$ are new.

Converting grammar to CNF

- Eliminate terminals on the right side:
  - For each terminal $a \in \Sigma$, add new non-terminal $N_a$ and production $N_a \rightarrow a$. For each production of the form $U \rightarrow w$, for $|w| = 2$, replace all terminals $a$ in $w$ by corresponding $N_a$.

  - Should be clear get same language.

Example

- Start with $S \rightarrow UaabU, U \rightarrow aU | bU | \varepsilon$

- Add new start & eliminate $\varepsilon$-productions:
  - $S_0 \rightarrow S$
    - $U \rightarrow aU | bU | a | b$
    - $S \rightarrow UaabU | aabU | Uaab | aab$

- Eliminate unit productions:
  - $S_0 \rightarrow UaabU | aabU | Uaab | aab$
    - $U \rightarrow aU | bU | a | b$
  - Note $S$ gone as not accessible!

Example

- Shorten long productions & eliminate terminals
  - $S_0 \rightarrow U C_0 | A D_0 | U E_0 | A F_0 | U \rightarrow A U | B U | a | b$
  - $C_0 \rightarrow A C_1$
  - $C_1 \rightarrow A C_2$
  - $C_2 \rightarrow B U$
  - $D_0 \rightarrow A D_1$
  - $D_1 \rightarrow B U$
  - $E_0 \rightarrow A E_1$
  - $E_1 \rightarrow A B$
  - $F_0 \rightarrow A B$
  - $A \rightarrow a$
  - $B \rightarrow b$
Ambiguity

- Java/C statements
  - Cond → if (Exp) Statement
    Cond → if (Exp) Statement else Statement
  - if (x>0)
    if (y>0)
      return 1;
    else
      return 2;
  - return 3;
  - What is returned if x>0 & y ≤ 0? What if x ≤ 0?

Bad News about Ambiguity

- Undecidable whether arbitrary CFG is ambiguous.
- There are inherently ambiguous languages
  - L = {a^i b^j c^k | i, j, k ≥ 0 & (i = j or j = k)}
- Undecidable whether arbitrary CFL is inherently ambiguous.

Pushdown Automata

- A pushdown automaton is a sextuple, (K,Σ,Γ,δ,s,A), where
  - K is a finite set of states,
  - Σ is a finite input alphabet,
  - Γ is a finite stack alphabet,
  - s ∈ K is the start state, and
  - A ⊆ (K × Σ|ε| × Γ*) × (Q × Γ|ε),
  - A ⊆ K is the set of accepting, states.

Configurations

- A configuration of a pda M is an elt (q,w,γ) of K × Σ* × Γ* representing the current state q, the input w left to be read, and the stack contents γ
  - Stack written from top down: c b a where c on top.
  - Initial configuration is (s,w,ε)
  - Define (q, cw, γ γ_{rest}) ⊳_M (q', w, γ' γ_{rest}) iff
    ((q, c, γ), (q', γ')) ∈ Δ
- As usual ⊳_M* is reflexive, transitive closure
Accepting

- A computation C of M is an accepting computation iff:
  - \( C = (s, w, \varepsilon) \vdash M^*(q, \varepsilon, \varepsilon) \), and
  - \( q \in A \). *Need empty stack and accepting state!*
- M accepts a string w iff at least one of its computations accepts.
  - \( L(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \} \)
- Note, in any configuration, can have 0, 1, 2, ... possible moves.

Rejecting

- Computation C of M is rejecting iff
  - \( C = (s, w, \varepsilon) \vdash M^*(q, \varepsilon, \alpha) \),
  - C is not an accepting computation, and
  - M has no moves that it can make from \((q, \varepsilon, \alpha)\).
- M rejects w iff all computations are rejecting
  - Can have w s.t. M neither accepts nor rejects w.
  - Lots of \( \varepsilon \)-moves pushing or popping from stack

Example

- Pda for \( \{a^n b^n \mid n \geq 0\} \)
- Pda for \( \{w \in \{a,b\}^* \mid \text{w reversed}\} \)

A PDA M is deterministic iff:
- \( \Delta_M \) contains no pairs of transitions that compete with each other, and
- Whenever M is in an accepting configuration it has no available moves.
- Pda for \( \{w \in \{a,b\}^* \mid \text{w reversed}\} \)