Lecture 7: Chomsky Normal Form & PDA's

CSCI 81
Spring, 2012

Kim Bruce

Normal Forms

- Because of ε-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 $A \to BC$, $A \to a$, or $S \to \varepsilon$, where S is start symbol, & neither B nor C can be S.
- Advantages:
  - No ε-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 $S_0 \to S$
- Eliminate ε-productions from all vbles except S_0
  - If $A \to \varepsilon$ is a rule, then drop it and for all rules of the form $B \to w$, add all rules of the form $B \to w'$ where w' formed by dropping one or more A's from right side of w.
  - Note: Don't add $B \to \varepsilon$ if it has already been dropped.

Converting grammar to CNF

- Eliminate unit productions (size of right is 1)
  - If $A \to B$ is a rule, then drop it, and for each production $B \to w$, add $A \to w$.
  - Note: If $A \to w$ is a unit production that was already eliminated, then don't add it back.
- Eliminate long right sides:
  - If $A \to W_1 \ldots W_n$ where each $W_i \in V$, replace by
    - $A \to W_1X_1$, $X_1 \to W_2X_2$, ..., $X_{n-2} \to W_{n-1}W_n$ 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 $N_a \to a$. For each production of the form $U \to w$, for $|w| = 2$, replace all terminals a in w by corresponding N_a.
- Should be clear get same language.

Example

- Start with $S \to UaabU$, $U \to aU \mid bU \mid \varepsilon$
- Add new start & eliminate ε-productions:
  - $S_0 \to S$
    $U \to aU \mid bU \mid a \mid b$
    $S \to UaabU \mid aabU \mid Uaab \mid aab$
- Eliminate unit productions:
  - $S_0 \to UaabU \mid aabU \mid Uaab \mid aab$
    $U \to aU \mid bU \mid a \mid b$
  - Note S gone as not accessible!

Example

- Shorten long productions & eliminate terminals

$S_0 \to U\,C_0 \mid AD_0 \mid UE_0 \mid AF_0$
$C_0 \to AC_1$
$C_1 \to AC_2$
$C_2 \to BU$
$D_0 \to AD_1$
$D_1 \to BU$
$E_0 \to AE_1$
$E_1 \to AB$
$F_0 \to AB$

$U \to AU \mid BU \mid a \mid b$

$A \to a$
$B \to b$

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
  - $L = \{a^i b^j c^k \mid i, j, k \geq 0 \ \& \ (i = j \text{ or } j = k)\}$
- Undecidable whether arbitrary CFL is inherently ambiguous.

Pushdown Automata

- A pushdown automaton is a sextuple, $(K,\Sigma,\Gamma,\delta,s,A)$, where
  - K is a finite set of states,
  - Σ is a finite input alphabet,
  - Γ is a finite stack alphabet,
  - $s \in K$ is the start state, and
  - $\Delta \subseteq (K \times \Sigma \cup \{\varepsilon\} \times \Gamma^*) \times (Q \times \Gamma\varepsilon)$,
  - $A \subseteq K$ is the set of accepting, states.

Configurations

- A configuration of a pda M is an elt (q,w,γ) of $K \times \Sigma^* \times \Gamma^*$ 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 $(q, cw, \gamma\ \gamma_{rest}) \vdash_M (q', w, \gamma'\ \gamma_{rest})$ iff
  $((q, c, \gamma), (q', \gamma')) \in \Delta$
- As usual $\vdash_M^*$ 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
  - $C = (s, w, \varepsilon) \vdash_M^* (q, w', \alpha)$,
  - C is not an accepting computation, and
  - M has no moves that it can make from (q, ε, α).
- M rejects w iff all computations are rejecting
  - Can have w s.t. M neither accepts nor rejects w.
  - Lots of ε-moves pushing or popping from stack

Example

- Pda for $\{a^n b^n \mid n \geq 0\}$
- Pda for $\{w\ c\ w^R \mid w\ \varepsilon\ \{a,b\}^*\}$
- A PDA M is deterministic iff:
  - Δ_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 $\{w\ w^R \mid w\ \varepsilon\ \{a,b\}^*\}$