COMS3261: Computer Science Theory

Fall 2013

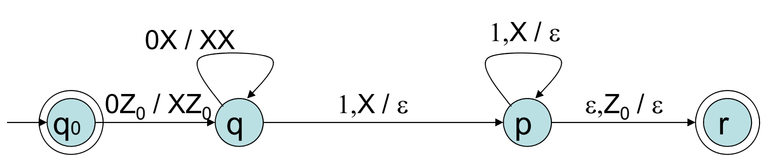
Mihalis Yannakakis

Lecture 13, 10/16/13

Deterministic PDA (DPDA)

- In every situation (for every ID) at most 1 choice of move
- $\forall$state q, $\forall$input symbol a, $\forall$stack symbol X, $|\delta(q,a,X)|\leq 1$
  i.e., $\delta(q,a,X)$ has at most one member (p,α)
- Furthermore, if $\delta(q,\varepsilon,X) \neq \varnothing$ then $|\delta(q,\varepsilon,X)|=1$ and all $\delta(q,a,X)=\varnothing$ for all $a\in\Sigma$
- Acceptance by final state.

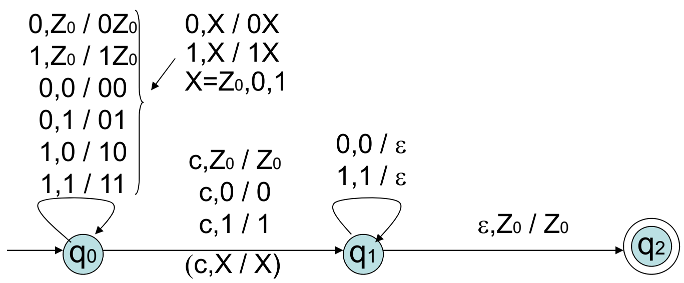
- Example: DPDA for $\{ 0^n1^n | n\geq 0 \}$



DPDA

- PDA for { ww^R | w $\in \Sigma^*$} was nondeterministic

- { wcw^R | w $\in \{0,1\}^*$} can be accepted by a DPDA: c marks the middle of the string, so DPDA knows when to change from pushing to popping



DPDA

- PDA for { ww^R | w $\in \Sigma^*$} was nondeterministic

- Theorem: { ww^R | w $\in \Sigma^*$} cannot be accepted by any DPDA
  $\Rightarrow$ DPDAs weaker than PDAs

- Theorem: Every regular language accepted by a DPDA

Proof: Accepted by a DFA = DPDA that ignores the stack.

Acceptance by ∅ stack is weaker for DPDA

- Once DPDA empties stack, it cannot do anything any more (no moves on empty stack)

$\Rightarrow$ if string is accepted then cannot accept any extension $\Rightarrow$ language L=N(P) is prefix-free: $w \in L \Rightarrow$ no prefix in L

Example: {0}* not prefix-free

Theorem: L=N(P) for some DPDA P iff L is prefix-free and L=L(P') for some DPDA P'

Acceptance by ∅ stack for DPDA

Theorem: L=N(P') for some DPDA P' iff L is prefix-free and L=L(P) for some DPDA P

Parsers as in YACC are really DPDAs, usually accept by empty stack.

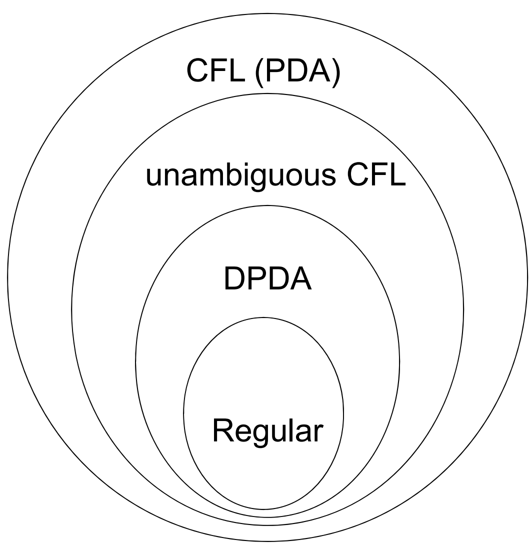
- L may not have prefix-free property, but can change L to L' = L$ where $ is an end-marker (new symbol)
- Then L' has prefix-free property and L'=N(P') for some DPDA P'
- Parsers attach an $ and accept by ∅ stack , i.e., if parser accepts w$ then w is a legal program

DPDA $\Rightarrow$ Unambiguous CFG

- Theorem: If L is accepted by a DPDA then L has an unambiguous CFG.
- Proof: If L=N(P) for a DPDA then construction from PDA to CFG yields an unambiguous grammar: leftmost derivations of grammar mimic computation of PDA P $\Rightarrow$ unique
- For L= L(P) attach endmarker \$ to get L' =L\$
- Construct CFG G' for L' which is unambiguous by construction. Then make \$ into variable and add production $\$ \rightarrow \varepsilon$ to get a CFG G for L.
- Since G' has unique leftmost derivations:

  $S \Rightarrow_{lm}^{*} w\$ \Rightarrow_{lm} w$ unique

Relations



Pumping Lemma for CFLs

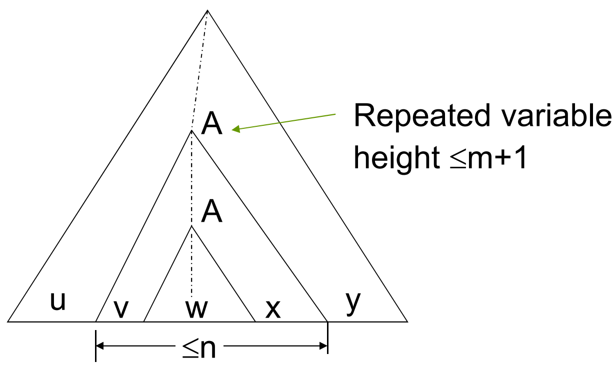
- For every CFL L
  $\exists$ integer n
  $\forall$ z $\in$L with $|z| \geq n$
  $\exists$ partition uvwxy = z with $|vwx| \leq n$, $|vx| > 0$
  $\forall$ $i \geq 0$ $uv^iwx^iy \in L$

- Can pump two substrings, at least one of them not empty

Proof of pumping lemma

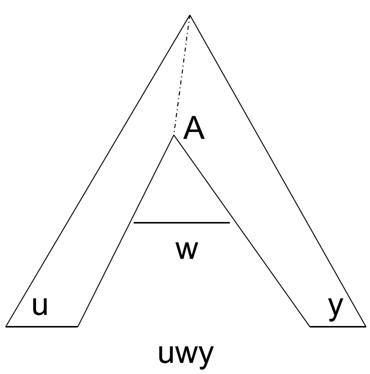
- Take a context-free grammar G for L, let m = #variables and b = max length of a rhs (body) of a production
- Pick $n = b^{m+2}$
- Given any z in L of length $|z| \geq n$, take a parse tree for z with minimum number of nodes
- All internal nodes have degree $\leq$ b
  $\Rightarrow$ If tree height =h then #leaves $\leq b^h$
- #leaves $\geq |z| \geq n = b^{m+2} \Rightarrow$ height of tree $\geq$ m+2 $\Rightarrow$
  $\Rightarrow \exists$ path of tree of length $\geq$ m+2 $\Rightarrow$ it repeats a variable in the last m+1 steps. Let A be such a variable.

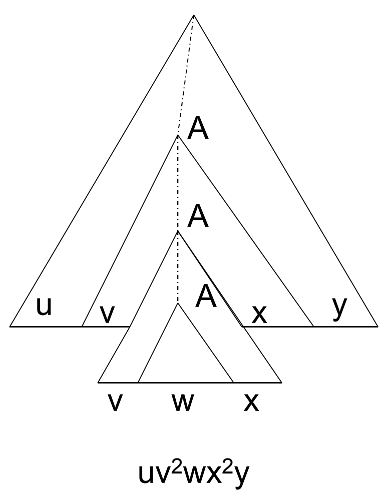
Proof of pumping lemma (ctd): Parse tree of z



height of upper A $\leq$ m+1 $\Rightarrow$ $|vwx| \leq b^{m+1} \leq n$

Proof of pumping lemma (ctd): Valid parse trees of other strings





$|vx| > 0$, i.e. at least one of v,x $\neq \varepsilon$ because otherwise uwy =z and we would get a smaller parse tree for z

Showing a language is not CFL

• Adversary game to show that a language is not context-free, similar to the game for regular languages

- Adversary
  1. Picks n
- We
  2. Pick string z in L, $|z| \geq n$
- Adversary
  3. Partitions z into uvwxy s.t. $|vwx| \leq n$ and $|vx| > 0$
- We
  4. Pick an $i \geq 0$, s.t. $uv^iwx^iy \notin L$

• Have to argue that no matter what the adversary does in steps 1 and 3 (i.e. what n he picks, and how he breaks up the string z), we can succeed.

Example application of pumping lemma

- $L = \{ a^kb^kc^k \mid k \geq 1 \}$ not CFL
- Adversary picks n
- We pick $z = a^nb^nc^n$
- Adversary partitions z = uvwxy
- $|vwx| \leq n \Rightarrow$ either in a-b part or in b-c part (not both a,c)
- Case 1: In a-b part. Pumping up v,x increases a's and/or b's but not c's
- Case 2: In b-c part. Pumping up v,x increases b's and/or c's but not a's

Example application of pumping lemma

- $L = \{ w \# w \mid w \in \{0,1\}^* \}$ not CFL
- Adversary picks n
- We pick $z = 0^n 1^n \# 0^n 1^n$
- Adversary partitions z = uvwxy
- $|vwx| \leq n \Rightarrow$ vwx cannot overlap both the block of 0's in the first half and the block of 0's in the second half;

  also it cannot overlap both blocks of 1's.
- $|vx| > 0 \Rightarrow$ uv^2wx^2y adds another # or increases the number of 0's or 1's in one half but not in the other half

  $\Rightarrow$ $uv^2wx^2y \notin L$