Harvard CS 121 and CSCI E-207

Lecture 10: CFLs: PDAs, Closure Properties, and Non-CFLs

Harry Lewis

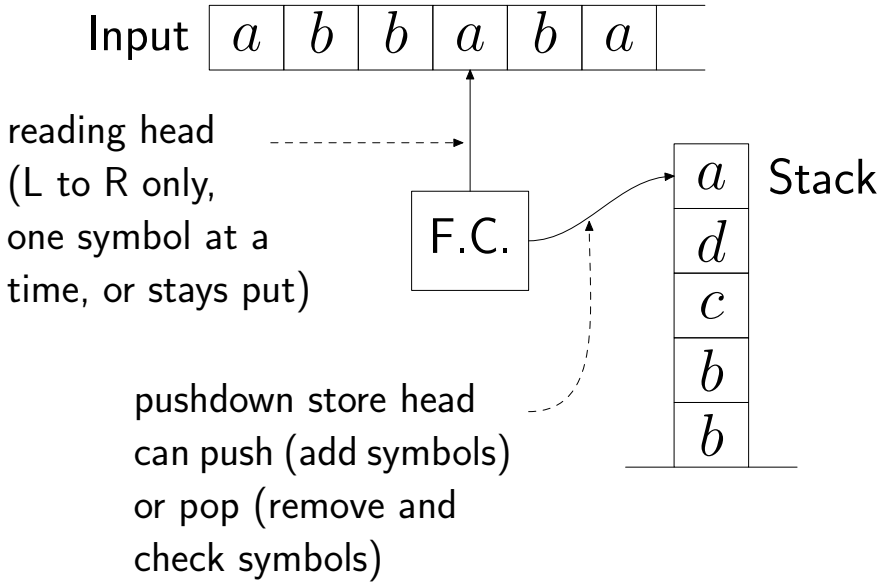
October 8, 2013

Reading: Sipser, pp. 119-128.

Pushdown Automata (review)

= Finite automaton + "pushdown store"

- The pushdown store is a stack of symbols of unlimited size which the machine can read and alter only at the top.



Transitions of PDA are of form $(q, \sigma, \gamma) \mapsto (q', \gamma')$, which means:

If in state q with σ on the input tape and γ on top of the stack, replace γ by γ' on the stack and enter state q' while advancing the reading head over σ.

(Nondeterministic) PDA for "even palindromes"

$$\{ww^R : w \in \{a,b\}^*\}$$

$(q, a, \varepsilon) \mapsto (q, a)$	Push a's
$(q, b, \varepsilon) \mapsto (q, b)$	and b's
$(q, \varepsilon, \varepsilon) \mapsto (r, \varepsilon)$	switch to other state
$(r, a, a) \mapsto (r, \varepsilon)$	pop a's matching input
$(r, b, b) \mapsto (r, \varepsilon)$	pop b's matching input

So the precondition (q, σ, γ) means that

- the next $|\sigma|$ symbols (0 or 1) of the input are σ and
- the top $|\gamma|$ symbols (0 or 1) on the stack are γ

(Nondeterministic) PDA for "even palindromes"

$$\{ww^R : w \in \{a,b\}^*\}$$

$(q, a, \varepsilon) \mapsto (q, a)$	Push a's
$(q, b, \varepsilon) \mapsto (q, b)$	and b's
$(q, \varepsilon, \varepsilon) \mapsto (r, \varepsilon)$	switch to other state
$(r, a, a) \mapsto (r, \varepsilon)$	pop a's matching input
$(r, b, b) \mapsto (r, \varepsilon)$	pop b's matching input

Need to test whether stack empty: push $\$$ at beginning and check at end.

$(q_0, \varepsilon, \varepsilon) \mapsto (q, \$)$
$(r, \varepsilon, \$) \mapsto (q_f, \varepsilon)$

Language recognition with PDAs

A PDA accepts an input string

If there is a computation that starts

- in the start state
- with reading head at the beginning of string
- and the stack is empty

and ends

- in a final state
- with all the input consumed

A PDA computation becomes "blocked" (i.e. "dies") if

- no transition matches *both* the input and stack

Equivalence of CFGs and PDAs

Thm: The class of languages recognized by PDAs is the CFLs.

I: For every CFG G,
there is a PDA M
with $L(M) = L(G)$.

II: For every PDA M,
there is a CFG G
with $L(G) = L(M)$.

Proof that every CFL is accepted by some PDA

Let $G = (V, \Sigma, R, S)$

We'll allow a generalized sort of PDA that can push *strings* onto stack.

E.g., $(q, a, b) \mapsto (r, cd)$

Proof that every CFL is accepted by some PDA

Let $G = (V, \Sigma, R, S)$

We'll allow a generalized sort of PDA that can push *strings* onto stack.

E.g., $(q, a, b) \mapsto (r, cd)$

Then corresponding PDA has just 3 states:

$q_{\text{start}} \sim$ start state

$q_{\text{loop}} \sim$ "main loop" state

$q_{\text{accept}} \sim$ final state

Stack alphabet = $V \cup \Sigma \cup \{\$\}$

CFL $\Rightarrow$ PDA, Continued: The Transitions of the PDA

- $\delta(q_{\text{start}}, \varepsilon, \varepsilon) = \{(q_{\text{loop}}, S\$)\}$

  "Start by putting $S\$$ on the stack, & go to q_{loop}"

- For each $A \in V$, $\delta(q_{\text{loop}}, \varepsilon, A) = \{(q_{\text{loop}}, w) : A \rightarrow w \in R\}$

  "Remove a variable from the top of the stack and replace it with a corresponding righthand side"

- For each $\sigma \in \Sigma$, $\delta(q_{\text{loop}}, \sigma, \sigma) = \{(q_{\text{loop}}, \varepsilon)\}$

  "Pop a terminal symbol from the stack if it matches the next input symbol"

- $\delta(q_{\text{loop}}, \varepsilon, \$) = \{(q_{\text{accept}}, \varepsilon)\}$.

  "Go to accept state if stack contains only $\$$."

Example

- Consider grammar G with rules $\{S \to aSb, S \to \varepsilon\}$

  (so $L(G) = \{a^n b^n : n \geq 0\}$)

- Construct PDA
  $M = (\{q_{\text{start}}, q_{\text{loop}}, q_{\text{accept}}\}, \{a, b\}, \{a, b, S, \$\}, \delta, q_{\text{start}}, \{q_{\text{accept}}\})$

  Transition Function δ :

Computation Corresponding to a Derivation

- Derivation $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$

$$
\begin{aligned}
(p, aabb, \varepsilon) &\vdash (q, aabb, S) && (1)\\
&\vdash (q, aabb, aSb) && (2)\\
&\vdash (q, abb, Sb) && (4)\\
&\vdash (q, abb, aSbb) && (2)\\
&\vdash (q, bb, Sbb) && (4)\\
&\vdash (q, bb, bb) && (3)\\
&\vdash (q, b, b) && (5)\\
&\vdash (q, \varepsilon, \varepsilon) && (5)
\end{aligned}
$$

Every language accepted by a PDA is CF

Proof: See Sipser

Closure Properties of CFLs

- **Thm:** The CFLs are the languages accepted by PDAs
- **Thm:** The CFLs are closed under
  - Union
  - Concatenation
  - Kleene $*$
  - Intersection with a regular set

The intersection of a CFL and a regular set is a CFL

Pf sketch: Let L_1 be CF and L_2 be regular

$L_1 = L(M_1)$, M_1 a PDA

$L_2 = L(M_2)$, M_2 a DFA

$Q_1 =$ state set of M_1

$Q_2 =$ state set of M_2

Construct a PDA with state set $Q_1 \times Q_2$ which keeps track of computation of both M_1 and M_2 on input.

Q: Why doesn't this argument work if M_1 and M_2 are both PDAs?

In fact, the intersection of two CFLs is not necessarily CF.

And the complement of a CFL is not necessarily CF

Q: How to prove that languages are not context free?

Pumping Lemma for CFLs

Lemma: If L is context-free, then there is a number p (the pumping length) such that any $s \in L$ of length at least p can be divided into $s = uvxyz$, where

1. $uv^ixy^iz \in L$ for every $i \geq 0$,
2. $v \neq \varepsilon$ or $y \neq \varepsilon$, and
3. $|vxy| \leq p$.

Pumping Lemma for CFLs (aka Yuvecksy'sTheorem ;)

Lemma: If L is context-free, then there is a number p (the pumping length) such that any $s \in L$ of length at least p can be divided into $s = uvxyz$, where

1. $uv^ixy^iz \in L$ for every $i \geq 0$,
2. $v \neq \varepsilon$ or $y \neq \varepsilon$, and
3. $|vxy| \leq p$.

Using the Pumping Lemma to Prove Non-Context-Freeness

$\{a^n b^n c^n : n \geq 0\}$ is not CF.

aaaaaaaaaaaaaaaaaa	*bbbbbbbbbbbbbbbbbbb*	*ccccccccccccccccccccccc*

What are v, y?

- Contain 2 kinds of symbols
- Contain only one kind of symbol

$\Rightarrow$ **Corollary:** CFLs not closed under intersection (why?)

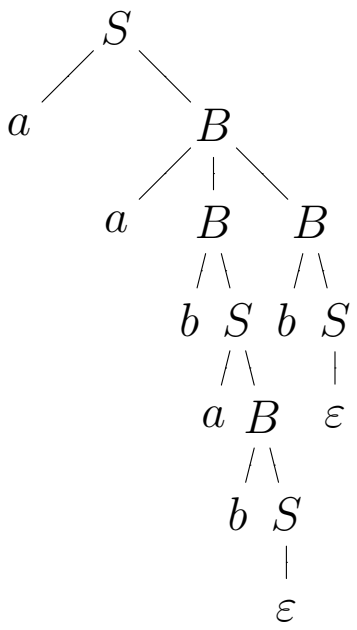
$\Rightarrow$ **Corollary:** CFLs not closed under complement (why?)

- Is the the intersection of 2 CFLs or the complement of a CFL *sometimes* a CFL?

What about $\{a, b\}^* - \{a^n b^n : n \geq 0\}$?

Recall: Parse Trees

$$
\begin{array}{lcl}
S & \rightarrow & aB \mid bA \mid \varepsilon \\
A & \rightarrow & aS \mid bAA \\
B & \rightarrow & bS \mid aBB
\end{array}
$$

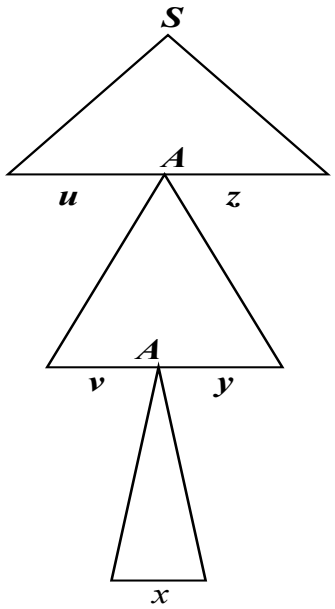


Parse tree for $aababb$, the "yield" of the tree

Height = max length path from S to a terminal symbol = 6 in above example.

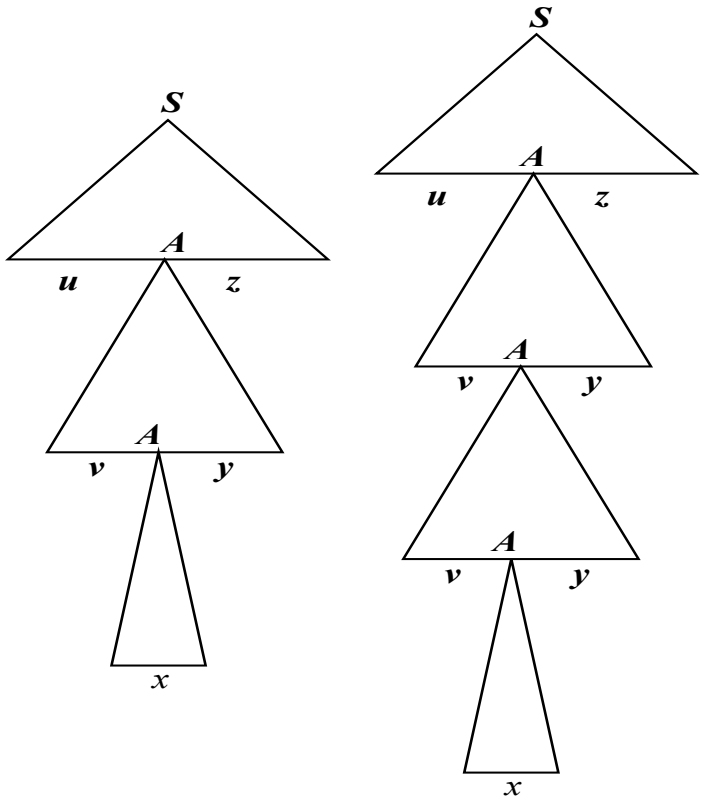
Proof of Pumping Theorem

Show that there exists a p such that any string s of length $\geq p$ has a parse tree of the form:



Proof of Pumping Theorem

Show that there exists a p such that any string s of length $\geq p$ has a parse tree of the form:



Finding "Repetition" in a big parse tree

- Since RHS of rules have bounded length, long strings must have tall parse trees.
- A tall parse tree must have a path with a repeated nonterminal.
- Let $p = b^m + 1$, where:

  $b =$ max length of RHS of a rule

  $m =$ # of variables

- Suppose T is the smallest parse tree for a string $s \in L$ of length at least p. Then

  Let $h =$ height of T. Then $b^h \geq p = b^m + 1$,

  $\Rightarrow h > m$,

  $\Rightarrow$ Path of length h in T has a repeated variable.

Final annoying details

- **Q:** Why is v or y nonempty?
- **Q:** How to ensure $|vxy| \leq p$?

The converse of the CF Pumping Lemma is False

Some non-context-free languages satisfy conclusion of Pumping Lemma, e.g. ?

Some Other CF Closure Properties

Let $L_1/L_2 = \{w : wx \in L_1 \text{ for some } x \in L_2\}$. Then

- If L is CF and R is regular then L/R is CF