Harvard CS 121 and CSCI E-207

Lecture 11: Pushdown Automata and Context-Free Languages

Salil Vadhan

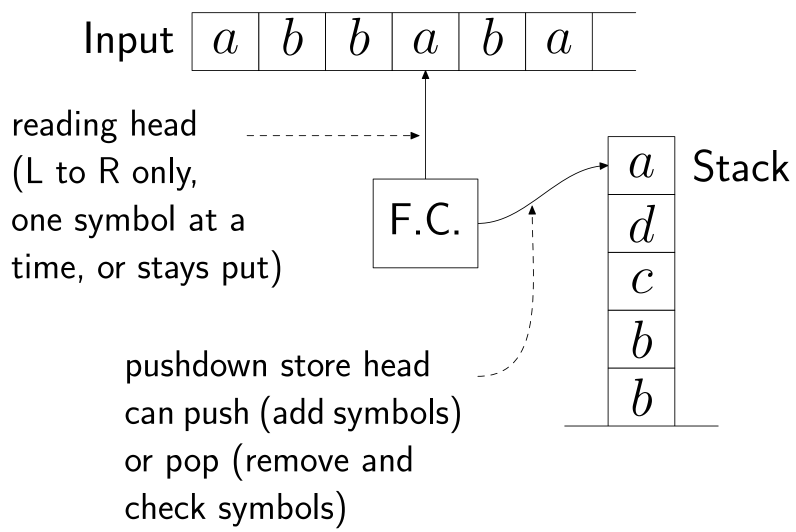
October 9, 2012

- **Reading:** Sipser, §2.2.

Pushdown Automata

= 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

Formal Definition of a PDA

- $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

  $Q =$ states

  $\Sigma =$ input alphabet

  $\Gamma =$ stack alphabet

  $\delta =$ transition function

  $$Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to P(Q \times (\Gamma \cup \{\varepsilon\})).$$

  $q_0 =$ start state

  $F =$ final states

Computation by a PDA

- M accepts w if we can write $w = w_1 \cdots w_m$, where each $w_i \in \Sigma \cup \{\varepsilon\}$, and there is a sequence of states $r_0, \ldots, r_m$ and stack strings $s_0, \ldots, s_m \in \Gamma^*$ that satisfy

1. $r_0 = q_0$ and $s_0 = \varepsilon$.
2. For each i, $(r_{i+1}, \gamma') \in \delta(r_i, w_{i+1}, \gamma)$ where $s_i = \gamma t$ and $s_{i+1} = \gamma' t$ for some $\gamma, \gamma' \in \Gamma \cup \{\varepsilon\}$ and $t \in \Gamma^*$.
3. $r_m \in F$.

- $L(M) = \{w \in \Sigma^* : M \text{ accepts } w\}$.

PDA for $\{w \in \{a,b\}^* : \#_a(w) = \#_b(w)\}$

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

Transitions:

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

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

- $\delta(q_{\text{loop}}, \varepsilon, A) = \{(q_{\text{loop}}, w)\}$ for each rule $A \to w$

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

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

  "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 :

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

  Corresponding Computation:

Proof That For Every PDA M there is a CFG G Such That $L(M) = L(G)$

- First modify PDA M so that
  - Single accept state.
  - All accepting computations end with empty stack.
  - In every step, push a symbol or pop a symbol but not both.

Design of the grammar G equivalent to PDA M

- Variables: A_{pq} for every two states p, q of M.
- Goal: A_{pq} generates all strings that can take M from p to q, beginning & ending w/empty stack.
- Rules:
  - For all states p, q, r, $A_{pq} \to A_{pr}A_{rq}$.
  - For states p, q, r, s and $\sigma, \tau \in \Sigma$, $A_{pq} \to \sigma A_{rs}\tau$ if there is a stack symbol γ such that $\delta(p, \sigma, \varepsilon)$ contains (r, γ) and $\delta(s, \tau, \gamma)$ contains (q, ε).
  - For every state p, $A_{pp} \to \varepsilon$.
- Start variable: $A_{q_{\text{start}}q_{\text{accept}}}$.

Sketch of Proof that the Grammar is Equivalent to the PDA

- **Claim:** $A_{pq} \Rightarrow^* w$ if and only if w can take M from p to q, beginning & ending w/empty stack.

$\Rightarrow$ Proof by induction on length of derivation.

$\Leftarrow$ Proof by induction on length of computation.

  - Computation of length 0 (base case): Use $A_{pp} \to \varepsilon$.
  - Stack empties sometime in middle of computation: Use $A_{pq} \to A_{pr}A_{rq}$.
  - Stack does not empty in middle of computation: Use $A_{pq} \to \sigma A_{rs}\tau$.

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 (aka Yuvecksy's Theorem ;)

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*	*cccccccccccccccccccccc*

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?)