COMS3261:
Computer Science Theory

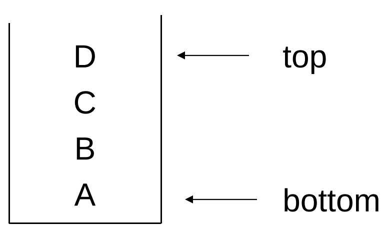
Fall 2013

Mihalis Yannakakis

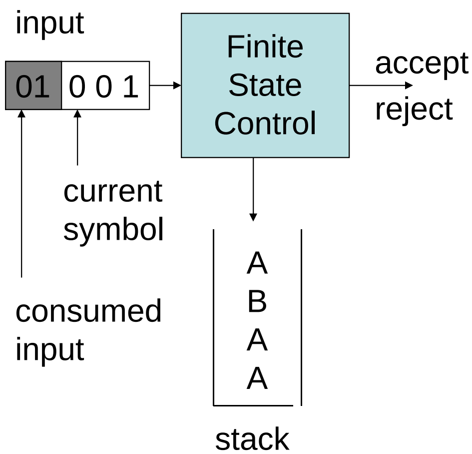
Lecture 11, 10/9/13

Pushdown Automata (PDA)

- PDA = ε-NFA + stack (pushdown store)
- Stack is the right data structure (Last-In-First-Out principle) for implementing recursion (recursive procedure calls):
  - A calls B which calls C which calls D …
  - Keep record of active calls in stack so we know how to return properly when each procedure call ends



Pushdown Automaton



- Transition of PDA

In one step, depending on:
- current state
- current input symbol or ε
- symbol on top of stack

the PDA consumes the input symbol, and:
- moves to next state
- can replace the top of the stack by some string
  (i.e. does nothing, pops the stack or pops and pushes another string)

ε-transition: spontaneous, does not read or consume input

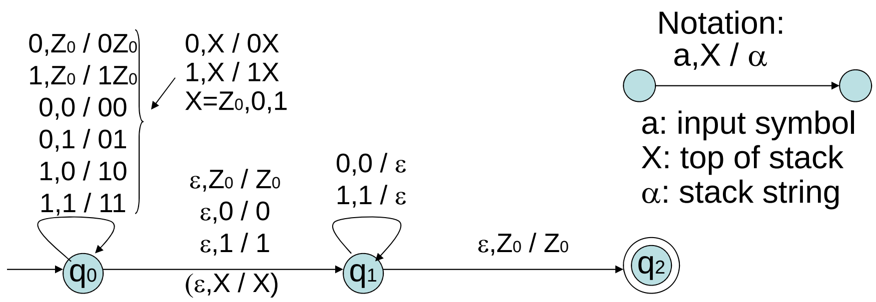
Formal Definition

- PDA $A = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$
- Q = finite set of states
- Σ = finite input alphabet
- Γ = finite stack alphabet
- $q_0 \in Q$ = initial (start) state
- $Z_0 \in \Gamma$ = initial (start) symbol on stack
- $F \subseteq Q$ = set of accepting states
- $\delta : Q\times(\Sigma \cup \{\varepsilon\})\times\Gamma \to 2^{Q\times \Gamma^*}$ = transition function
- $\delta(q,a,X)$ or $\delta(q,\varepsilon,X)$ = set of tuples (p,α), $p \in Q$, $\alpha\in\Gamma^*$, one for each choice of move on state q, input a, stack top=X
- $\alpha = \varepsilon$: pop; α=X: no change; α=Y: replace X by Y; α=YX: push Y

Note: no transition on empty stack

Example: L = { ww^R | $w \in \{0,1\}^*$ }

- PDA reads w and stores it in the stack.
- When it reaches the middle (guess nondeterministically), it pops the stack comparing with the 2^{nd} half of the input
- Nondeterminism critical
- Formally $\Gamma=\{Z_0,0,1\}$, $Q=\{q_0,q_1,q_2\}$, $F=\{q_2\}$
- q_0 : 1^{st} half, q_1: 2^{nd} half, q_2: accept



Transition Table

- Can specify transitions also by transition table
- Rows: states
- Columns: (a,X) or (ε,X) where a=input symbol, X=stack symbol
- entry= set of (p,α) pairs where p=next state, α=stack string

	$0,Z_0$	$1,Z_0$	
$\rightarrow q_0$	$\{q_0,0Z_0\}$	$\{q_0,1Z_0\}$	
q_1			
$*\ q_2$			

Language of a PDA

- Instantaneous Description of a PDA:
  ID (q,w,γ) = (state, remaining input, stack)
- Configuration of PDA: (q,γ) =
  "full" state (memory) of PDA = state + stack
  (convention: stack written with top on left, bottom on right)
- Transition = move from ID to ID:
  If $(p,\alpha) \in \delta(q,a,X)$ then $(q,aw,X\beta)$ |-- $(p,w,\alpha\beta)$
  If $(p,\alpha) \in \delta(q,\varepsilon,X)$ then $(q,w,X\beta)$ |-- $(p,w,\alpha\beta)$
- |--* = reflexive transitive closure of |-- (move in 0 or more steps)

- Language accepted by PDA A:
- $L(A) = \{ w \in \Sigma^* \mid (q_0,w,Z_0)$ |--* (p,ε,γ) for some $p \in F, \gamma \in \Gamma^* \}$

Example of computation tree

- On input 0110

$(q_0,0110,Z_0) \rightarrow (q_1,0110,Z_0) \rightarrow (q_2,0110,Z_0)$

$(q_0,110,0Z_0) \rightarrow (q_1,110,0Z_0)$

$(q_0,10,10Z_0) \rightarrow (q_1,10,10Z_0)$

$(q_0,0,110Z_0)$

$(q_1,0,0Z_0)$

(q_1,ε,Z_0)

(q_2,ε,Z_0)

Properties of computations

- (q,x,α) |--* $(p,y,\beta) \Rightarrow (q,xw,\alpha\gamma)$ |--* $(p,yw,\beta\gamma)$
  for all $w \in \Sigma^*, \gamma \in \Gamma^*$

  i.e. appending a string to end of input or bottom of stack gives another legal computation (these are not examined in computation, so don't matter)
- (q,xw,α) |--* $(p,yw,\beta) \Rightarrow (q,x,\alpha)$ |--* (p,y,β)

  i.e. if the computation did not get to the w segment of the input, then it is irrelevant

Example: Proof of language

- $L(A) = \{ ww^R \mid w \in \{0,1\}^* \}$
- Two directions: $\subseteq$ and $\supseteq$
- $\supseteq$ For any string ww^R in the language, show an accepting computation. Formally by induction on length of w, show: (q_0,ww^R,Z_0) |--* (q_0,w^R,w^RZ_0) and (q_1,w^R,w^RZ_0) |--* (q_1,ε,Z_0)
  Then (q_0,ww^R,Z_0) |--*(q_0,w^R,w^RZ_0) |-- (q_1,w^R,w^RZ_0) |--* (q_1,ε,Z_0) |-- (q_2,ε,Z_0)
- $\subseteq$: PDA enters accepting state q_2 only from q_1 when top stack symbol is Z_0. Suppose (q_0,x,Z_0) |--* (q_1,ε,Z_0) |--(q_2,ε,Z_0) , will show $x = ww^R$ for some w.
- Will show by induction on |x| that (q_0,x,α) |--* (q_1,ε,α) for some α implies $x = ww^R$ for some w.

Proof ctd.

- Basis: $|x| = 0 \Rightarrow x = \varepsilon \Rightarrow$ ok
- Induction step: $x = a_1 \ldots a_n$, where $n>0$

  From ID (q_0,x,α) 2 choices: (q_1,x,α) and $(q_0,a_2 \ldots a_n,a_1\alpha)$
- Moves from q_1 consume input and stack symbols, so cannot have (q_1,x,α) |--* (q_1,ε,α) $\Rightarrow$ 2nd choice
- Must have $(q_0,a_2 \ldots a_n,a_1\alpha)$ |-- … |-- $(q_1,a_n,a_1\alpha)$ |-- (q_1,ε,α)

  $\Rightarrow a_n = a_1$. and $(q_0,a_2 \ldots a_n,a_1\alpha)$ |--* $(q_1,a_n,a_1\alpha)$

  $\Rightarrow (q_0, a_2 \ldots a_{n-1},a_1\alpha)$ |--* $(q_1,\varepsilon,a_1\alpha)$ (by properties)

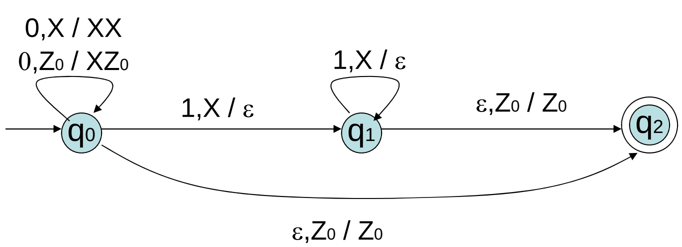
  $\Rightarrow a_2 \ldots a_{n-1} = yy^R$ for some y (by induction hypothesis)

  $\Rightarrow x = ww^R$ for $w= a_1y$

Example: $L =\{ 0^n1^n \mid n \geq 0 \}$

- Stack with 1 symbol X can be used as a counter that can be incremented (push X), decremented (pop X), tested for 0
- Algorithm: while reading 0's push X's to stack (counter++),

  while reading 1's, pop X's from stack (counter - -),
  if Z_0 (bottom-of-stack) accept



Example: Balanced Parentheses

- Examples: (()) , () (), (() ()) …. are balanced
- (() , ()) (, ( )) ( ))   are not balanced
- Essential in many contexts: valid arithmetic expressions,

beginning/ends of code blocks: begin–end in Pascal, { } in C

- Extension to different types of parentheses

  - arithmetic expressions
  - tags in markup languages (HTML, XML)
  - beginnings / ends of recursive calls in recursive programs

- Match between corresponding left and right parentheses (of the same type if multiple types), where each right parenthesis matches the first previous unmatched left parenthesis (of the same type)

Example : Balanced Parentheses ctd.

- CFG for balanced parentheses (of one type)

  $S \rightarrow \varepsilon \mid SS \mid (S)$

- Pushdown automaton

  Stack symbols Z_0 (bottom-of-stack) and X

  Algorithm: On a left (, push a X on the stack,

  on a right) , pop a X from the stack

  [if top of stack not X then no move: failure]

  at all times if top=Z_0, can make ε transition to an accepting state that has no transitions

HW: Translate the algorithm to a formal PDA

Extend to parentheses of multiple types