BU CS 332 – Theory of Computation

Lecture 7:

- More on CFGs
- Pushdown Automata

Midterm I 2/24

Reading:

Sipser Ch 2.1-2.3

My OH 4-6 today

Friday OH 2:30-4:30

MCS B30

Mark Bun

February 12, 2020

Context-Free Grammar (Formal)

A CFG is a 4-tuple $G = (V, \Sigma, R, S)$

- V is a finite set of variables
- Σ is a finite set of terminal symbols (disjoint from V)
- R is a finite set of production rules of the form $A \rightarrow w$, where $A \in V$ and $w \in (V \cup \Sigma)^*$
- $S \in V$ is the start symbol

Context-Free Grammar

Example Grammar G

$S \rightarrow 0S1A \mid 1S0A \mid \varepsilon$

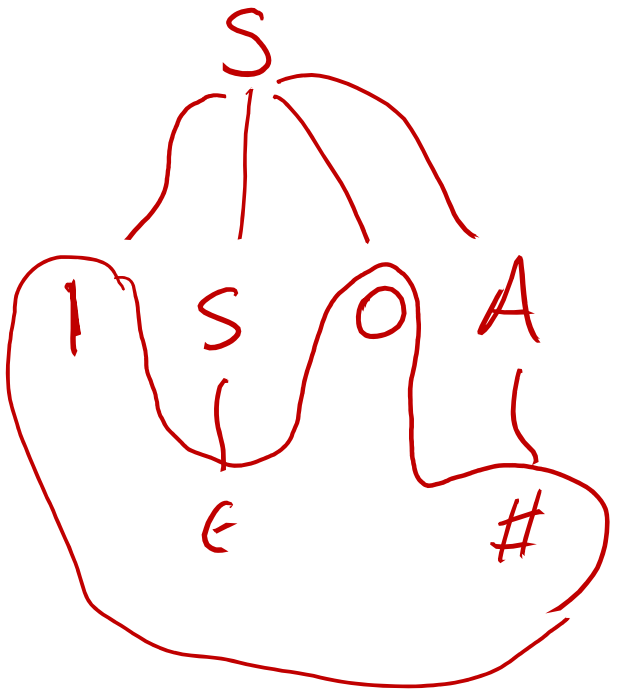
$A \rightarrow \# \mid \varepsilon$

Derivation

$S \Rightarrow 1S0A \Rightarrow 1\varepsilon 0A = 10A$

$\Rightarrow 10\#$

Parse Tree

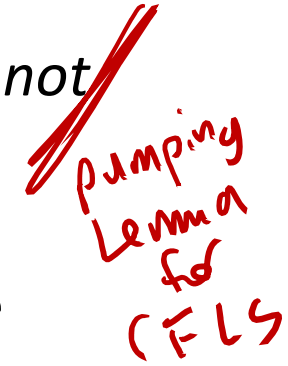


Context-Free Languages

L is a *context-free language* if it is the language of some CFG

Questions about CFLs

1. Which languages are *not* context-free?
2. How do we recognize whether $w \in L$?
3. What are the closure properties of CFLs?



Pumping Lemma for context-free languages

Let L be a context-free language.

Then there exists a "pumping length" p such that

For every $w \in L$ where $|w| \geq p$,
w can be split into five parts $w = uvxyz$ where:

1. $|vy| > 0$
2. $|vxy| \leq p$
3. $uv^ixy^iz \in L$ for all $i \geq 0$

Pumping Lemma example

Claim: $L = \{a^n b^n c^n | n \geq 0\}$ is not context-free

Proof: Assume L is context-free with pumping length p

1. Find $w \in L$ with $|w| \geq p$ $w = a^p b^p c^p$
2. Show that w cannot be pumped

   If $w = uvxyz$ with $|vy| > 0, |vxy| \leq p$, then…

Case 1: v, y both contain only one kind of symbol

Case 2: Either v or y contains two kinds of symbols

Pumping Lemma example

$v = a^k \quad y = c^l$

$v = a^k \quad y = a^l$

uv^2xy^2z has either too many a's or too many c's

Claim: $L = \{a^n b^n c^n | n \geq 0\}$ is not context-free

Proof: Assume L is context-free with pumping length p

1. Find $w \in L$ with $|w| \geq p$
2. Show that w cannot be pumped

   If $w = uvxyz$ with $|vy| > 0, |vxy| \leq p$, then…

Case 1: v, y both contain only one kind of symbol

⇒ one of the symbols a, b, c is not in either v or y

uv^2xy^2z

"Missed" symbol appears p times

At least one non-missed symbol appear $> p$ times

Case 1: Both v and y each contain ≥ 1 kind of symbol

a) $v = a^k$, $y = a^l$

b) $v = a^k$, $y = b^l$

~~c)~~ $v = a^k$, $y = c^l$

Not possible
$\Rightarrow x$ contains $b^p \Rightarrow |vxy| > p$

d) $v = b^k$, $y = b^l$

e) $v = b^k$, $y = c^l$

f) $v = c^k$, $y = c^l$

b) i) $k > 0$ $|w'| = uv^2xy^2z$: # c's = p, # a's > p $\Rightarrow w' \notin L$

ii) $l > 0$ $|w'| = uv^2xy^2z$: # c's = p, # b's > p $\Rightarrow w' \notin L$

Pumping Lemma example

Claim: $L = \{a^n b^n c^n | n \geq 0\}$ is not context-free

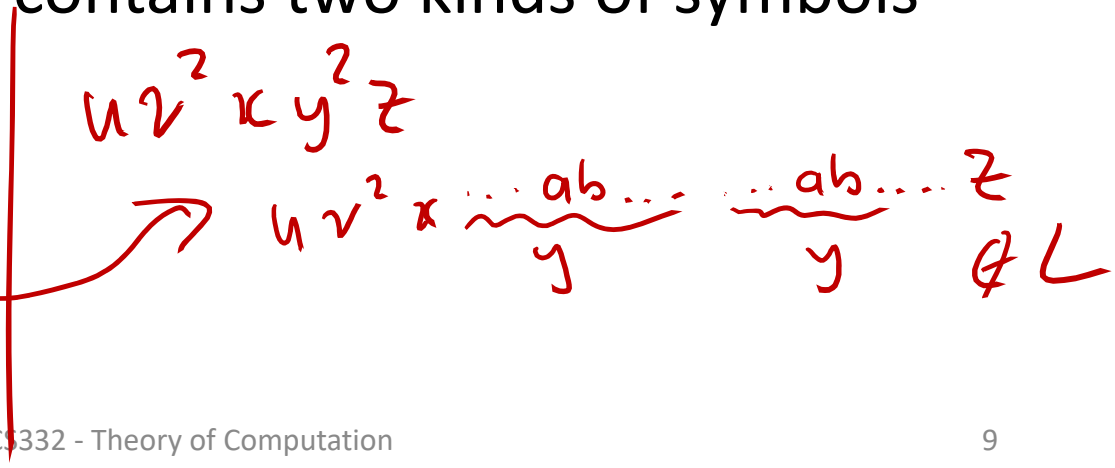
Proof: Assume L is context-free with pumping length p

1. Find $w \in L$ with $|w| \geq p$
2. Show that w cannot be pumped

   If $w = uvxyz$ with $|vy| > 0, |vxy| \leq p$, then…

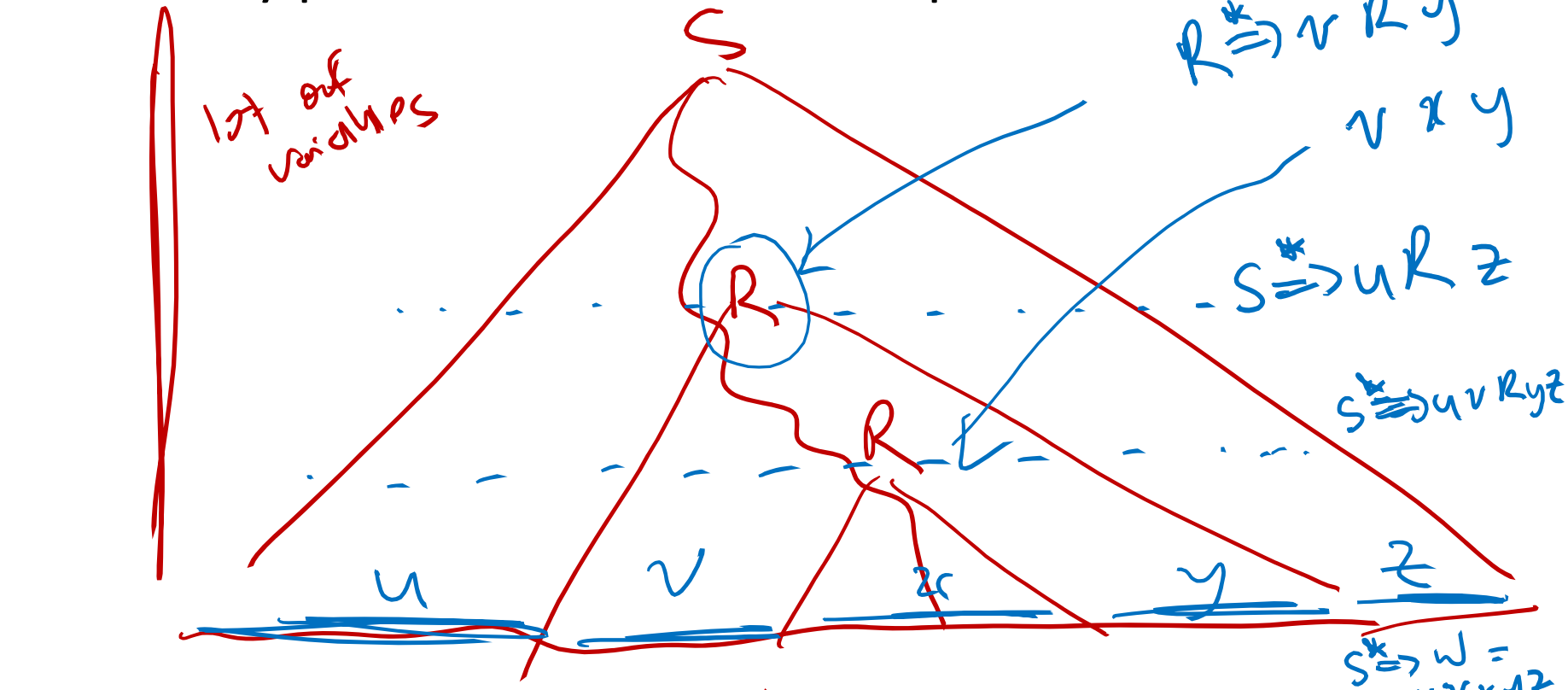
Case 2: Either v or y contains two kinds of symbols

a) v contains ab
b) v contains bc
c) y contains ab
d) y contains bc



Pumping Lemma: Proof idea

Let L be a context-free language. If $w \in L$ is long enough, then every parse tree for w has a repeated variable.



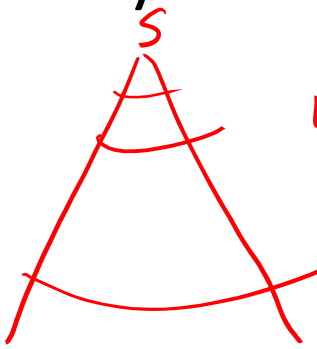
Pumping Lemma Proof

What does "long enough" mean? (How do we choose the pumping length p?)

$R \to AxBCyz \ldots \#S$

$\leq b$ for every rule

- Let G be a CFG for L
- Suppose the right-hand side of every rule in G uses at most b symbols
- Let $p = b^{|V|+1}$



Level 1: width $\leq b$

Level 2: width $\leq b^2$

Level h: width $\leq b^h$

Claim: If $w \in L$ with $|w| \geq p$, then the smallest parse tree for w has height at least $|V| + 1$

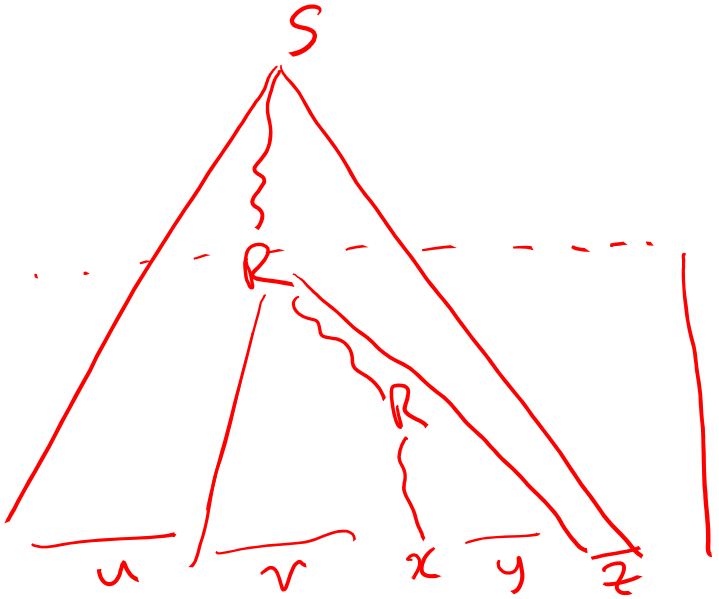
least # of vertices

why? If w had a parse tree of height $\leq |V|$, then $|w| \leq b^{|V|} < p$, a contradiction.

Pumping Lemma Proof

Claim: If $w \in L$ with $|w| \geq p$, then the smallest parse tree for w has height at least $|V| + 1$

- By the pigeonhole principle, there is a path down the parse tree with a repeated variable R
- Choose two such occurrences within the bottom $|V| + 1$ levels



1) $|vy| > 0$ because this is the smallest parse tree. If vy were ϵ, then the parse tree for $w = uxz$ would be smaller
2) $|vxy| \leq p$ because the top R is at most $|V|+1$ levels from the bottom
3) $uv^i x y^i z \in L$ for every i because we can apply the rule $R \Rightarrow vRy$ as many times as we want

Context-Free Languages

L is a *context-free language* if it is the language of some CFG

Questions about CFLs

1. Which languages are *not* context-free?
2. How do we recognize whether $w \in L$?
3. What are the closure properties of CFLs?

Pumping Lemma

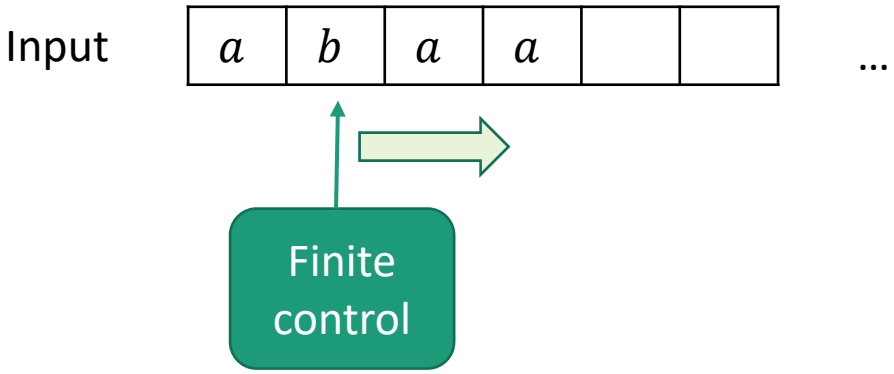
Pushdown Automata (PDA)

Pushdown Automata

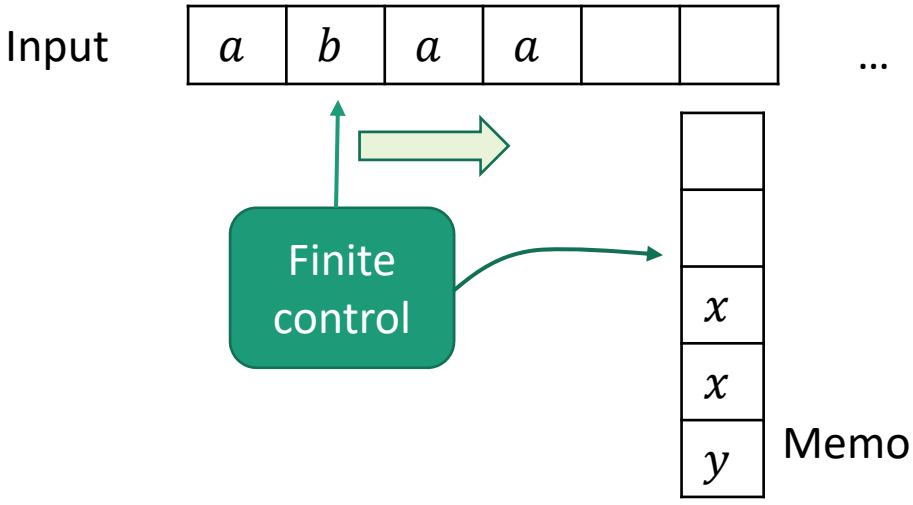
Regular Expressions : Finite Automata ::

Context-Free Languages : ???

Pushdown Automata



Finite Automata (FAs):
Machine with a *finite* amount of unstructured memory

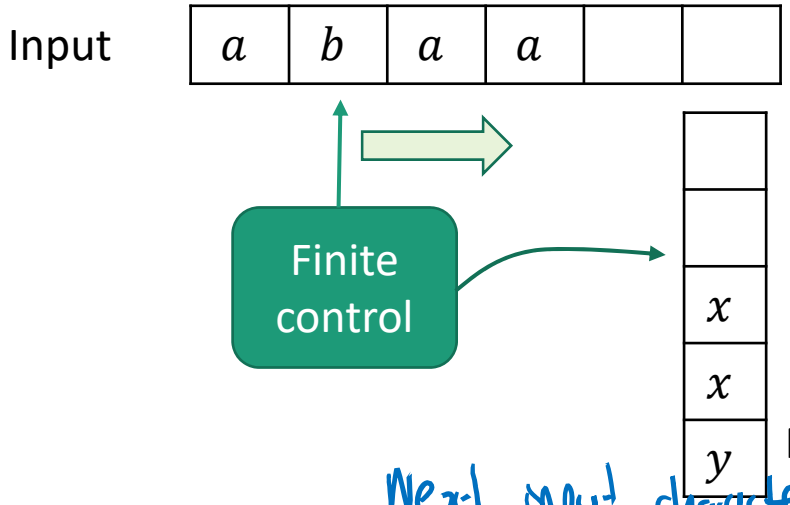


Pushdown Automata (PDAs):
Machine with *unbounded structured* memory in the form of a stack

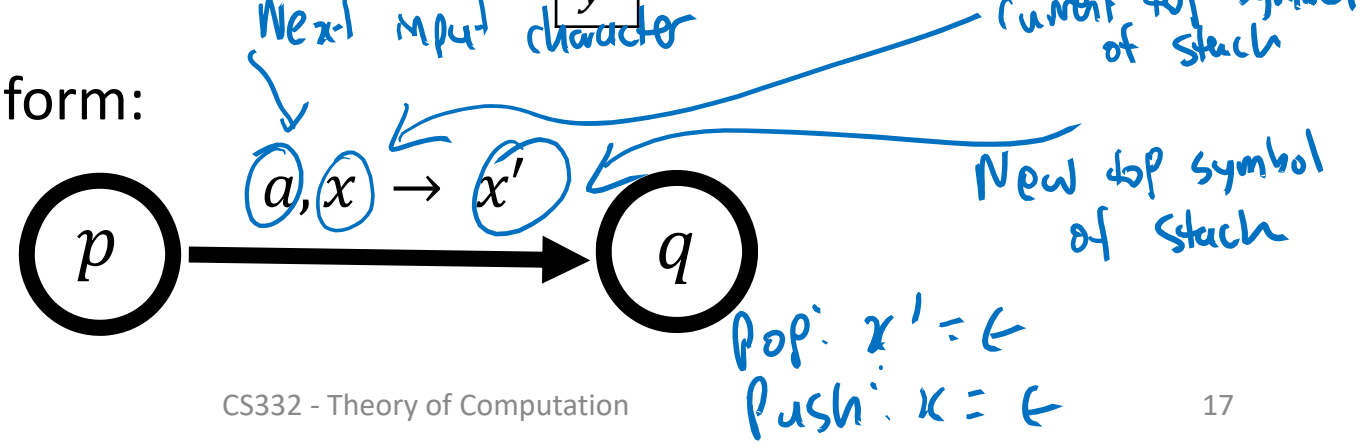
Memory: Infinite Stack

Pushdown Automaton (the idea)

- Nondeterministic finite automaton + stack
- Stack has unlimited size, but machine can only manipulate (push, pop, read) symbol at the top



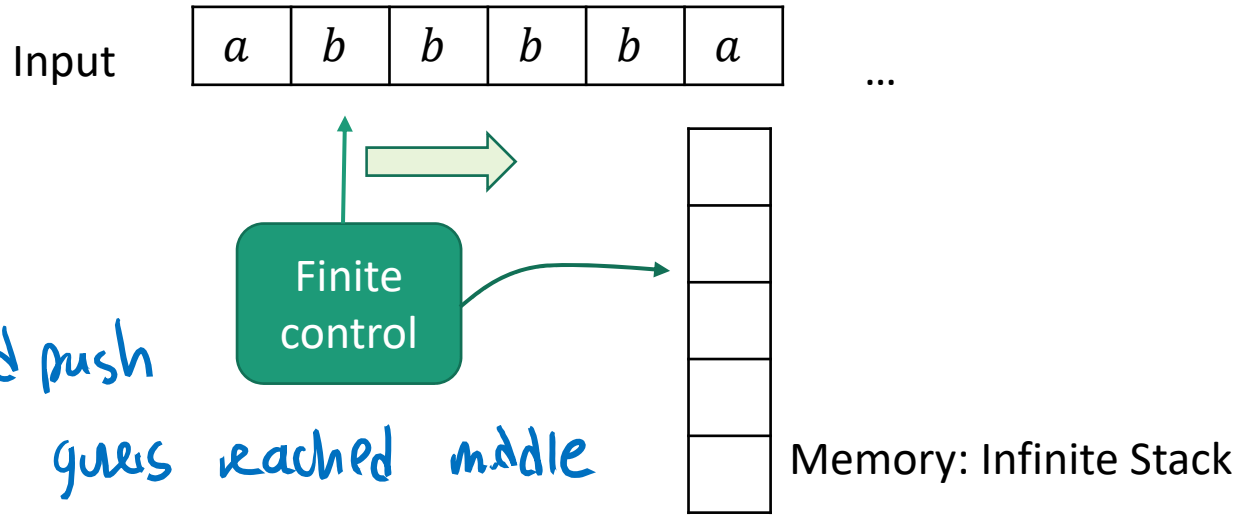
Transitions of the form:



Example: Even Palindromes

HW: All palindromes

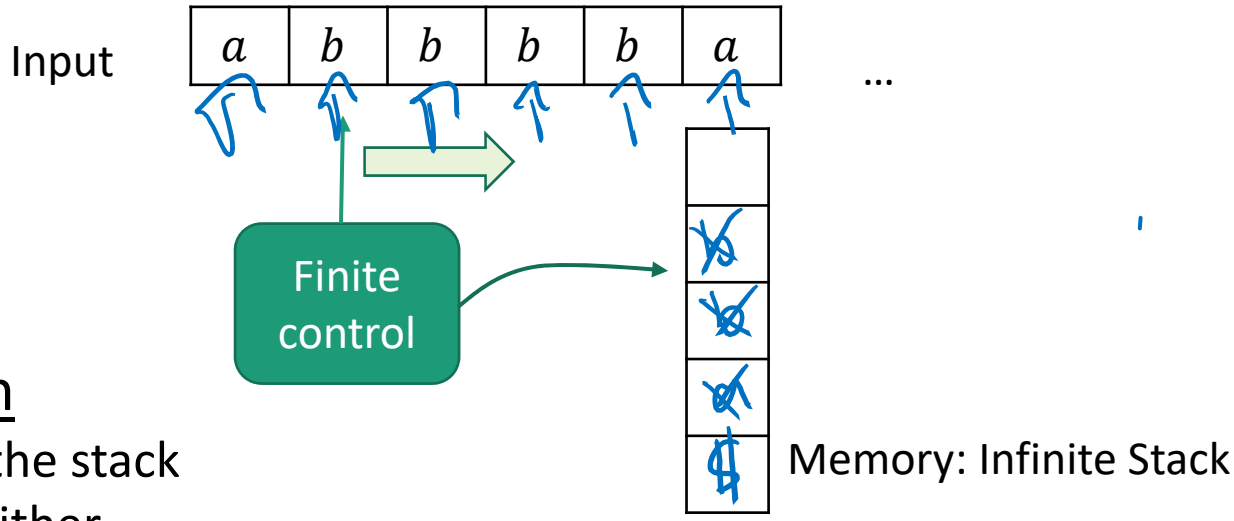
$$L = \{ww^R | w \in \{0,1\}^*\}$$



- Push $
- Repeatedly
  - Read a character and push
  - Non-deterministically guess reached middle & move on
  - Repeatedly read a character and check if it matches top of stack & pop
- Accept if done reading input & stack is empty has $ on top

Example: Even Palindromes

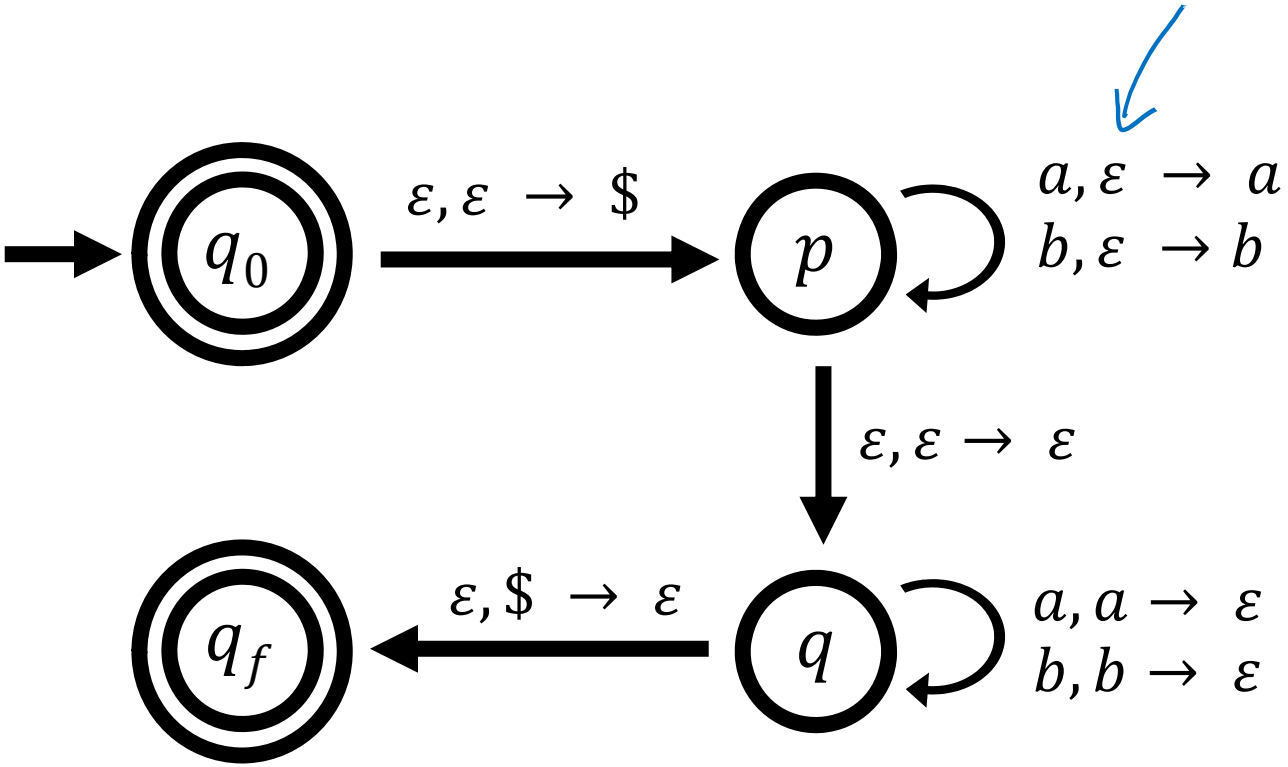
$$L = \{ww^R | w \in \{0,1\}^*\}$$



Algorithmic Description

1. Place the marker $ on the stack
2. Nondeterministically, either
   a) Read a character and push it to the stack, or
   b) Go to the next step
3. Nondeterministically, either
   a) Pop the stack if it matches the next character or
   b) Go to the next step
4. Accept if the top of the stack is $

Example: Even Palindromes



Pushdown Automaton (formal)

A PDA is a 6-tuple (sorry) $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

- Q is a finite set of states
- Σ is the input alphabet
- Γ is the stack alphabet
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$ is the transition function
- q_0 is the start state
- F is the set of final states

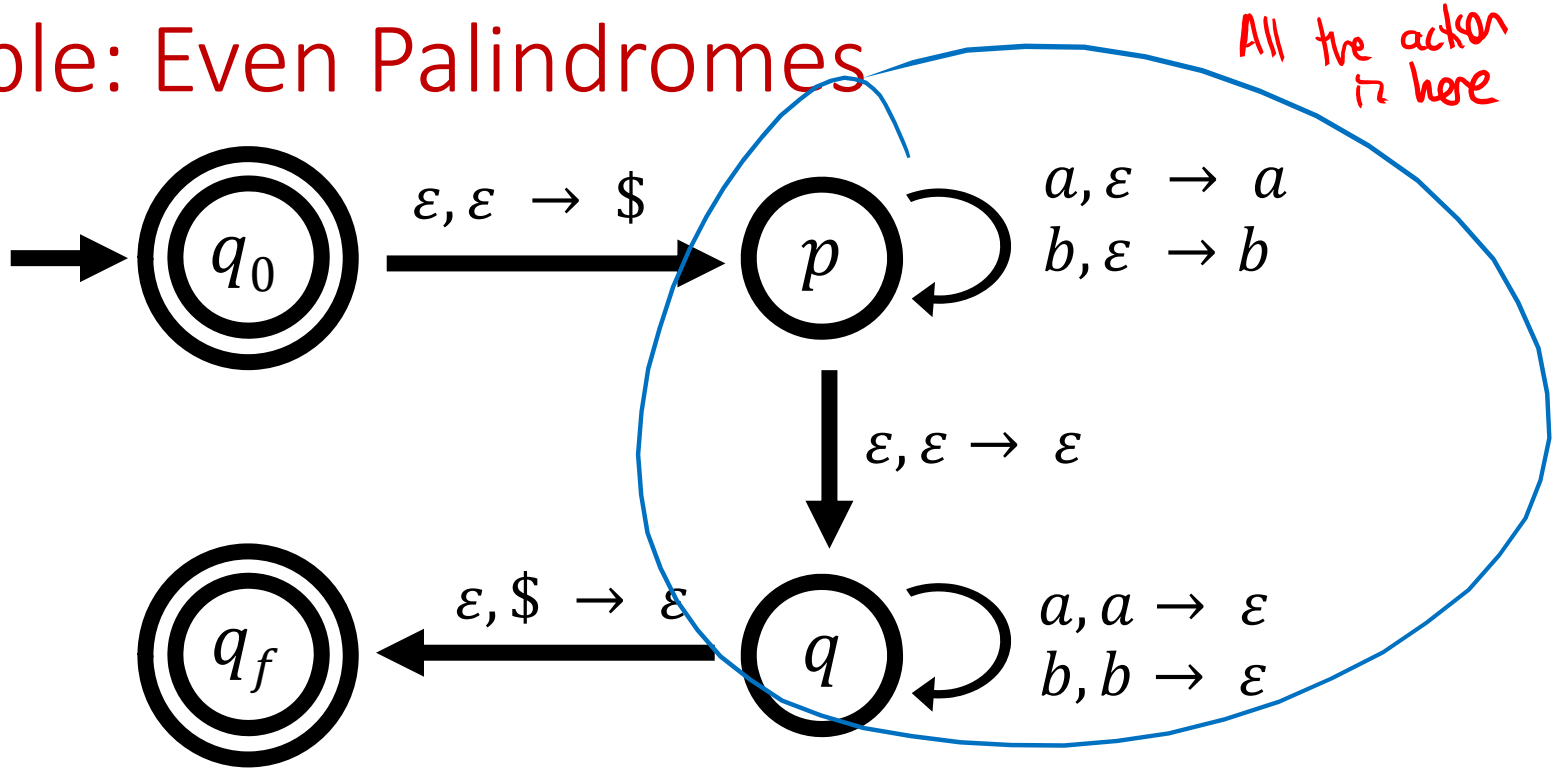
$\delta(q, a, x) = (p, x')$

$q \xrightarrow{a, x \to x'} p$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$, $\Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$

M ***accepts*** a string w if, starting from q_0 and an empty stack, **there exists** a path to an accept state that can be followed by reading all of w.

Example: Even Palindromes



$Q = \{q_0, p, q, q_f\}$

$\Sigma = \{a, b\}$

$\Gamma = \{a, b, \$\}$

$F = \{q_0, q_f\}$

$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow P(Q \times \Gamma_\varepsilon)$

$\delta(p, b, \varepsilon) = \{(p, b)\}$

$\delta(q, a, a) = \{(q, \epsilon)\}$

$\delta(q, a, b) = \phi$