CSC 320
Foundations of Computer Science

Lecture 10

Instructor: Dr. Ulrike Stege

Territory Acknowledgement

We acknowledge and respect the lək̓ʷəŋən peoples on whose traditional territory the university stands and the Songhees, Esquimalt and W̱SÁNEĆ peoples whose historical relationships with the land continue to this day.

This meeting will be recorded

"Please be aware our sessions are being screen-recorded to allow students who are not able to attend to watch later and will be posted in Brightspace."

Deadlines; Assessment

10%

Quizzes

Quiz 1-8: 1% each
Quiz 9: 2%

25%

Assignments

Assignment 1-5: 5% each

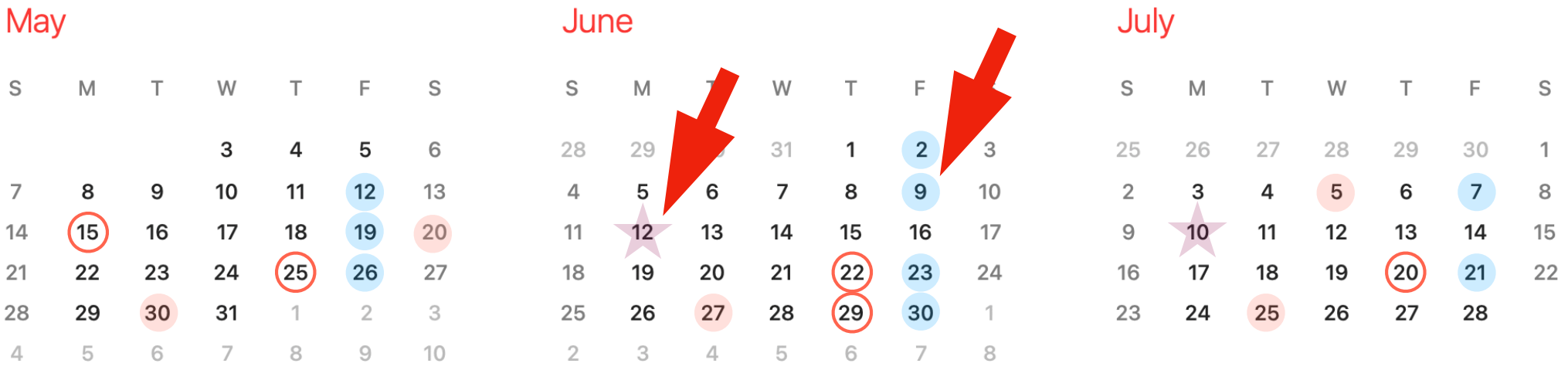
25%

Midterms

Midterm 1: 10%
Midterm 2: 15%

40%

Final Exam



Timed quizzes (~30 min)
Review before starting quiz

Last time ….

- Ambiguous grammars
- Inherently ambiguous languages
- Chomsky Normal Form
- Pushdown Automata

Chomsky Normal Form (CNF)

- Restricted (simplified) constrains on grammar
- A context-free grammar $G = (V, \Sigma, R, S)$ is in **CNF** if every rule is of the form
  - $A \rightarrow BC$ or $A \rightarrow a$ where
    - $a \in \Sigma$
    - $A, B, C \in V$
    - B, C may not be the start variable
    - $S \rightarrow \epsilon$ is permitted where S is start variable

Right hand side: two variables or one terminal; nothing else

Start variable not on right-hand side of rule

No other ϵ-substitutions permitted

Theorem: Any context-free language is generated by a context-free grammar in CNF

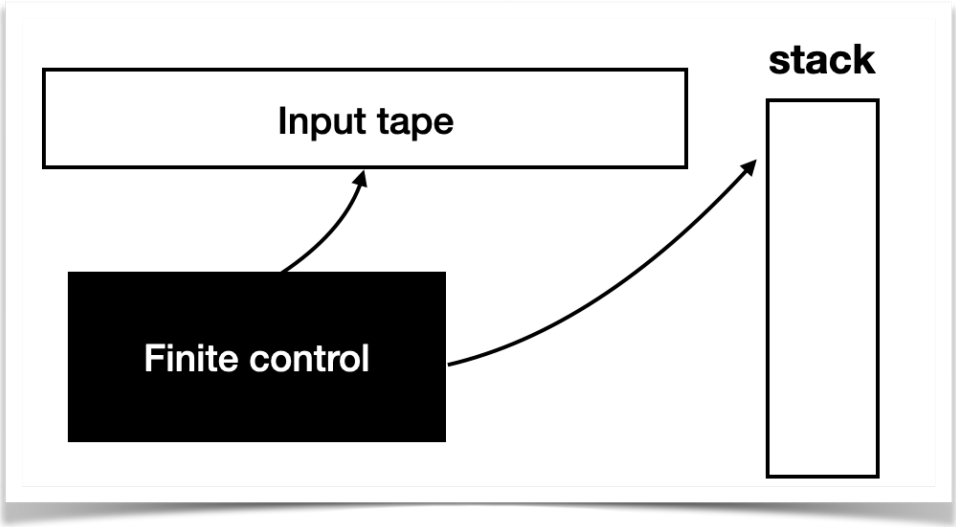
Up next in Context-Free Languages

- Context-free grammars
- Pushdown automata
- The set of languages recognized by pushdown automata is exactly the set of context-free languages

Definition

- A **pushdown automaton (PDA)** is a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ with
  - Q: finite set of states
  - Σ: finite **input alphabet**
  - Γ: finite **stack alphabet**
  - δ: $Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow \mathcal{P}(Q \times \Gamma_\epsilon)$ transition function
  - $q_0 \in Q$: start state
  - $F \subseteq Q$: set of accept states

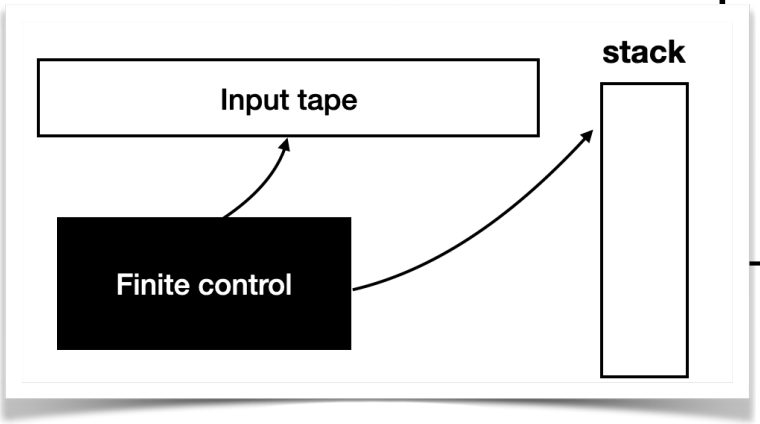
$$\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$$
$$\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$$

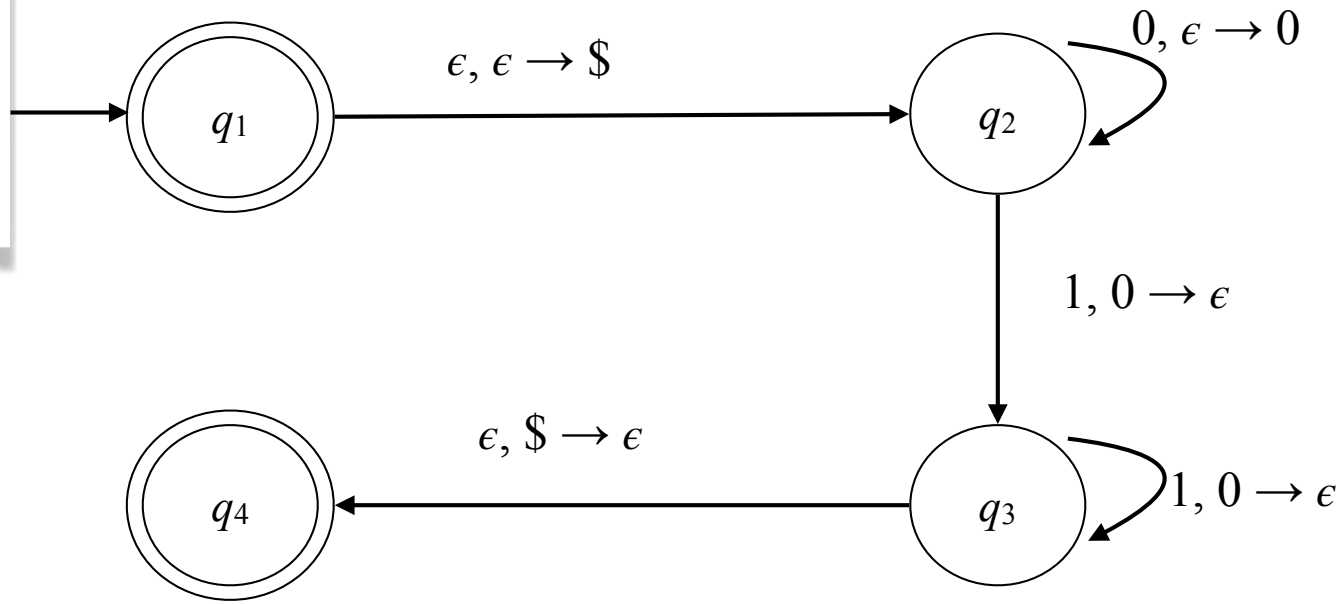


- $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ means: when M is in state r_i reading w_{i+1} from input and top stack symbol is a, then M can do the following: move into state r_{i+1} and replace top stack symbol by b

- If $a = \epsilon$ then top stack symbol is ignored and symbol b is pushed onto stack

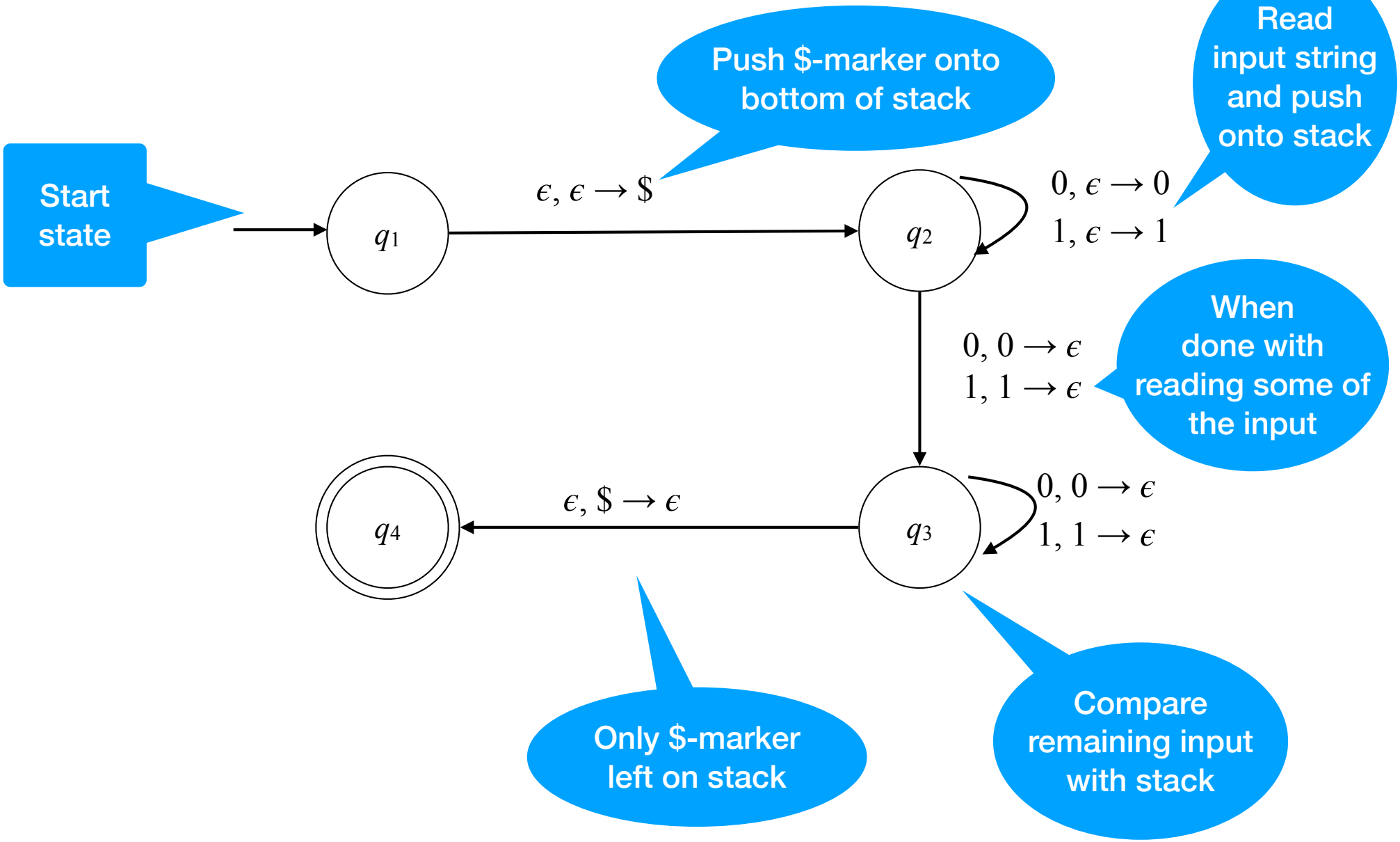
- If $b = \epsilon$ then top stack symbol a is removed from stack





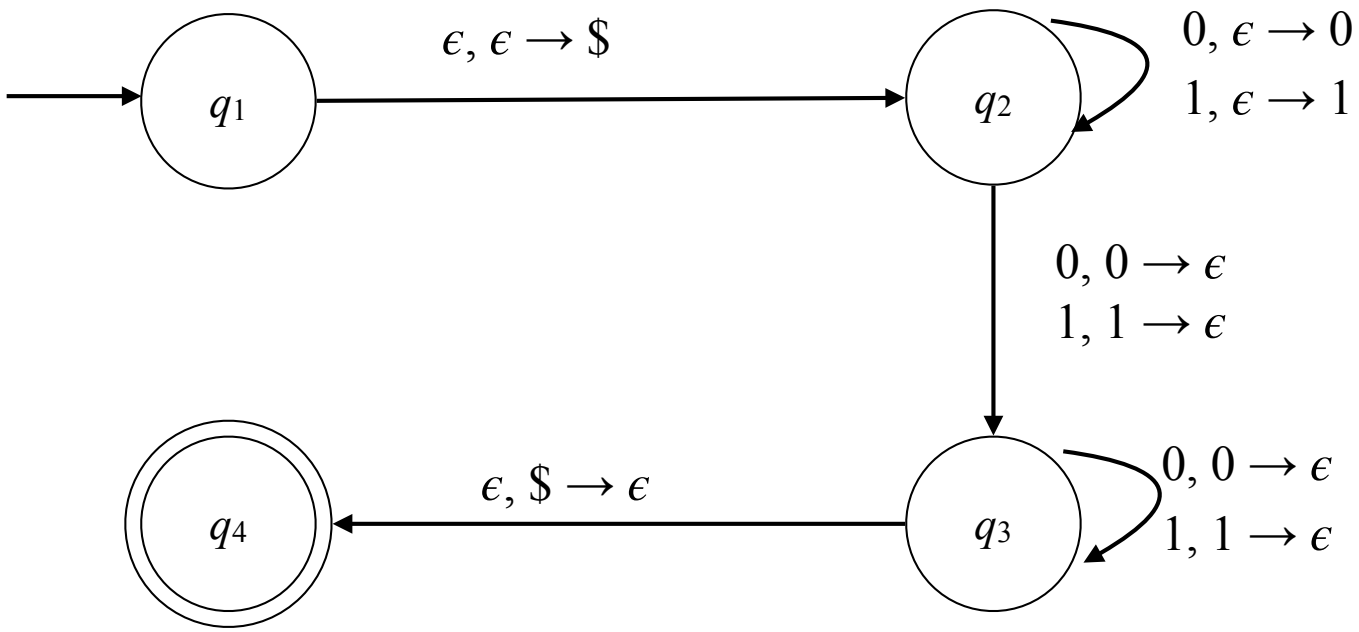
Designing PDA for

$$\{ww^R \mid w \in \{0,1\}^*\} \setminus \{\epsilon\}$$



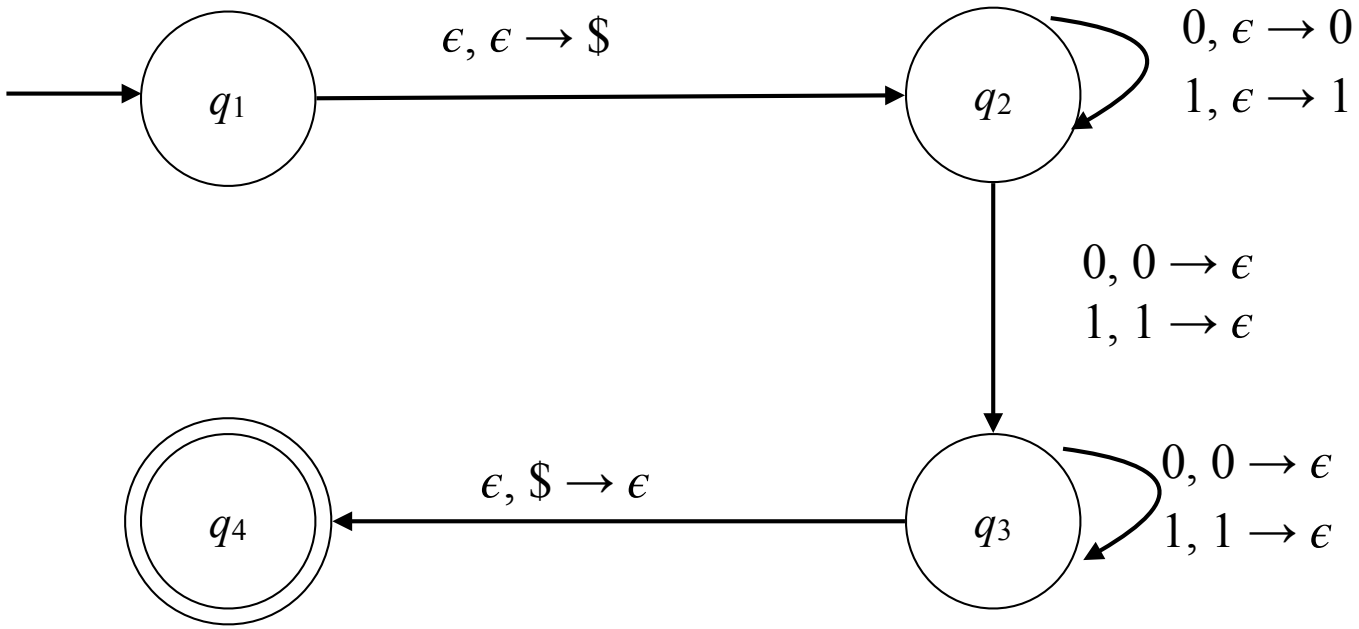
Designing PDA for

$$\{ww^R \mid w \in \{0,1\}^*\} \setminus \{\epsilon\}$$



Note: only strings accepted by the machine are of form ww^R
However: not every possible computation branch will yield acceptance, and every string of form ww^R has accepting branch in computation tree

Your turn



Accepting state sequence of computation for input $w = 10100101$?

A. $q_1\ q_2\ q_2\ q_3\ q_4$

B. $q_1\ q_2\ q_2\ q_2\ q_3\ q_3\ q_3\ q_3\ q_4$

C. $q_1\ q_2\ q_2\ q_2\ q_2\ q_3\ q_3\ q_3\ q_3\ \ q_4$

D. $q_1\ q_2\ q_2\ q_2\ q_2\ q_2\ q_3\ q_3\ q_3\ q_3\ q_3\ q_4$

E. None of the above

1 0 1 0 0 1 0 1
$q_1\ q_2\ \ q_2\ q_2\ q_2\ q_2\ \ q_3\ q_3\ q_3\ q_3\ \ q_4$

0
1
0
1
$

Questions

- Are PDAs nondeterministic?
- Are context-free grammars nondeterministic?
- How can one prove that every regular language is also accepted by a pushdown automaton?

Quiz 4—Question 6

Let $\Sigma = \{0,1\}$, and let L_5 be a language over Σ with $L_5 = \varnothing$. Then for DFA M with $L(M) = L_5$, and for NFA N with $L(N) = L_5$:

- M has at least one accept and one reject state.
- N has at least one accept and one reject state.
- N can be a DFA with $Q = \emptyset$.
- N must have a non-accept state, but cannot have an accept state.
- none of the above

Back to context-free languages, pushdown automata and context-free grammars

Next

Theorem: A language is context-free if and only if some PDA recognizes it

Proof idea
if: Since every context free language L can be produced by a context-free grammar G, $L = L(G)$, convert G into PDA M with $L(M) = L(G) = L$

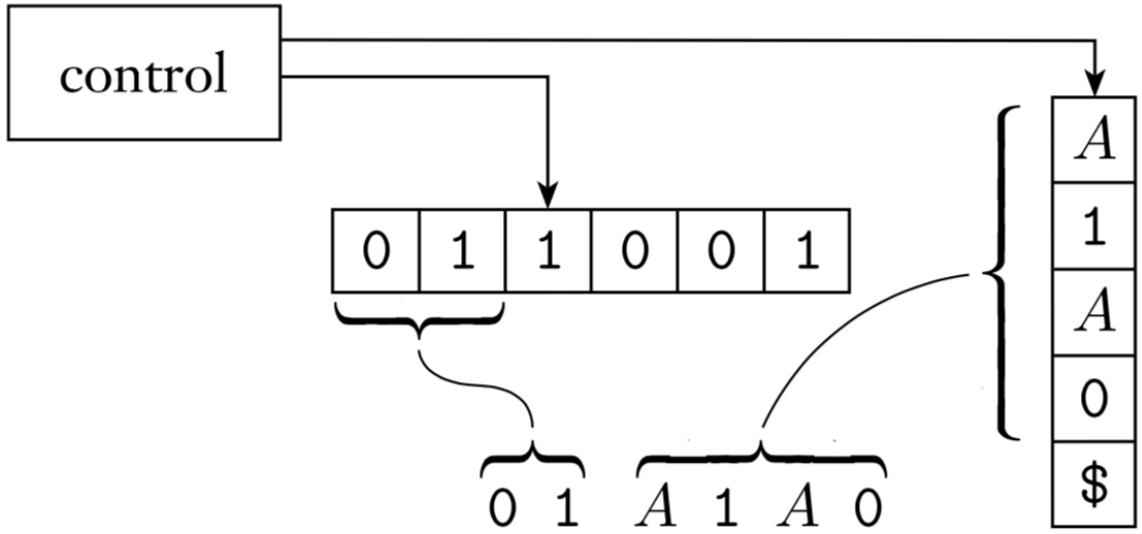
only if: Given pushdown automaton M, create context free grammar G with $L(G) = L(M)$

If: If language is context free then some PDA recognizes it

$G = (V, \Sigma, R, S)$ context-free, $L = L(G)$; design PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

- Idea: if G generates w build M such that it accepts w by determining whether there is a derivation (sequence of substitutions) in G for w
  - Each *step* of derivation yields ***intermediate string*** of variables and terminals

PDA representing derived intermediate string $01A1A0$



Note: PDA simulates G's leftmost derivation

If: If language is context-free then some PDA recognizes it

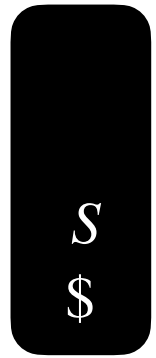
$G = (V, \Sigma, R, S)$ context-free, $L = L(G)$, design PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

- Idea: if G generates w build M such that it accepts w by determining whether there is a derivation (sequence of substitutions) in G for w
  - Each *step* of derivation yields intermediate string of variables and terminals
  - M nondeterministically checks for substitutions from G for w: ie, $S \overset{*}{\Rightarrow} w$
    - First: M puts S on stack
    - Replace top variable symbol by intermediate string
    - Pop top terminal symbol if corresponds to current symbol in w

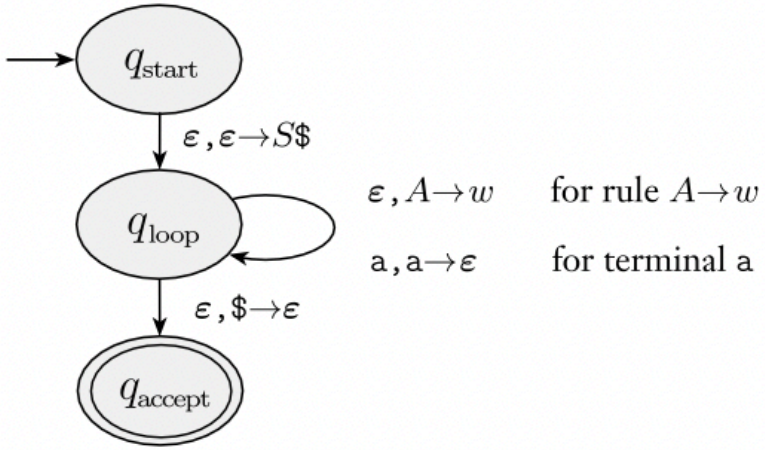
If: If language is context free then some PDA recognizes it

$G = (V, \Sigma, R, S)$ context-free, $L = L(G)$, design PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

- Places marker symbol ($) and start variable S onto (empty) stack



- For each top stack symbol
  - If variable, A, then choose from G some rule
    $A \rightarrow u$, $u = \alpha_1\ \alpha_2\ \ldots\ \alpha_k$, $\alpha_i \in \Gamma$: substitute
    A with $\alpha_1\ \alpha_2\ \ldots\ \alpha_k$ (α_1 is new top symbol)
  - If terminal, a, read next input symbol w_i
    pop top if $w_i = a$; (reject if $w_i \neq a$)
  - If $ go to accept state



Detailed description of M

For $G = (V, \Sigma, R, S)$ context-free design $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ with

- $Q = \{q_{\text{start}}, q_{\text{loop}}, q_{\text{accept}}\} \cup$ set of auxiliary states to push right hands of rules in R onto stack
- $\Gamma = V \cup \Sigma \cup \{\$\}$
- $q_0 = q_{\text{start}}$
- $F = \{q_{\text{accept}}\}$

M's transitions

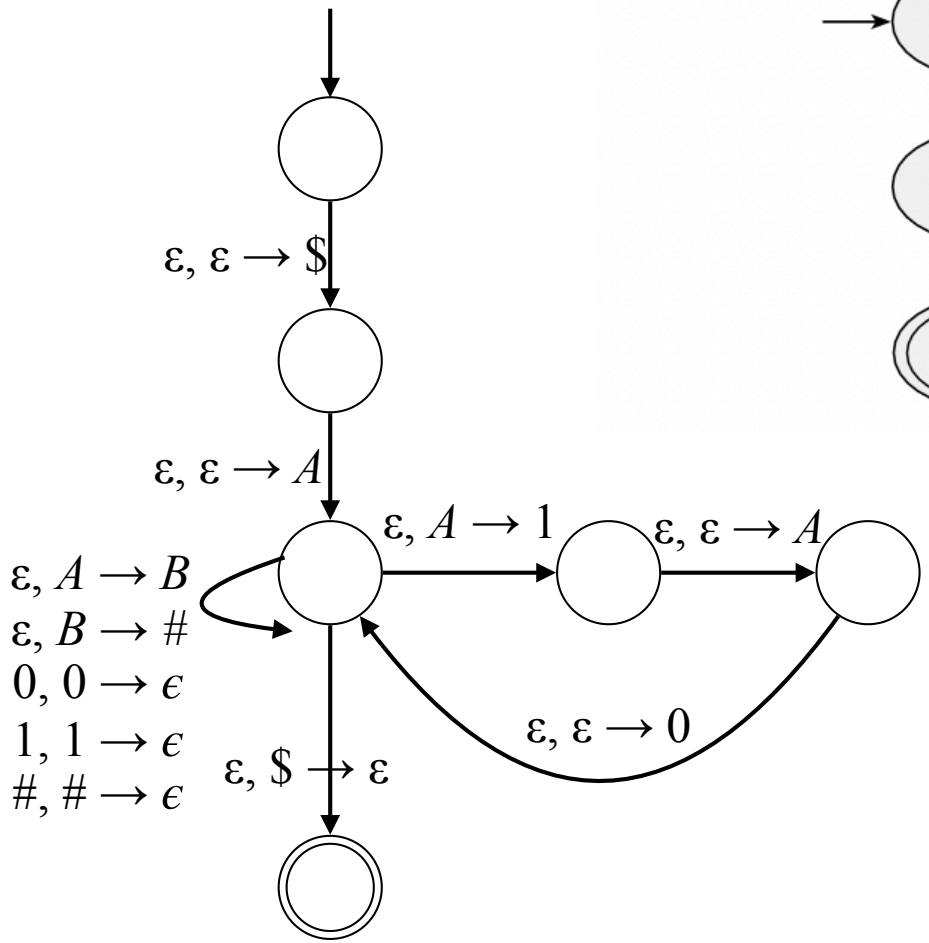
- In q_{start} when reading ε and top symbol ε: push first $\$$ and then S onto stack and move into q_{loop}
- For each rule $A \rightarrow \alpha_1 \alpha_2 \ldots \alpha_k$, in R: In q_{loop} for top stack symbol A: replace A by $\alpha_1 \alpha_2 \ldots \alpha_k$ and remain in q_{loop}
- For each terminal $a \in \Sigma$: If a is top stack symbol then pop a and remain in q_{loop}
- If $\$$ is top stack symbol then pop $\$$ and move into q_{accept}

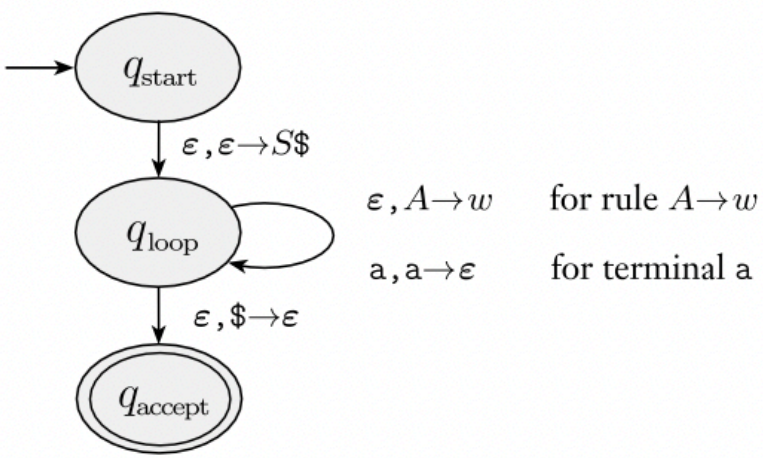
Example

$A \to 0A1$

$A \to B$

$B \to \#$





Context free languages

Theorem: A language is context free if and only if some PDA recognizes it

Proof idea

if: Since every context free language L can be produced by context free grammar G, $L = L(G)$, convert G into PDA M with $L(M) = L(G) = L$

only if: Given pushdown automaton M, create context free grammar G with $L(G) = L(M)$

If a PDA recognizes some language then it is context-free (Proof idea only)

Step 1: Simplify PDA

- Single accept state
- $ always popped exactly before moving into accept state
- Transition: either push or pop, not both at the same time

Step 2: Design grammar

If a PDA recognizes some language then it is context-free (Proof idea only)

Step 1: Simplify PDA M

- **Single accept state**

Add new state, ε-transition (don't read, pop $) from each (original) accept state to new state

Make new state only accept state

- $ always popped exactly before moving into accept state

Only transition from start state: push $ onto empty stack

- **Transition: either push or pop, not both at the same time**

Every transition that replaces top stack symbol replace by two transitions: first one pop's the symbol and a second following directly after pushing the (original) replacement into stack

Step 2: Design grammar

Your turn: Is this language regular? Context-free?

$$B = \{a^n b^n c^n \mid n \geq 0\}$$

Next: Pumping Lemma for context-free languages