CSC 320
Foundations of Computer Science

Lecture 9

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%	25%	25%	40%
Quizzes	Assignments	Midterms	Final Exam
Quiz 1-8: 1% each Quiz 9: 2%	Assignment 1-5: 5% each	Midterm 1: 10% Midterm 2: 15%	

May

S	M	T	W	T	F	S
			3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

June

S	M	T	W	T	F	S
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

July

S	M	T	W	T	F	S
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	

Timed quizzes (~30 min)
Review before starting quiz

Last time ….

- Reductions
- Context-free grammars & context-free languages

Definition: Context-Free Grammars

A **context-free grammar** is a 4-tuple (V, Σ, R, S)

- V: finite set of **variables**
- Σ: finite set of **terminals** (disjoint from V)
- R: finite set of (substitution) **rules**
  - each rule in R: a variable substituted by a string over variables and terminals
- $S \in V$: **start variable**
- The right hand side of a rule may be ϵ

More terminology

- Given grammar $G = (V, \Sigma, R, S)$
- Let u, v, and w be strings of variables and terminals, and
- let $A \rightarrow w$ be a rule of G
- Then
  - uAv **yields** uwv, written $uAv \Rightarrow uwv$
  - u **derives** v, written $u \overset{*}{\Rightarrow} v$, if $u = v$ or if a sequence $u_1, u_2, \ldots, u_k$ exists for $k \geq 0$ and $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$
  - The **language of grammar** G is: $L(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$

The class of languages described by context-free grammars is the class of **context-free languages**

Leftmost derivations

- We call a derivation of string w in grammar G **leftmost derivation** if
    - at every step the *leftmost remaining variable* is replaced

How do we know a grammar describes a language?

- $L(G) = \{1^n 0^n \mid n \geq 0\}$
- $G = (V, \Sigma, R, S)$ with
  - $V = \{S\}$; $\Sigma = \{0,1\}$; R: $S \to 1S0 \mid \epsilon$

- If $w \in L$ then $w \in L(G)$
  - Proof by induction of length of string in language
- If $w \in L(G)$ then $w \in L$
  - Proof by induction of length of derivation

Your turn

- $L = \{w \in \{0,1\}^* \mid w$ has an equal number of 0s and 1s$\}$

- Can you come up a context-free grammar G with $L(G) = L$?

- Possible solution:

$$S \longrightarrow 0S1S \mid 0S1S \mid \epsilon$$

Today: More on Grammars

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

Ambiguous grammars

- A string w is derived **ambiguously** in context-free grammar G if it has at least two *different leftmost derivations*

- Such a grammar is called **ambiguous**

- When parsing ambiguous strings in programming language: there are no unique instructions what code to generate since outcomes of the different instructions can be different!

    - Eg: $a + a \cdot a$ unclear unless PEDMAS is applied additionally

    -
```
if (condition1)
    if (condition2)
        statement1;
    else
        statement2;
```

Example of an ambiguous grammar

- Given $G = (V, \Sigma, R, E)$ with $V = \{E\}, \Sigma = \{a, +, \cdot, (\ ,\ )\}$, and R is given by:
  - $E \longrightarrow E{+}E \mid E{\cdot}E \mid (E) \mid a$

Two leftmost derivations for strings in language of $E \longrightarrow E{+}E \mid E{\cdot}E \mid (E) \mid a$

a+a·a

- $E \Longrightarrow E{+}E \Longrightarrow a{+}E \Longrightarrow a{+}E{\cdot}E \Longrightarrow a{+}a{\cdot}E \Longrightarrow a{+}a{\cdot}a$
- $E \Longrightarrow E{\cdot}E \Longrightarrow E{+}E{\cdot}E \Longrightarrow a{+}E{\cdot}E \Longrightarrow a{+}a{\cdot}E \Longrightarrow a{+}a{\cdot}a$
- $a{+}a{\cdot}a$ is derived ambiguously in G
- Therefore G is ambiguous

Draw the corresponding different parse trees!

Language with ambiguous grammar: For many (but not all!) possible to determine equivalent unambiguous grammar

We learned …

- What context-free grammars are
- What a language of a context-free grammar is
- That ambiguous grammars exists

Next…

- Chomsky Normal Form (CNF)
  - Helps dealing with ambiguity
  - Constraint grammar rules

Chomsky Normal Form

- Restricted (simplified) constrains on grammar
- A context-free grammar $G = (V, \Sigma, R, S)$ is in **Chomsky normal form (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

CNF

- Grammars in Chomsky Normal Form are often unambiguous
- Coming up: We show every context-free language has a grammar in Chomsky Normal Form
- Unfortunately not every context-free language has unambiguous grammar
  - Ie: *inherently ambiguous* context-free languages exist
    - languages that only admits ambiguous grammar
    - **Example**: $L = \{0^m 1^n 2^k \,|\, m = n \text{ or } m = k\}$

Recommended homework: What is a context-free grammar for L?

Theorem

Any context-free language is generated by a context-free grammar in Chomsky normal form (CNF)

Proof. Any context-free language is generated by a context-free grammar in CNF

- **Idea.** Given context free grammar G, convert G into CNF
  - If rule violates CNF condition: replace with equivalent one(s) that satisfy CNF
    - Add new start variable
    - Eliminate all ϵ-*rules* of form $A \rightarrow \epsilon$
    - Eliminate all *unit rules* of form $A \rightarrow B$
    - Convert remaining rules

.

Goal

- Given context-free grammar $G = (V, \Sigma, R, S)$, convert into context-free grammar $G' = (V', \Sigma, R', S_0)$ in CNF with $L(G) = L(G')$

Step 1. Add new start variable

- Let $S_0 \notin V$
- Add new start variable S_0 and rule $S_0 \to S$
  - Thus: Start variable in G' not on right-hand side of rule

Step 2. Eliminate all ϵ-*rules* of form $A \to \epsilon$

Repeat until *all* ϵ-rules *not* involving S_0 are eliminated

- Given $A \to \epsilon$, $A \neq S_0$
- For each $W \to uAv$ and occurrence of A, with u, v strings of variables and terminals
  - add new rule $W \to uv$
  - For $W \to A$, add $W \to \epsilon$ unless $W \to \epsilon$ was previously removed
- Remove $A \to \epsilon$

Step 3. Eliminate *unit rules*

- Repeat until all unit rules are eliminated
  - Given $A \to B$
    - For each appearance of $B \to u$ add $A \to u$ (unless this rule was removed previously)
    - As before, u is a string of variables and terminals
    - Remove $A \to B$

Step 4. Convert remaining rules

- Replace each rule $A \rightarrow u_1u_2 \cdots u_k$, where $k \geq 3$ and each u_i is a variable or terminal symbol, with
  - $A \rightarrow u_1A_1$, $A_1 \rightarrow u_2 A_2$, $A_2 \rightarrow u_3A_3$, ... , and $A_{k-2} \rightarrow u_{k-1}u_k$
    The A_i's are new variables
- Replace any terminal u_i in the preceding rule(s) with new variable U_i and add rule $U_i \rightarrow u_i$

Example
Transforming G into CNF grammar G'

- Given grammar G with
    - $S \longrightarrow ASA \mid aB$
    - $A \longrightarrow B \mid S$
    - $B \longrightarrow b \mid \epsilon$

CNF: every rule of 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

Step 1: Add new start symbol

- $S \to ASA \mid aB$
- $A \to B \mid S$
- $B \to b \mid \epsilon$

- $\boldsymbol{S_0 \to S}$
- $S \to ASA \mid aB$
- $A \to B \mid S$
- $B \to b \mid \epsilon$

Step 2: Eliminate ϵ-*rules*

- $S_0 \rightarrow S$
- $S \rightarrow ASA \mid aB$
- $A \rightarrow B \mid S$
- $\underline{\boldsymbol{B}} \rightarrow b \mid \underline{\boldsymbol{\varepsilon}}$

- $S_0 \rightarrow S$
- $\underline{S} \rightarrow ASA \mid \underline{aB}$
- $\underline{A} \rightarrow \underline{B} \mid S$
- $\underline{B} \rightarrow b \mid \underline{\varepsilon}$

For each $W \rightarrow uXv$ and occurrence of X: add $W \rightarrow uv$
For $W \rightarrow X$: add $W \rightarrow \epsilon$ unless $W \rightarrow \epsilon$ was removed previously

Step 2: Eliminate ϵ-*rules* ($X \longrightarrow \epsilon$)

- $S_0 \to S$
- $S \to ASA \mid \underline{aB}$
- $A \to \underline{B} \mid S$
- $\underline{B} \to b \mid \underline{\varepsilon}$

- $S_0 \to S$
- $S \to ASA \mid \underline{aB} \mid \boldsymbol{a}$
- $A \to \underline{B} \mid S \mid \epsilon$
- $B \to b \cancel{\mid \varepsilon}$

For each $W \to uXv$ and occurrence of X: add $W \to uv$
For $W \to X$: add $W \to \varepsilon$ unless $W \to \varepsilon$ was removed previously

Step 2: Eliminate ϵ-*rules* ($X \longrightarrow \epsilon$)

Summary: Elimination of $B \longrightarrow \epsilon$

- $S_0 \rightarrow S$
- $S \rightarrow ASA \mid \underline{aB}$
- $A \rightarrow \underline{B} \mid S$
- $\underline{B} \rightarrow b \mid \underline{\varepsilon}$

- $S_0 \rightarrow S$
- $S \rightarrow ASA \mid aB \mid a$
- $A \rightarrow B \mid S \mid \epsilon$
- $B \rightarrow b$

For each $W \rightarrow uXv$ and occurrence of X: add $W \rightarrow uv$

For $W \rightarrow X$: add $W \rightarrow \epsilon$ unless $W \rightarrow \epsilon$ was removed previously

Step 2: Eliminate ϵ-*rules* ($X \longrightarrow \epsilon$)

Previously removed: $B \longrightarrow \epsilon$

- $S_0 \rightarrow S$
- $S \rightarrow \underline{A}S\underline{A} \mid aB \mid a$
- $\underline{\boldsymbol{A}} \rightarrow B \mid S \mid \underline{\boldsymbol{\varepsilon}}$
- $B \rightarrow b$

- $S_0 \rightarrow S$
- $S \rightarrow \underline{ASA} \mid aB \mid a \mid \boldsymbol{SA} \mid \boldsymbol{AS} \mid \boldsymbol{S}$
- $\underline{A} \rightarrow B \mid S \cancel{\mid \varepsilon}$
- $B \rightarrow b$

For each $W \rightarrow uXv$ and occurrence of X: add $W \rightarrow uv$
For $W \rightarrow X$: add $W \rightarrow \epsilon$ unless $W \rightarrow \epsilon$ was removed previously

Step 2: Eliminate ϵ-*rules* ($X \longrightarrow \epsilon$)

Previously removed: $B \longrightarrow \epsilon$

- $S_0 \rightarrow S$
- $S \rightarrow ASA \mid aB \mid a$
- $\underline{A} \rightarrow B \mid S \mid \underline{\varepsilon}$
- $B \rightarrow b$

- $S_0 \rightarrow S$
- $S \rightarrow \underline{ASA} \mid aB \mid a \mid \boldsymbol{SA} \mid \boldsymbol{AS} \mid \cancel{\boldsymbol{S}}$
- $\underline{A} \rightarrow B \mid S \cancel{\mid \varepsilon}$
- $B \rightarrow b$

For each $W \rightarrow uXv$ and occurrence of X: add $W \rightarrow uv$
For $W \rightarrow X$: add $W \rightarrow \epsilon$ unless $W \rightarrow \epsilon$ was removed previously

Step 2: Eliminate ϵ-*rules* ($X \longrightarrow \epsilon$)

Previously removed:
$B \longrightarrow \epsilon$, $A \longrightarrow \epsilon$

- $S_0 \rightarrow S$
- $S \rightarrow \underline{ASA} \mid aB \mid a$
- $\underline{A} \rightarrow B \mid S \mid \underline{\varepsilon}$
- $B \rightarrow b$

- $S_0 \rightarrow S$
- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $A \rightarrow B \mid S$
- $B \rightarrow b$

Step 3: Eliminate unit rules

Previously removed:
$B \longrightarrow \epsilon, A \longrightarrow \epsilon$

- $\underline{S_0 \rightarrow S}$
- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $A \rightarrow B \mid S$
- $B \rightarrow b$

Given $X \rightarrow Y$: for each appearance of $Y \rightarrow u$ add $X \rightarrow u$, unless previously removed

Step 3: Eliminate unit rules

Previously removed:
$B \longrightarrow \epsilon$, $A \longrightarrow \epsilon$

- $\underline{S_0 \rightarrow S}$
- $\underline{S} \rightarrow \underline{ASA} \mid \underline{aB} \mid \underline{a} \mid \underline{SA} \mid \underline{AS}$
- $A \rightarrow B \mid S$
- $B \rightarrow b$

- $S_0 \rightarrow \cancel{S \mid}\ \boldsymbol{ASA} \mid \boldsymbol{aB} \mid \boldsymbol{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS}$
- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $A \rightarrow B \mid S$
- $B \rightarrow b$

Given $X \rightarrow Y$: for each appearance of $Y \rightarrow u$ add $X \rightarrow u$, unless previously removed

Step 3: Eliminate unit rules

Previously removed:
$B \longrightarrow \epsilon$, $A \longrightarrow \epsilon$,
$S_0 \longrightarrow S$

- $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $\underline{A} \rightarrow \underline{B} \mid S$
- $B \rightarrow b$

Given $X \rightarrow Y$: for each appearance of $Y \rightarrow u$ add $X \rightarrow u$, unless previously removed

Step 3: Eliminate unit rules

Previously removed:
$B \longrightarrow \epsilon$, $A \longrightarrow \epsilon$,
$S_0 \longrightarrow S$

- $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $\underline{A} \rightarrow \underline{B} \mid S$
- $\underline{B} \rightarrow \underline{b}$

- $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $A \rightarrow \cancel{B} \mid S \mid \boldsymbol{b}$
- $B \rightarrow b$

Given $X \rightarrow Y$: for each appearance of $Y \rightarrow u$ add $X \rightarrow u$, unless previously removed

Step 3: Eliminate unit rules

Previously removed:
$B \longrightarrow \epsilon$, $A \longrightarrow \epsilon$, $S_0 \longrightarrow S$, $A \longrightarrow B$

- $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $A \rightarrow B \mid S$
- $B \rightarrow b$

- $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $A \rightarrow S \mid b$
- $B \rightarrow b$

Given $X \rightarrow Y$: for each appearance of $Y \rightarrow u$ add $X \rightarrow u$, unless previously removed

Step 3: Eliminate unit rules

Previously removed:
$B \longrightarrow \epsilon$, $A \longrightarrow \epsilon$, $S_0 \longrightarrow S$,
$A \longrightarrow B$

- $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $\underline{A} \rightarrow \underline{S} \mid b$
- $B \rightarrow b$

Given $X \rightarrow Y$: for each appearance of $Y \rightarrow u$ add $X \rightarrow u$, unless previously removed

Step 3: Eliminate unit rules

Previously removed:
$B \longrightarrow \epsilon$, $A \longrightarrow \epsilon$, $S_0 \longrightarrow S$,
$A \longrightarrow B$

- $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $\underline{S} \rightarrow \underline{ASA} \mid \underline{aB} \mid \underline{a} \mid \underline{SA} \mid \underline{AS}$
- $\underline{A} \rightarrow \underline{S} \mid b$
- $B \rightarrow b$

- $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $\underline{S} \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $\underline{A} \rightarrow \cancel{S \mid} \, b \mid \boldsymbol{ASA} \mid \boldsymbol{aB} \mid \boldsymbol{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS}$
- $B \rightarrow b$

Given $X \rightarrow Y$: for each appearance of $Y \rightarrow u$ add $X \rightarrow u$, unless previously removed

Step 3: Eliminate unit rules

Previously removed: $B \longrightarrow \epsilon$, $A \longrightarrow \epsilon$, $S_0 \longrightarrow S$, $A \longrightarrow B$, $A \longrightarrow S$

- $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $A \rightarrow S \mid b$
- $B \rightarrow b$

- $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
- $\underline{S} \rightarrow \underline{ASA} \mid \underline{aB} \mid a \mid \underline{SA} \mid \underline{AS}$
- $A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS$
- $B \rightarrow b$

Given $A \rightarrow S$: for each appearance of $S \rightarrow u$ add $A \rightarrow u$, unless previously removed

Step 4: Convert remaining rules

- $\underline{S_0} \rightarrow \underline{ASA} \mid aB \mid a \mid SA \mid AS$

- $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

- $A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS$

- $B \rightarrow b$

- $S_0 \rightarrow A\underline{SA} \mid aB \mid a \mid SA \mid AS$

- $S \rightarrow A\underline{SA} \mid aB \mid a \mid SA \mid AS$

- $A \rightarrow b \mid A\underline{SA} \mid aB \mid a \mid SA \mid AS$

- $B \rightarrow b$

Step 4. Convert remaining rules

- Replace each rule $A \rightarrow u_1u_2\cdots u_k$, where $k \geq 3$ and each u_i is a variable or terminal symbol, with
  - $A \rightarrow u_1A_1$, $A_1 \rightarrow u_2\, A_2$, $A_2 \rightarrow u_3A_3$, ... , and $A_{k\text{-}2} \rightarrow u_{k\text{-}1}u_k$
    The A_i's are new variables.
- Replace any terminal u_i in the preceding rule(s) with new variable U_i and add rule $U_i \rightarrow u_i$

Step 4: Convert remaining rules

- $\underline{S_0} \rightarrow A\underline{SA} \mid aB \mid a \mid SA \mid AS$
- $S \rightarrow A\underline{SA} \mid aB \mid a \mid SA \mid AS$
- $A \rightarrow b \mid A\underline{SA} \mid aB \mid a \mid SA \mid AS$
- $B \rightarrow b$

- $S_0 \rightarrow A\boldsymbol{A_1} \mid aB \mid a \mid SA \mid AS$
- $S \rightarrow A\boldsymbol{A_1} \mid aB \mid a \mid SA \mid AS$
- $A \rightarrow b \mid A\boldsymbol{A_1} \mid aB \mid a \mid SA \mid AS$
- $B \rightarrow b$
- $\boldsymbol{A_1} \rightarrow \boldsymbol{SA}$

Step 4: Convert remaining rules

- $S_0 \rightarrow AA_1 \mid \underline{a}B \mid a \mid A_1 \mid AS$
- $S \rightarrow AA_1 \mid \underline{a}B \mid a \mid A_1 \mid AS$
- $A \rightarrow b \mid AA_1 \mid \underline{a}B \mid a \mid A_1 \mid AS$
- $B \rightarrow b$
- $A_1 \rightarrow SA$

- $S_0 \rightarrow AA_1 \mid \boldsymbol{U}B \mid a \mid SA \mid AS$
- $S \rightarrow AA_1 \mid \boldsymbol{U}B \mid a \mid SA \mid AS$
- $A \rightarrow b \mid AA_1 \mid \boldsymbol{U}B \mid a \mid SA \mid AS$
- $B \rightarrow b$
- $A_1 \rightarrow SA$
- $\boldsymbol{U} \rightarrow \boldsymbol{a}$

Chomsky Normal Form

R $\longrightarrow$ **R'**

- $S \rightarrow ASA \mid aB$
- $A \rightarrow B \mid S$
- $B \rightarrow b \mid \varepsilon$

CNF: every rule of 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

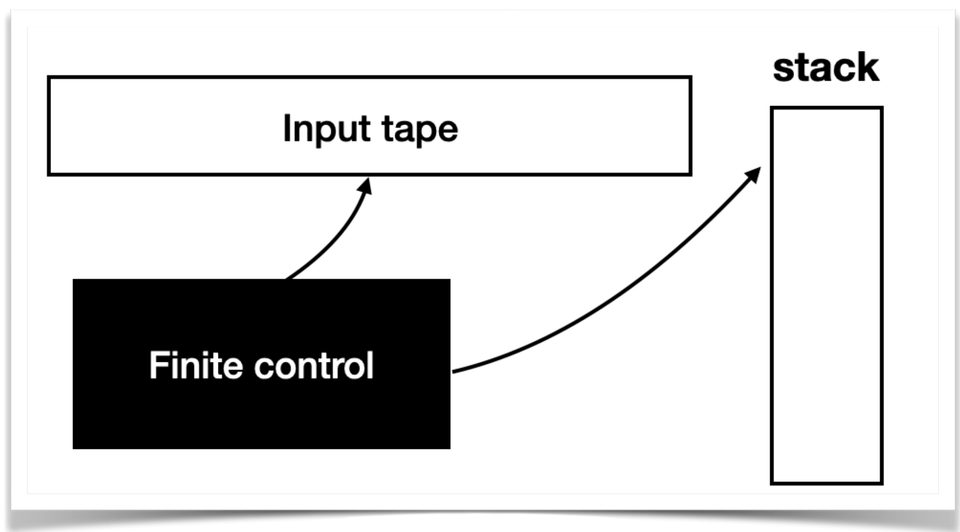
- $S_0 \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$
- $S \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$
- $A \rightarrow b \mid AA_1 \mid UB \mid a \mid SA \mid AS$
- $B \rightarrow b$
- $A_1 \rightarrow SA$
- $U \rightarrow a$

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

Pushdown Automata

- Think of: nondeterministic finite automaton with addition of **stack**
- Stack: provides additional memory
- We will show: languages recognized by pushdown automata are exactly the context free-languages



Definition

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

$$\boxed{\begin{aligned}\Sigma_\epsilon &= \Sigma \cup \{\epsilon\} \\ \Gamma_\epsilon &= \Gamma \cup \{\epsilon\}\end{aligned}}$$

Computation of PDA

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA. Then M **accepts** input $w \in \Sigma^*$ if w can be written as $w = w_1 w_2 \ldots w_m$, $|w| \leq m$, where: $w_i \in \Sigma_\varepsilon$ and there exist states $r_0, r_1, \ldots, r_m \in Q$ and strings $s_0, s_1, \ldots, s_m \in \Gamma^*$ such that

Each s_i: sequence of stack contents that M has on accepting branch (of computation)

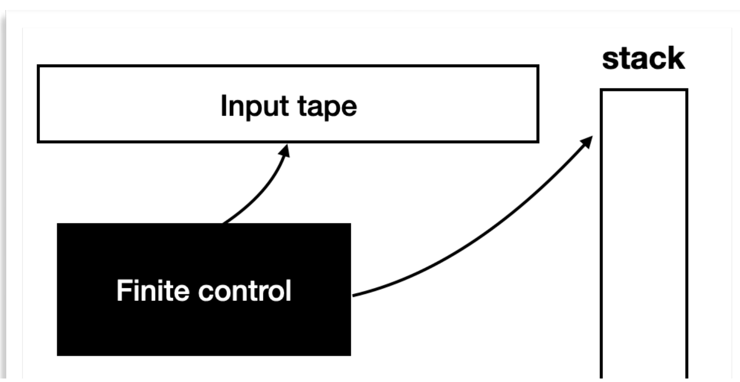
- $r_0 = q_0$ and $s_0 = \epsilon$

M starts computation with empty stack

- for $i = 0, \ldots, m\text{-}1$: $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ with $s_i = at$, $s_{i+1} = bt$, $a, b \in \Gamma_\epsilon$ and $t \in \Gamma^*$

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

- $r_m \in F$

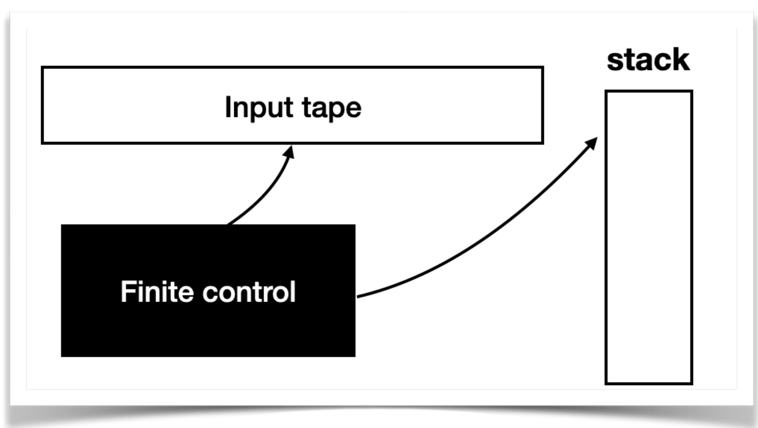


Note

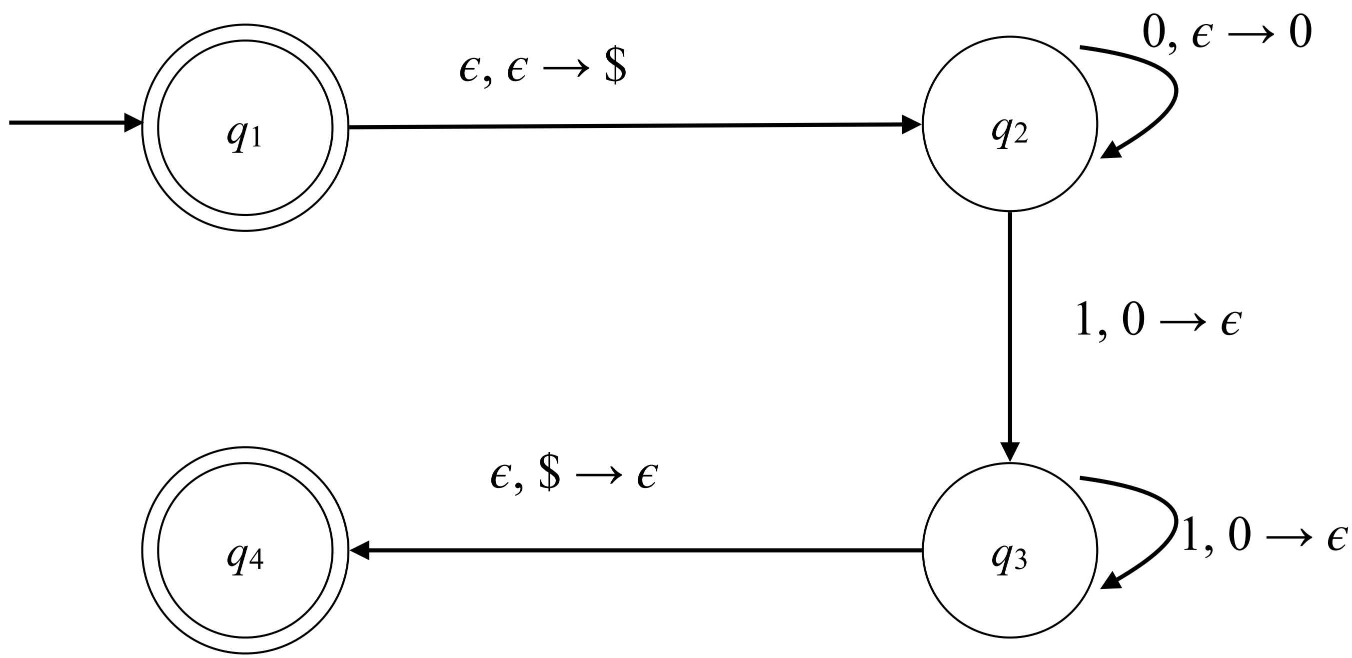
- $(\boldsymbol{r}_{\mathbf{i+1}}, \boldsymbol{b}) \in \delta(\boldsymbol{r}_{\boldsymbol{i}}, \boldsymbol{w}_{\boldsymbol{i}+\mathbf{1}}, \boldsymbol{a})$ means: when M is in state r_i reading w_{i+1} from input and top stack symbol is a, then M can do the following: move into state $r_{\mathrm{i}+1}$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

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

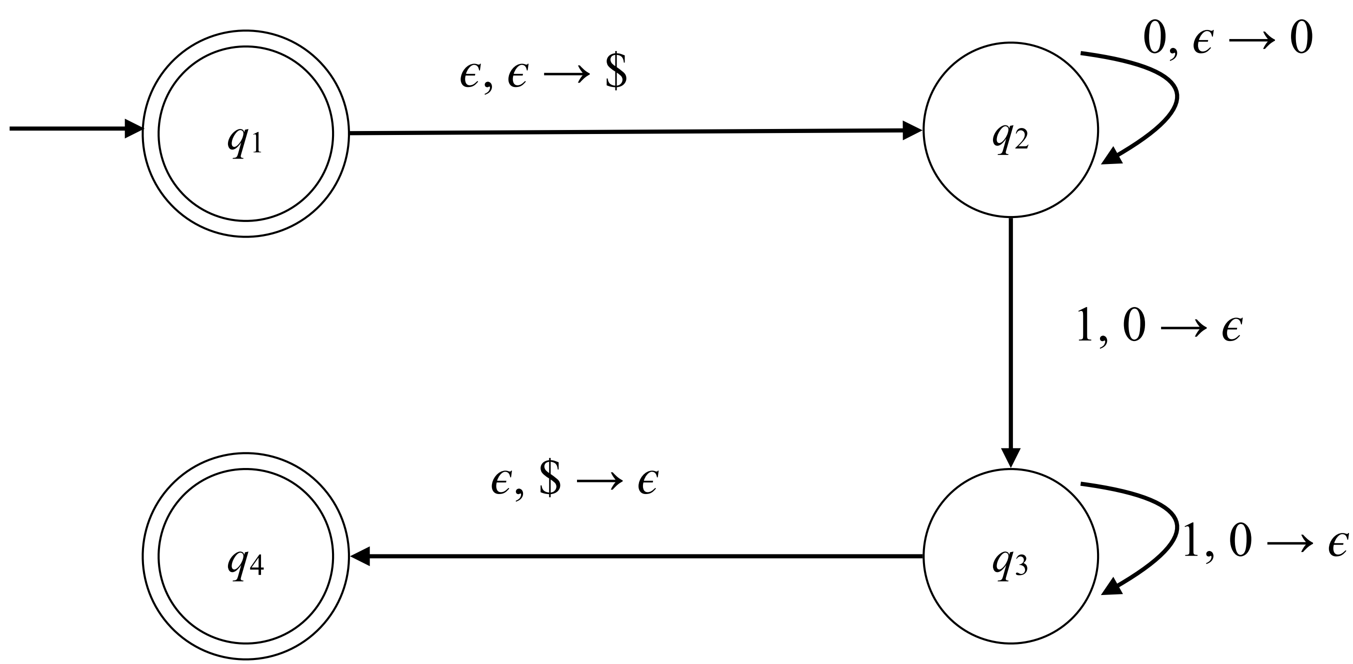
- If $b = \epsilon$ then top stack symbol a is removed from stack



Example: state diagram representation of PDA



$\Sigma = \{0,1\}$, $\mathbf{\Gamma} = \{0, \$\}$



$\Sigma = \{0,1\}$, $\boldsymbol{\Gamma} = \{0, \$\}$

Inputs: $w = 0011$

Computation reading $w = 0011$:

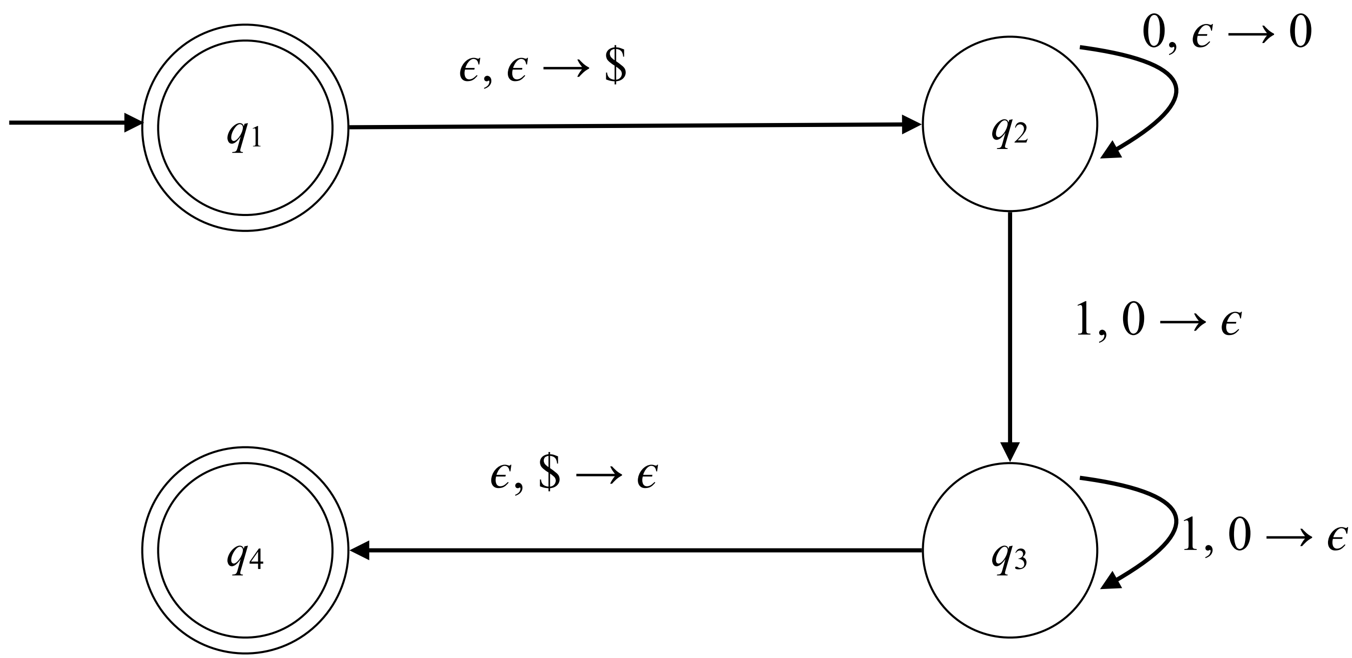
- In q_1 reading no input symbol and ignoring stack content, move to q_2 and push symbol \$ onto stack
- In q_2 reading first input symbol 0 and ignoring stack content, remain in q_2 and push symbol 0 onto stack
- In q_2 reading second input symbol 0 and ignoring stack content, remain in q_2 and push symbol 0 onto stack
- In q_2 reading third input symbol 1 and while 0 is top stack symbol, move to q_3 and pop symbol 0 from stack
- In q_3 reading fourth input symbol 1 and while 0 is top stack symbol, remain in q_3 and pop symbol 0 from stack
- In q_3 reading no input symbol and while \$ is top stack symbol, move to q_4 and pop symbol \$ from stack

Possible computations for input s

Is $s \in L(M)$?

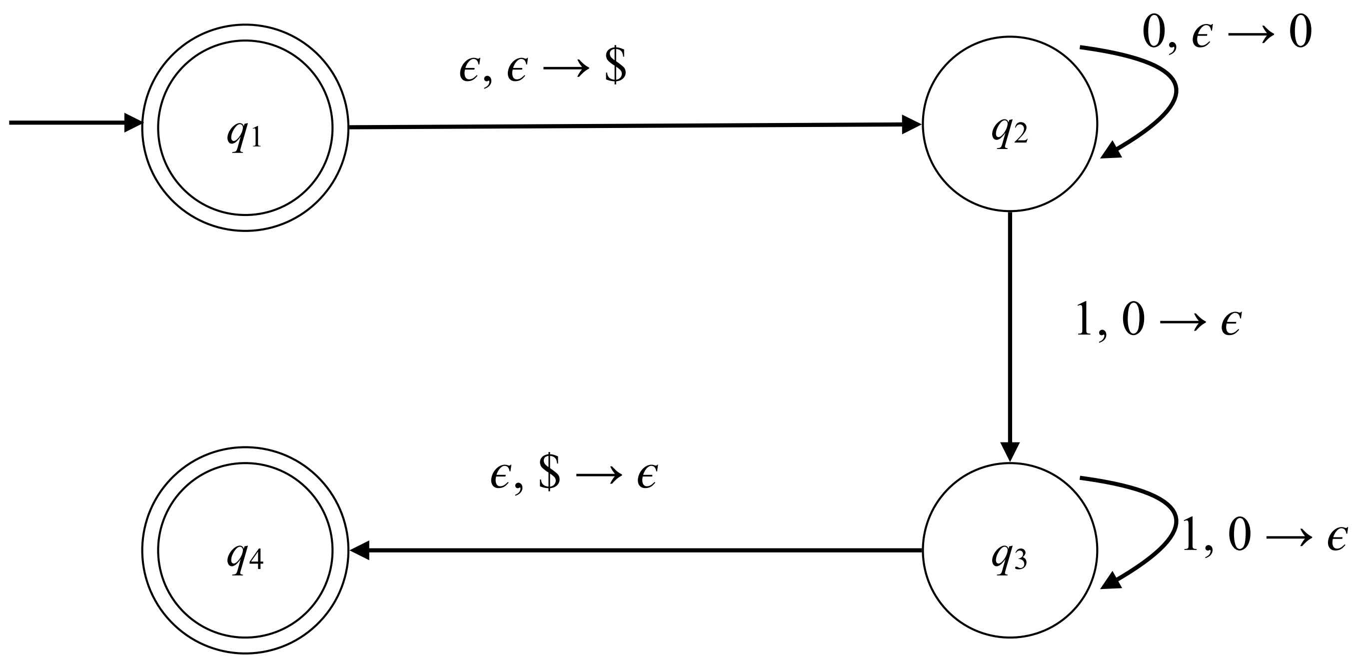
Input: $s = 00011$

Input: $s = \epsilon$



$\Sigma = \{0,1\}$, $\boldsymbol{\Gamma} = \{0, \$\}$

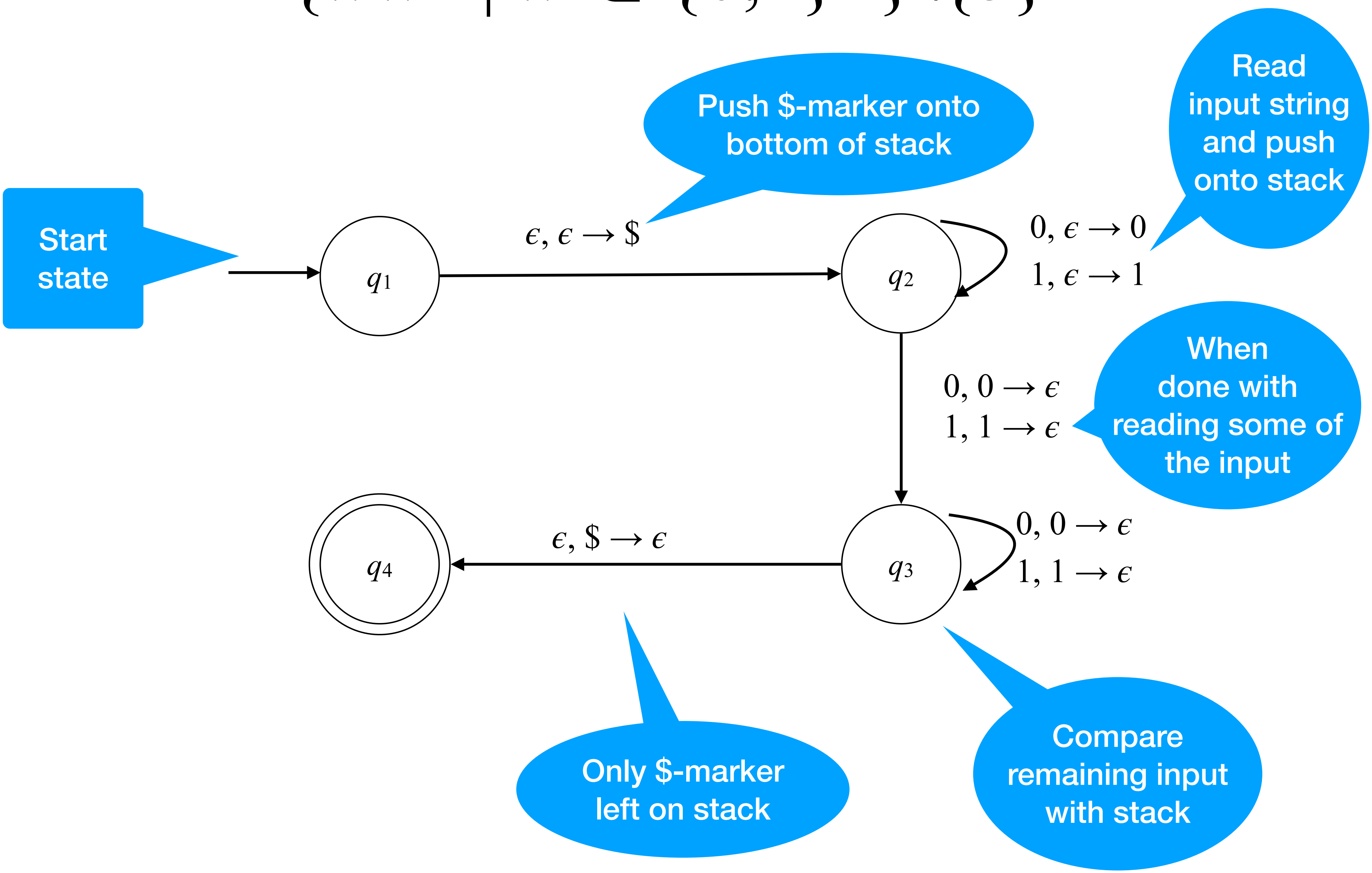
What is $L(M)$?



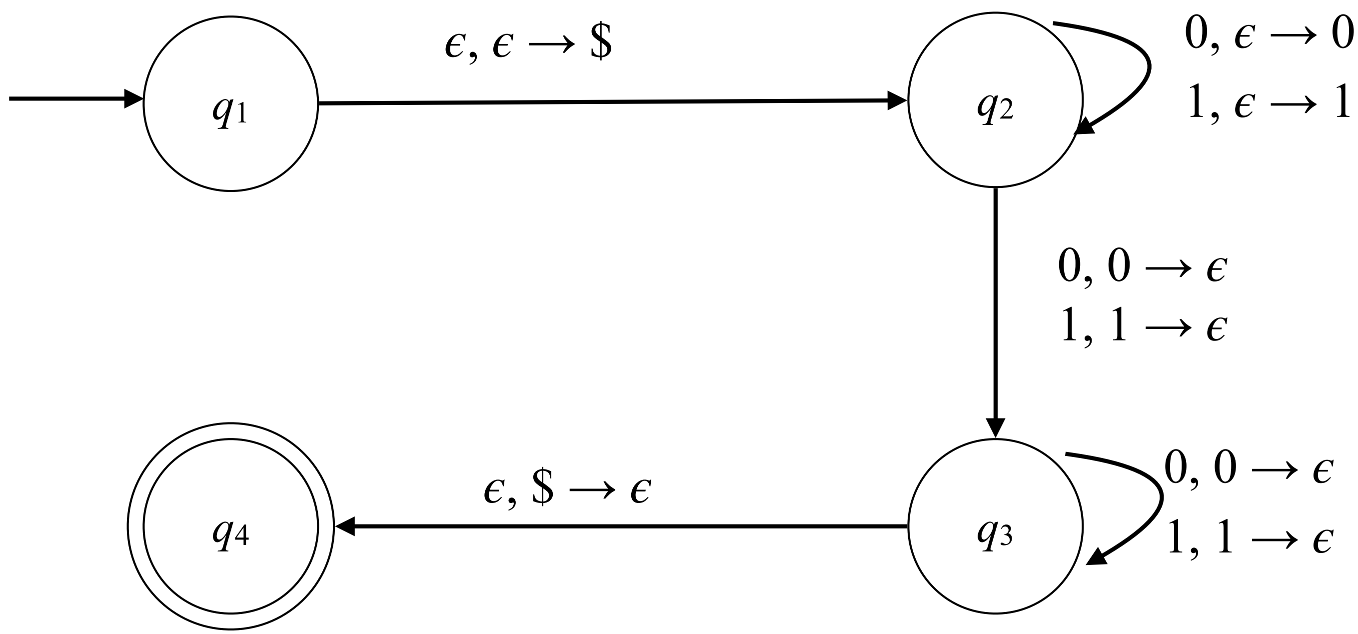
$L = \{0^n 1^n \mid n \geq 0\}$

$\Sigma = \{0,1\}$, $\mathbf{\Gamma} = \{0, \$\}$

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

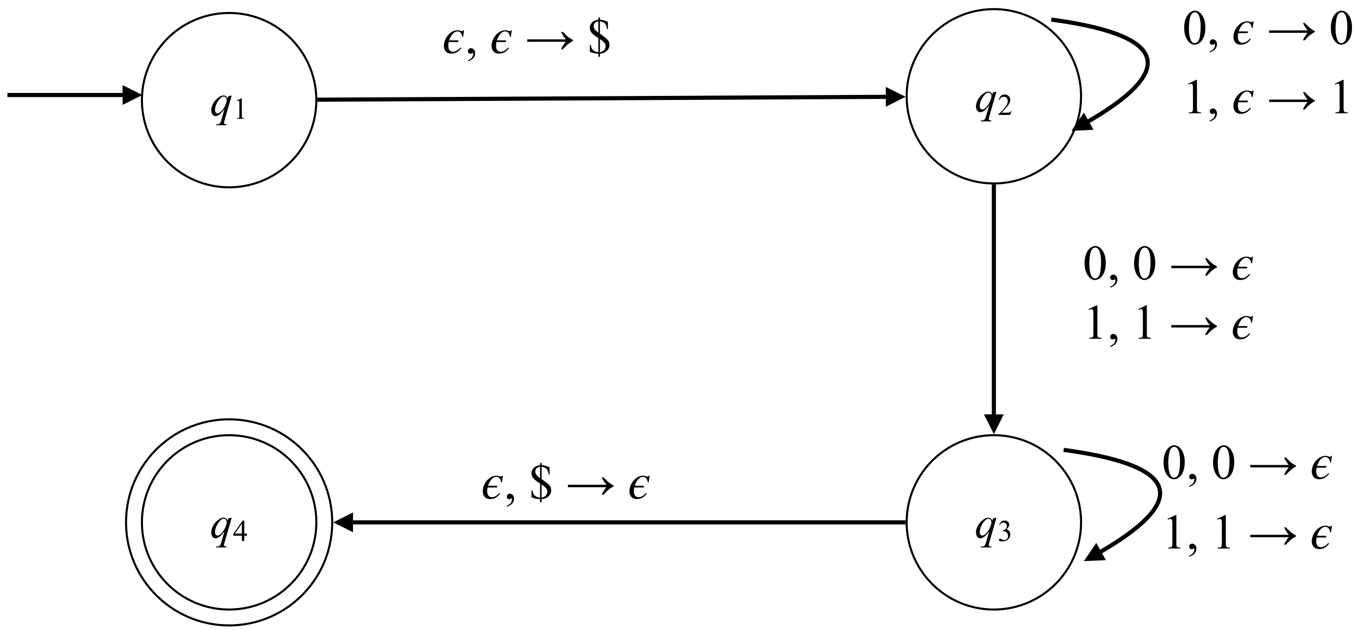


Designing PDA for $\{ww^R \mid w \in \{0,1\}^*\} \setminus \{\epsilon\}$



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

Questions

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

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 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 $L(G) = L(M)$