# CS 6041
# Theory of Computation

## Review 2

**Kun Suo**

Computer Science, Kennesaw State University
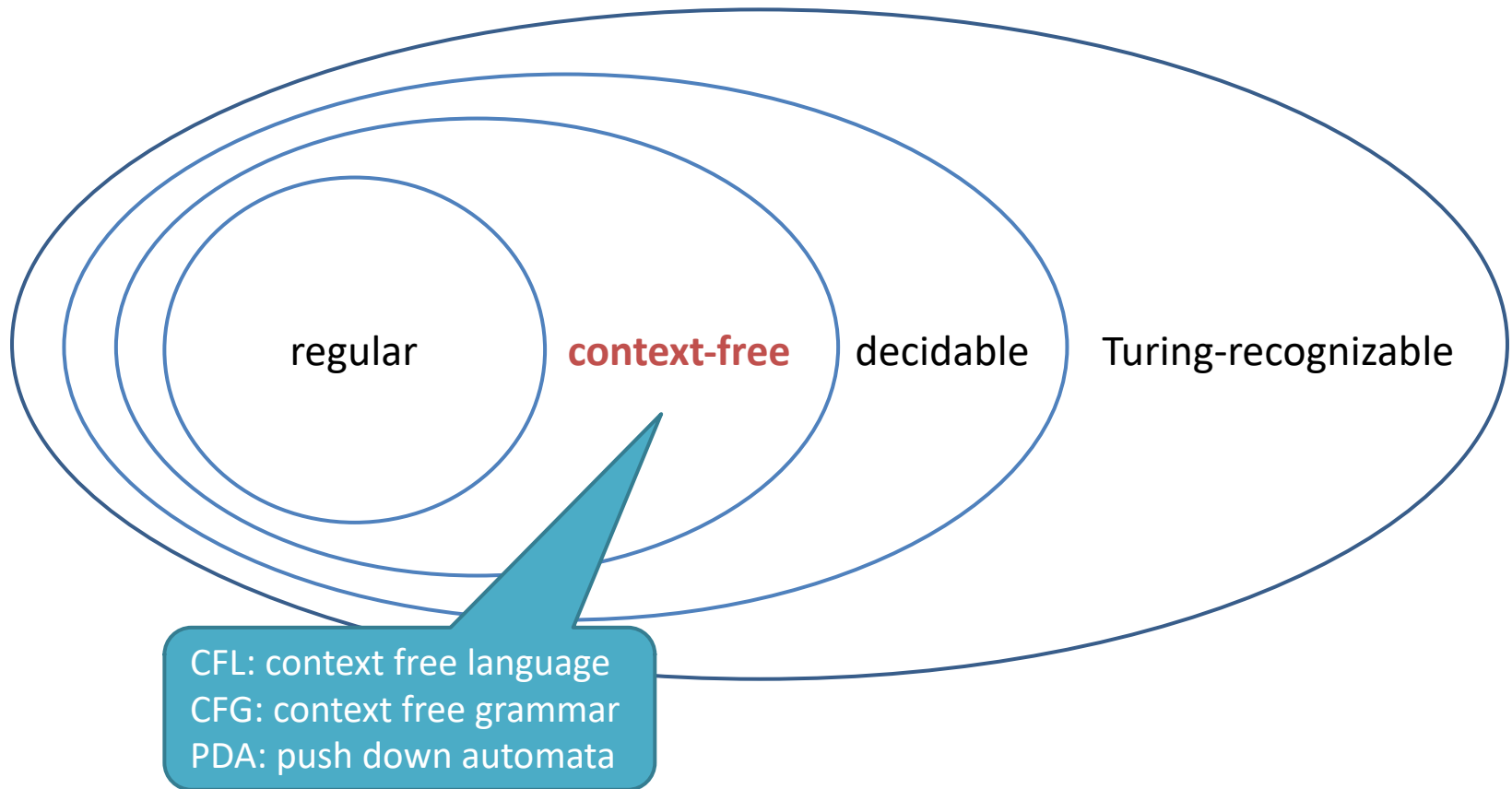
https://kevinsuo.github.io/

# Exam 2 – similar as Exam 1

- ## 10 True/False question
  - o 2 points each

- ## 4 short answer question
  - o 20 points each

- ## 100 = 2*10 + 4*20

# Context-free language



regular    **context-free**    decidable    Turing-recognizable

CFL: context free language
CFG: context free grammar
PDA: push down automata

# Context Free Grammar

- Example, $G_1$

Variable:
A, B

Start variable:
A

3 substitution rules
(productions)

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

Terminals:
0, 1, #

$A \Rightarrow 0A1$
$\Rightarrow 00A11$
$\Rightarrow 000A111$
$\Rightarrow 000B111$
$\Rightarrow 000\#111$

# The language of grammar

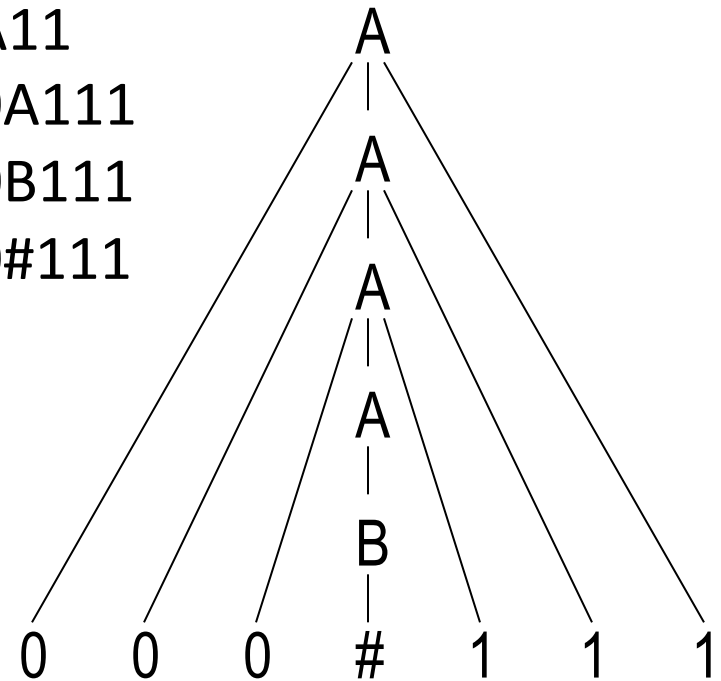- Grammar $G_1$:

  $A \rightarrow 0A1$

  $A \rightarrow B$

  $B \rightarrow \#$

- The language of $G_1$:

  $L(G_1)=\{\ 0^n\#1^n \mid n \geq 0\ \}$

- Context-free language

  o Languages generated by context-free grammars

$A \Rightarrow 0A1$
$\Rightarrow 00A11$
$\Rightarrow 000A111$
$\Rightarrow 000B111$
$\Rightarrow 000\#111$



000#111

# Definition of context-free grammar

- Context-free grammar is a 4-tuple G=(V,$\Sigma$,R,S),

  1) V: finite variable set

  2) $\Sigma$: finite terminal set

  3) R: finite rule set

      (A$\rightarrow$w, w$\in$(V$\cup\Sigma$)* )

  4) S$\in$V: start variable

# Design CFG for languages

- Design CFG is much difficult than designing an automata for language

- Basic idea:

  1. divide CFL into small parts

  2. design CFG for each small part

  3. combine them together

# Design context-free grammar

- Design CFG for $\{w \mid w = 0^n1^n \text{ or } w = 1^n0^n, n \geq 0\}$

# Design context-free grammar

- Design CFG for $\{w \mid w=0^n1^n$ or $w=1^n0^n, n \geq 0\}$

  - Design CFG for $\{w \mid w=0^n1^n, n \geq 0\}$

  - Design CFG for $\{w \mid w=1^n0^n, n \geq 0\}$

# Design context-free grammar

- Design CFG for $\{w \mid w = 0^n 1^n \text{ or } w = 1^n 0^n, n \geq 0\}$

  o Design CFG for $\{w \mid w = 0^n 1^n, n \geq 0\}$

  Generating same number of 0 and 1
  Generating 0 before 1

  01   0011   000111   00..011..1

# Design context-free grammar

- Design CFG for $\{w\,|\,w=0^n1^n$ or $w=1^n0^n, n\geq 0\}$

  o Design CFG for $\{w\,|\,w=0^n1^n, n\geq 0\}$

  Generating same number of 0 and 1
  Generating 0 before 1

  01   0011   000111   00..011..1

# Design context-free grammar

- Design CFG for $\{w \mid w=0^n1^n$ or $w=1^n0^n, n \geq 0\}$

  - Design CFG for $\{w \mid w=0^n1^n, n \geq 0\}$

    Generating same number of 0 and 1
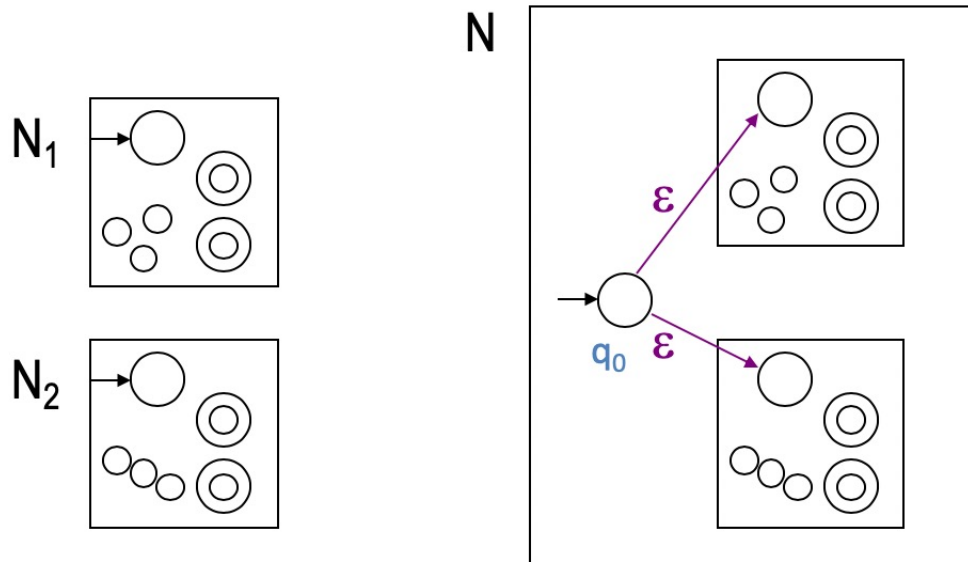    Generating 0 before 1

  01  0011  000111  00..011..1

# Design context-free grammar

- Design CFG for $\{w \mid w=0^n1^n \text{ or } w=1^n0^n, n \geq 0\}$

  o Design CFG for $\{w \mid w=0^n1^n, n \geq 0\}$

  Generating same number of 0 and 1
  Generating 0 before 1

  01    0011    000111    00..01..11

  $S \rightarrow 0S1$

# Design context-free grammar

- Design CFG for $\{w \mid w=0^n1^n \text{ or } w=1^n0^n, n \geq 0\}$

  - Design CFG for $\{w \mid w=0^n1^n, n \geq 0\}$

    - $G_1 = (\{S\}, \{0,1\}, \{S \rightarrow 0S1, S \rightarrow \varepsilon\}, S)$

  - Design CFG for $\{w \mid w=1^n0^n, n \geq 0\}$

    - $G_2 = (\{S\}, \{0,1\}, \{S \rightarrow 1S0, S \rightarrow \varepsilon\}, S)$

# Design context-free grammar

- Design CFG for $\{w \mid w = 0^n1^n$ or $w = 1^n0^n, n \geq 0\}$

  o Design CFG for $\{w \mid w = 0^n1^n, n \geq 0\}$

    ▸ $G_1 = (\{S_1\}, \{0,1\}, \{S_1 \rightarrow 0S_11, S_1 \rightarrow \varepsilon\}, S_1)$

  o Design CFG for $\{w \mid w = 1^n0^n, n \geq 0\}$

    ▸ $G_2 = (\{S_2\}, \{0,1\}, \{S_2 \rightarrow 1S_20, S_2 \rightarrow \varepsilon\}, S_2)$

# Design context-free grammar

- Design CFG for $\{w \mid w=0^n1^n \text{ or } w=1^n0^n, n \geq 0\}$

  - Design CFG for $\{w \mid w=0^n1^n, n \geq 0\}$

    - $G_1 = (\{S_1\}, \{0,1\}, \{S_1 \rightarrow 0S_11, S_1 \rightarrow \varepsilon\}, S_1)$

  - Design CFG for $\{w \mid w=1^n0^n, n \geq 0\}$

    - $G_2 = (\{S_2\}, \{0,1\}, \{S_2 \rightarrow 1S_20, S_2 \rightarrow \varepsilon\}, S_2)$

# Design context-free grammar

- Design CFG for $\{w \mid w=0^n1^n$ or $w=1^n0^n, n \geq 0\}$

  - Design CFG for $\{w \mid w=0^n1^n, n \geq 0\}$

    - $G_1 = (\{S_1\}, \{0,1\}, \{S_1 \rightarrow 0S_11, S_1 \rightarrow \varepsilon\}, S_1)$

  - Design CFG for $\{w \mid w=1^n0^n, n \geq 0\}$

    - $G_2 = (\{S_2\}, \{0,1\}, \{S_2 \rightarrow 1S_20, S_2 \rightarrow \varepsilon\}, S_2)$

  - $G = (\{S, S_1, S_2\}, \{0,1\},$

    $\{ S \rightarrow S_1, S \rightarrow S_2, S_1 \rightarrow 0S_11, S_1 \rightarrow \varepsilon, S_2 \rightarrow 1S_20, S_2 \rightarrow \varepsilon\},$

    $S)$

# Design CFG for languages

- Design CFG is much difficult than designing an automata for language

- Other ideas:

  1. Simulate the regular expressions

  2. Look for a pattern from example strings

  3. ...

# Design CFG for languages

- L={w| w has at least three 1s}, $\Sigma = \{0,1\}$

$\Sigma^*1\,\Sigma^*\,1\,\Sigma^*\,1\,\Sigma^*$

# Design CFG for languages

- L={w| w has at least three 1s}, $\Sigma = \{0,1\}$

$\Sigma^*1 \Sigma^* 1 \Sigma^* 1 \Sigma^*$

$S \rightarrow R1R1R1R$

# Design CFG for languages

- L={w| w has at least three 1s}, $\Sigma = \{0,1\}$

$\Sigma^* 1 \Sigma^* 1 \Sigma^* 1 \Sigma^*$

S → R1R1R1R

R → 0R

R → 1R

R → ε

# Design CFG for languages

- L={w| w has odd length}, Σ = {0,1}

$$\Sigma(\Sigma\ \Sigma)^{\ *}$$

# Design CFG for languages

- L={w| w has odd length}, Σ = {0,1}

$$\Sigma(\Sigma\,\Sigma)^{*}$$

S → 0

S → 1

# Design CFG for languages

- L={w| w has odd length}, Σ = {0,1}

$\Sigma(\Sigma\ \Sigma)^{*}$

S → 0

S → 1

S → S00

S → S01

S → S10

S → S11

# Design CFG for languages

- L={w| w has odd length}, Σ = {0,1}

Σ(Σ Σ) *

S → 0

S → 1

S → S00

S → S01

S → S10

S → S11

# Design CFG for languages

- L={w| w has odd length and the middle symbol is 0}, $\Sigma$ = {0,1}

0

000

001

100

101

00011

…

# Design CFG for languages

- L={w| w has odd length and the middle symbol is 0}, Σ = {0,1}

S → 0

S → 0S0

S → 0S1

S → 1S0

S → 1S1

0

000

001

100

101

00011

...

# Design CFG for languages

- L = $\{0^n1^n \mid n \geq 0\}$. $\Sigma = \{0,1\}$

ε, 01, 0011, …

S → 0S1 | ε

# Design CFG for languages

- L = $\{0^n 1^{2n} \mid n \geq 0\}$. $\Sigma = \{0,1\}$

ε, 011, 001111, …

S → 0S11 | ε

# Design CFG for languages

- L = {00*11*}. Σ = {0,1}

01, 011, 0011, …

How to design 00*

How to design 11*

# Design CFG for languages

- L = {00*11*}. Σ = {0,1}

How to design 00*

C → 0

C → 0C

# Design CFG for languages

- L = {00*11*}. Σ = {0,1}

How to design 11*

D → 1

D → 1D

# Design CFG for languages

- L = {00*11*}. Σ = {0,1}

How to design 00*11*

S → CD

C → 0C | 0

D → 1D | 1

How to design 00*

C → 0

C → 0C

How to design 11*

D → 1

D → 1D

# Ambiguity

- If a grammar generates the *same* string in several *different* ways, we say that the string is derived *ambiguously* in that grammar.

- If a grammar generates some string ambiguously, we say that the grammar is *ambiguous*.

- $G_5$: E $\rightarrow$

    E+E |

    E$\times$E |

    (E) | a

# **Ambiguity**

- $G_5: E \rightarrow$

    $E+E$ |

    $E \times E$ |

    $(E)$  | a
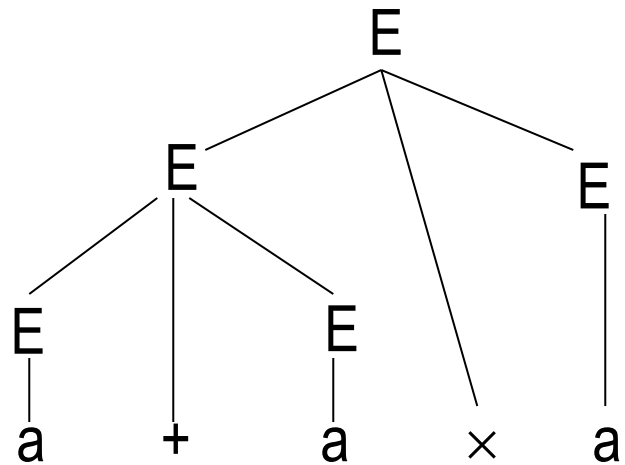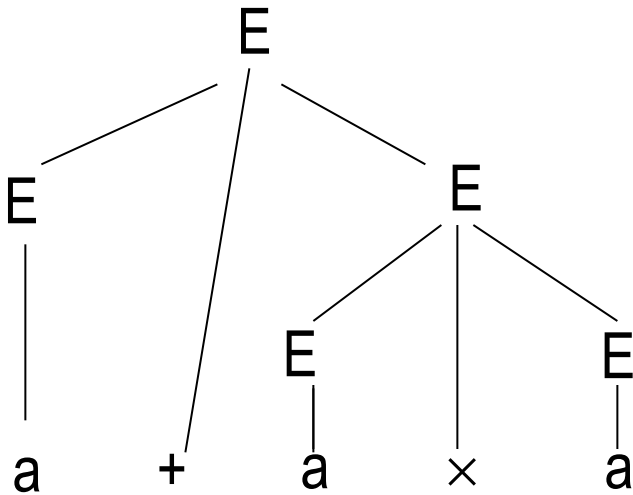
# Leftmost derivation

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

- $E \Rightarrow E+E$

  $\Rightarrow a+E$

  $\Rightarrow a+E \times E$

  $\Rightarrow a+a \times E \Rightarrow a+a \times a$

- $G_5: E \rightarrow$

  $E+E \mid$

  $E \times E \mid$

  $(E) \mid a$

# Two different leftmost derivation

- E
  $\Rightarrow$ E+E
  $\Rightarrow$ a+E
  $\Rightarrow$ a+E×E
  $\Rightarrow$ a+a×E
  $\Rightarrow$ a+a×a

- E
  $\Rightarrow$ E×E
  $\Rightarrow$ E+E ×E
  $\Rightarrow$ a+E×E
  $\Rightarrow$ a+a×E
  $\Rightarrow$ a+a×a

- $G_5$: E $\rightarrow$
  E+E |
  E×E |
  (E) | a

# Chomsky normal form (CNF)

- CNF: only allow CFG in the following forms

  o S→ε
  o A→BC
  o A→a

  Only start variable S can generate ε

  Variables can only generate:
  1, two variables
  2, single terminal

  ✓ S→ε          ✗ A→ε
  ✓ A→BC         ✗ A→B
  ✓ A→a          ✗ A→abcd
                 ✗ A→aB

# Techniques for CNF

- Add new start variable if needed

- $A \rightarrow \varepsilon$, merge above rules with A

- $A \rightarrow B$, replace B with terminals or other rules

- $A \rightarrow aB$, replace with $U \rightarrow a$, $A \rightarrow UB$

- $A \rightarrow abcd$, replace with $A \rightarrow aU_1$, $U_1 \rightarrow bU_2$, $U_2 \rightarrow cd$

- $A \rightarrow BCD$, similar as the above

✓ $S \rightarrow \varepsilon$

✓ $A \rightarrow BC$

✓ $A \rightarrow a$

✗ $A \rightarrow \varepsilon$

✗ $A \rightarrow B$

✗ $A \rightarrow abcd$

✗ $A \rightarrow aB$

# Chomsky normal form example

$G_6$:  $S \rightarrow ASA \mid aB$,
            ✖       ✖

   $A \rightarrow B \mid S$
            ✖      ✖

   $B \rightarrow b \mid \varepsilon$
            ✔      ✖

Get the CNF for $G_6$

✔ $S \rightarrow \varepsilon$        ✖ $A \rightarrow \varepsilon$

✔ $A \rightarrow BC$        ✖ $A \rightarrow B$

✔ $A \rightarrow a$        ✖ $A \rightarrow abcd$

                              ✖ $A \rightarrow aB$

# Step 1

$G_6$: $S \rightarrow ASA \mid aB$,
      ✗    ✗

     $A \rightarrow B \mid S$
       ✗  ✗

     $B \rightarrow b \mid \varepsilon$
       ✓  ✗

✓ $S \rightarrow \varepsilon$

✓ $A \rightarrow BC$

✓ $A \rightarrow a$

✗ $A \rightarrow \varepsilon$

✗ $A \rightarrow B$

✗ $A \rightarrow abcd$

✗ $A \rightarrow aB$

(1)   $S_0 \rightarrow S$ ✗

     $S \rightarrow ASA \mid aB$
      ✗    ✗

     $A \rightarrow B \mid S$
       ✗  ✗

     $B \rightarrow b \mid \varepsilon$
       ✓  ✗

> To simply the start variable

# Step 2

(2a)  $S_0 \to S$ ✗

$S \to ASA \mid aB$ ✗ ✗

$A \to B \mid S$ ✗ ✗

$B \to b \mid \varepsilon$ ✓ ✗

> Only start variable S can generate $\varepsilon$

✓ $S \to \varepsilon$
✓ $A \to BC$
✓ $A \to a$
✗ $A \to \varepsilon$
✗ $A \to B$
✗ $A \to abcd$
✗ $A \to aB$

$S_0 \to S$ ✗

$S \to ASA \mid aB \mid a$ ✗ ✗ ✓

$A \to B \mid S \mid \varepsilon$ ✗ ✗ ✗

$B \to b$ ✓

# Step 2

(2b)  $S_0 \rightarrow S$ ✗

$S \rightarrow ASA \mid aB \mid a$ ✗ ✗ ✓

$A \rightarrow B \mid S \mid \varepsilon$ ✗ ✗ ✗

> Only start variable S can generate $\varepsilon$

$B \rightarrow b$ ✓

✓ $S \rightarrow \varepsilon$
✓ $A \rightarrow BC$
✓ $A \rightarrow a$
✗ $A \rightarrow \varepsilon$
✗ $A \rightarrow B$
✗ $A \rightarrow abcd$
✗ $A \rightarrow aB$

$S_0 \rightarrow S$ ✗

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$
✗ ✗ ✓ ✓ ✓ ✗

$A \rightarrow B \mid S$ ✗ ✗

> No $A \rightarrow \varepsilon$ anymore

$B \rightarrow b$ ✓

# Step 3

(3a)  $S_0 \rightarrow S$ ✗

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

✗ ✗ ✓ ✓ ✓ ✗

$A \rightarrow B \mid S$

✗ ✗

$B \rightarrow b$ ✓

| ✓ | $S \rightarrow \varepsilon$ |
| ✓ | $A \rightarrow BC$ |
| ✓ | $A \rightarrow a$ |
| ✗ | $A \rightarrow \varepsilon$ |
| ✗ | $A \rightarrow B$ |
| ✗ | $A \rightarrow abcd$ |
| ✗ | $A \rightarrow aB$ |

$S_0 \rightarrow S$ ✗

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

✗ ✗ ✓ ✓ ✓

$A \rightarrow B \mid S$

✗ ✗

$B \rightarrow b$ ✓

# Step 3

(3b)  $S_0 \rightarrow S$ ✗

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
✗ ✗ ✓ ✓ ✓

$A \rightarrow B \mid S$
✗ ✗

$B \rightarrow b$ ✓

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
✗ ✗ ✓ ✓ ✓

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
✗ ✗ ✓ ✓ ✓

$A \rightarrow B \mid S$
✗ ✗

$B \rightarrow b$ ✓

✓ $S \rightarrow \varepsilon$

✓ $A \rightarrow BC$

✓ $A \rightarrow a$

✗ $A \rightarrow \varepsilon$

✗ $A \rightarrow B$

✗ $A \rightarrow abcd$

✗ $A \rightarrow aB$

# Step 3

(3c) $S_0 \rightarrow$ ASA | aB | a | SA | AS
     ✗   ✗   ✓   ✓   ✓

$S \rightarrow$ ASA | aB | a | SA | AS
     ✗   ✗   ✓   ✓   ✓

$A \rightarrow$ B | S
      ✗   ✗

$B \rightarrow$ b
      ✓

$S_0 \rightarrow$ ASA | aB | a | SA | AS
     ✗   ✗   ✓   ✓   ✓

$S \rightarrow$ ASA | aB | a | SA | AS
     ✗   ✗   ✓   ✓   ✓

$A \rightarrow$ b | S
      ✓   ✗

$B \rightarrow$ b
      ✓

✓ $S \rightarrow \varepsilon$

✓ $A \rightarrow BC$

✓ $A \rightarrow a$

✗ $A \rightarrow \varepsilon$

✗ $A \rightarrow B$

✗ $A \rightarrow abcd$

✗ $A \rightarrow aB$

# Step 3

(3d) $S_0 \rightarrow$ ASA | aB | a | SA | AS
❌ ❌ ✓ ✓ ✓

$S \rightarrow$ ASA | aB | a | SA | AS
❌ ❌ ✓ ✓ ✓

$A \rightarrow$ b | S
✓ ❌

$B \rightarrow$ b
✓

$S_0 \rightarrow$ ASA | aB | a | SA | AS
❌ ❌ ✓ ✓ ✓

$S \rightarrow$ ASA | aB | a | SA | AS
❌ ❌ ✓ ✓ ✓

$A \rightarrow$ b | ASA | aB | a | SA | AS
✓ ❌ ❌ ✓ ✓ ✓

$B \rightarrow$ b
✓

✓ $S \rightarrow \varepsilon$
✓ $A \rightarrow BC$
✓ $A \rightarrow a$
❌ $A \rightarrow \varepsilon$
❌ $A \rightarrow B$
❌ $A \rightarrow abcd$
❌ $A \rightarrow aB$

# Step 4

(4)  $S_0 \rightarrow$ ASA | aB | a | SA | AS

   $S \rightarrow$ ASA | aB | a | SA | AS

   $A \rightarrow$ b | ASA | aB | a | SA | AS

   $B \rightarrow$ b

<br>

   $S_0 \rightarrow AA_1$ | aB | a | SA | AS

   $S \rightarrow AA_1$ | aB | a | SA | AS

   $A \rightarrow$ b | $AA_1$ | aB | a | SA | AS

   $B \rightarrow$ b

   $A_1 \rightarrow$ SA

$S \rightarrow \varepsilon$

$A \rightarrow BC$

$A \rightarrow a$

$A \rightarrow \varepsilon$

$A \rightarrow B$

$A \rightarrow abcd$

$A \rightarrow aB$

# Step 5

(5) $S_0 \rightarrow AA_1 \mid aB \mid a \mid SA \mid AS$ ✗

$S \rightarrow AA_1 \mid aB \mid a \mid SA \mid AS$ ✗

$A \rightarrow b \mid AA_1 \mid aB \mid a \mid SA \mid AS$ ✗

$B \rightarrow b$

$A_1 \rightarrow SA$

✓ $S \rightarrow \varepsilon$
✓ $A \rightarrow BC$
✓ $A \rightarrow a$
✗ $A \rightarrow \varepsilon$
✗ $A \rightarrow B$
✗ $A \rightarrow abcd$
✗ $A \rightarrow aB$

$S_0 \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$ ✓✓✓✓✓

$S \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$ ✓✓✓✓✓

$A \rightarrow b \mid AA_1 \mid UB \mid a \mid SA \mid AS$ ✓✓✓✓✓✓

$B \rightarrow b$ ✓

$A_1 \rightarrow SA$ ✓

$U \rightarrow a$ ✓

# Step 5

(5) $S_0 \rightarrow AA_1 \mid UB \mid a \mid$

$\qquad SA \mid AS$

$S \rightarrow AA_1 \mid UB \mid a \mid$

$\qquad SA \mid AS$

$A \rightarrow b \mid AA_1 \mid UB \mid a \mid$

$\qquad SA \mid AS$

$B \rightarrow b$

$A_1 \rightarrow SA$

$U \rightarrow a$

$G_6: \quad S \rightarrow ASA \mid aB,$

$\qquad A \rightarrow B \mid S$

$\qquad B \rightarrow b \mid \varepsilon$

G6 in CNF ⟵

✓ $S \rightarrow \varepsilon$

✓ $A \rightarrow BC$

✓ $A \rightarrow a$

✗ $A \rightarrow \varepsilon$

✗ $A \rightarrow B$

✗ $A \rightarrow abcd$

✗ $A \rightarrow aB$

# What does PDA look like?

finite automaton

Memory = 1
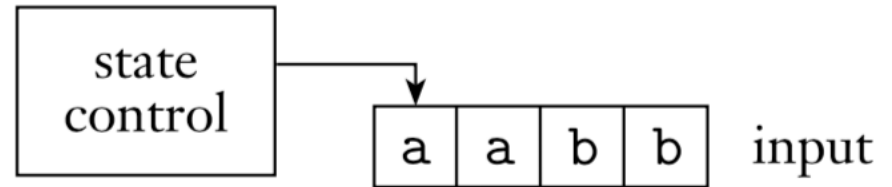
pushdown automaton

Memory = N

$ means empty stack
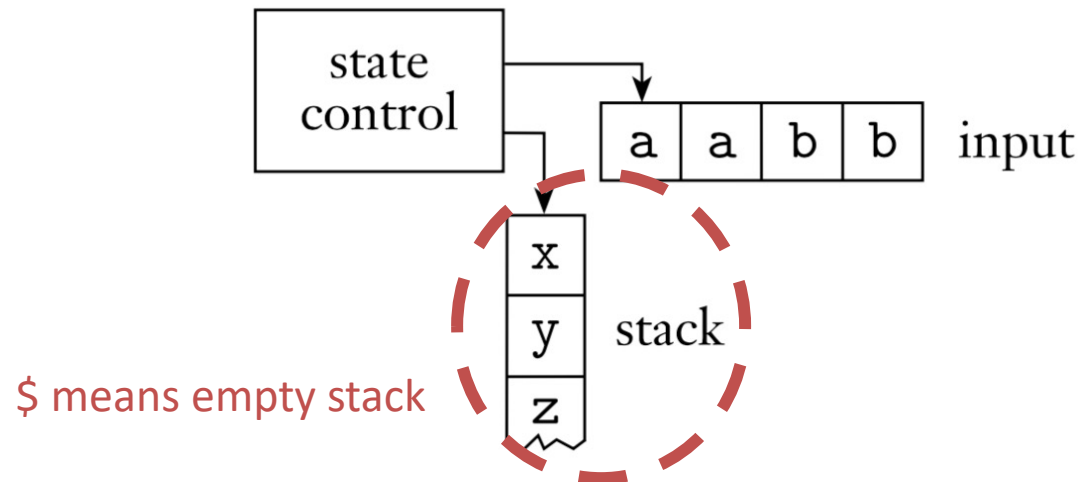
# What does PDA looks like?

pushdown automaton



$ means empty stack
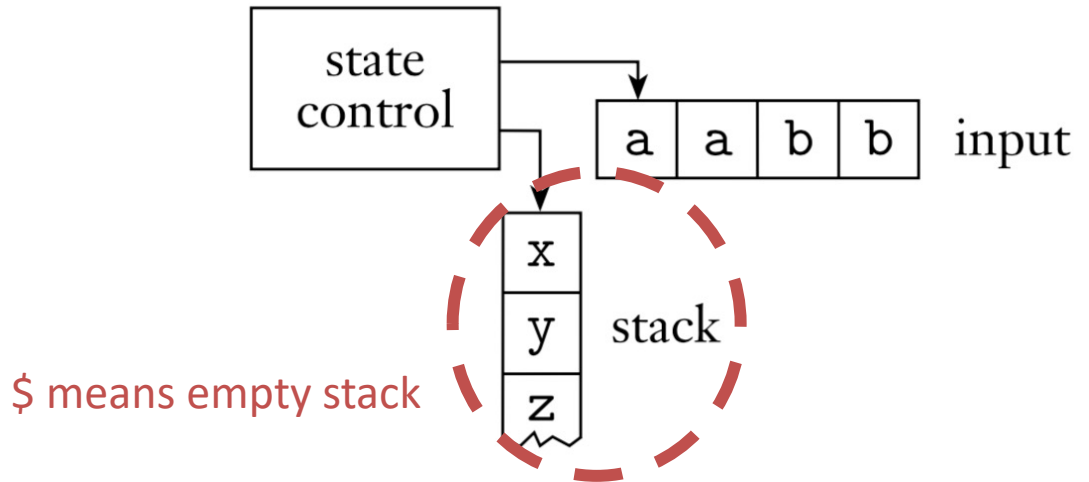
- Pushdown automata has more memories than finite automata

- PDA = finite automata + **A stack (unlimited size)**

# Stack operation

- Push: add to the top of stack

Push b into stack

Stack

Stack

# Stack operation

- Pop: remove from the top of stack



Pop c out of stack
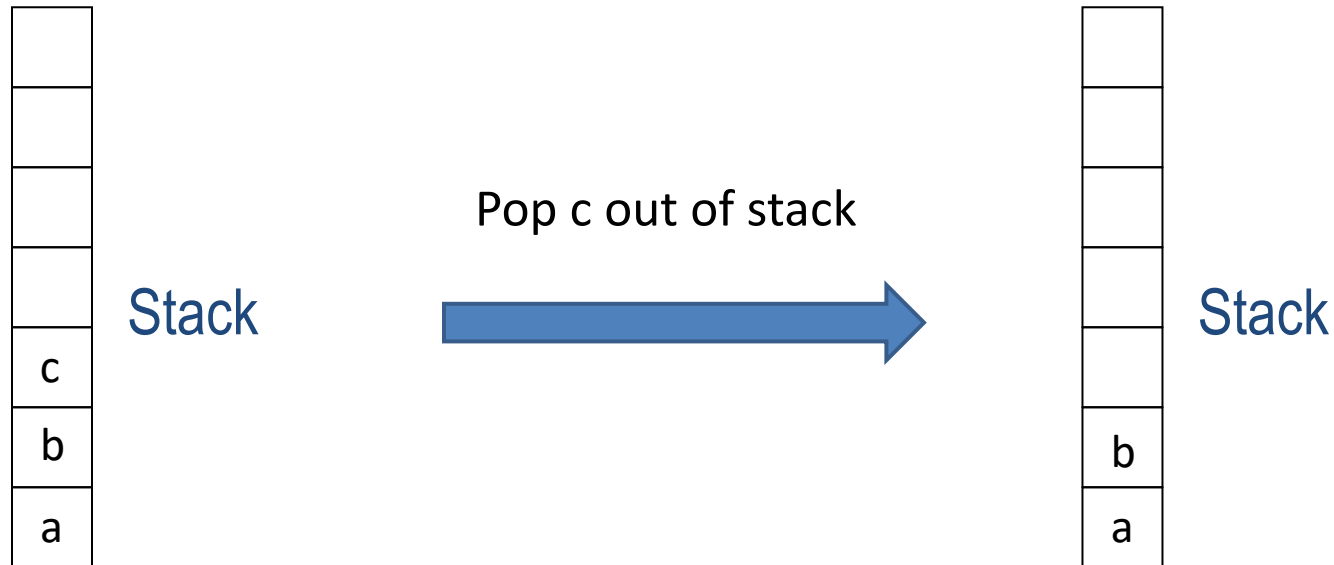
Stack

Stack

# Informal description for PDA to recognize some languages

- L = {w | w has some features}
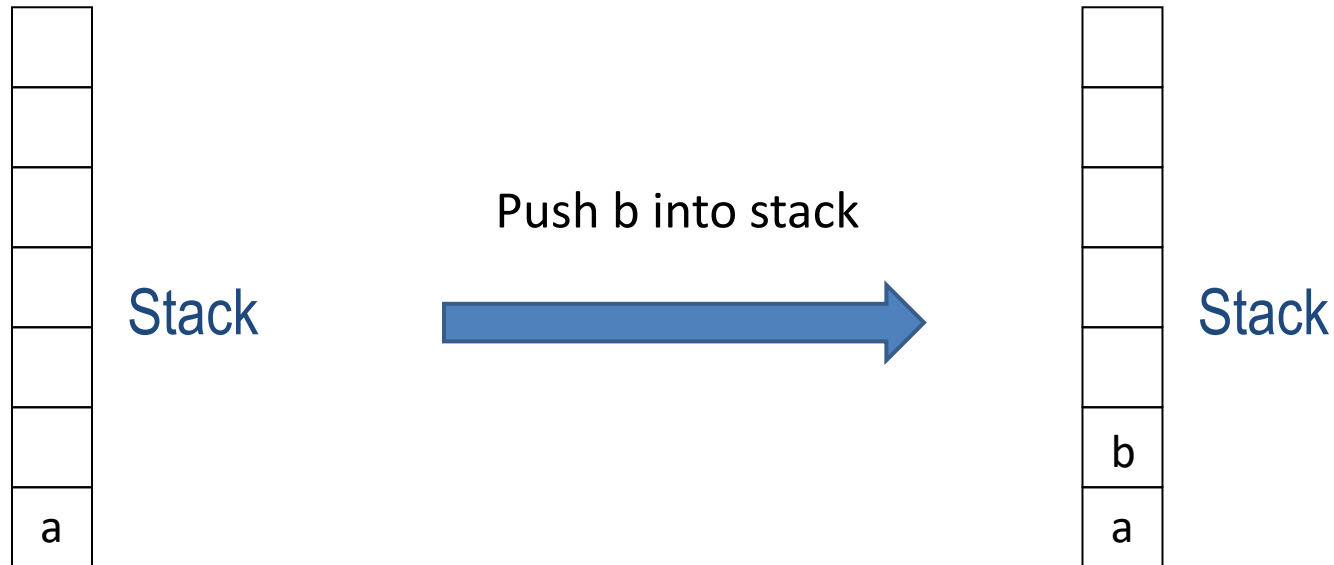
- Read symbols from input

  - **STEP1:** regular?
    - ▸ If the language is regular, do not need to use stack; if not regular, define operations on stack

  - **STEP2:** define operations:
    - ▸ When to push
    - ▸ When to pop

  - **STEP 3:** determine accept/reject:
    - ▸ Under which cases, accept
    - ▸ Under which cases, reject

Single direction, read only input tape

{ $0^n 1^n$ | n≥0 }

| | | $ | 0 | 0 | 1 | 1 | $ | | |

Single direction read only head

State controller

Stack

$
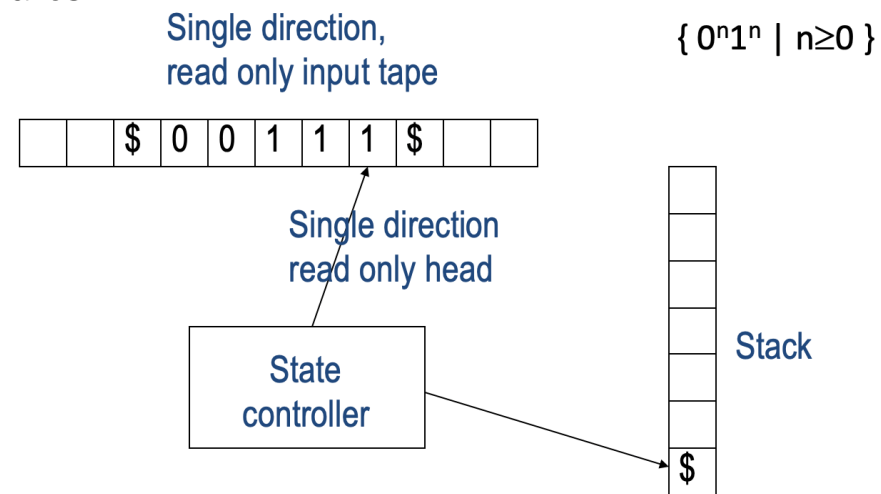
# **Definition of PDA** (non-deterministic)

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

  1) Q: set of states

  2) $\Sigma$: input alphabet, $\Sigma_\varepsilon=\Sigma\cup\{\varepsilon\}$

  3) $\Gamma$: stack alphabet, $\Gamma_\varepsilon=\Gamma\cup\{\varepsilon\}$

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

     transition function

  5) $q_0\in Q$: start state

  6) $F\subseteq Q$: accept state set

# Design PDA
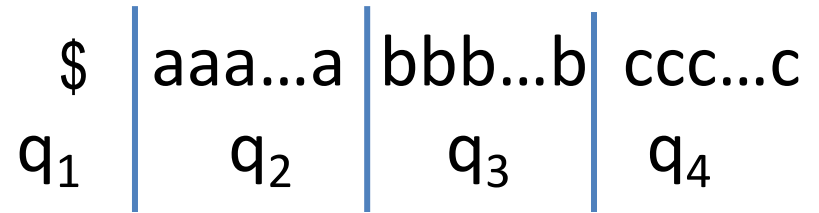
$$\$ \quad aaa...a \quad bbb...b \quad ccc...c$$
$$q_1 \quad\quad q_2 \quad\quad q_3 \quad\quad q_4$$

- ## $L(M_2)=\{\ a^n b^n c^m \mid m,n \geq 0\ \}$
    - ▸ Operation:
        - ☐ For an input a, and push a into stack
        - ☐ For an input b, pop one a from the stack
    - ▸ Determine accept/reject
        - ☐ If the stack is empty when finish reading b, then after reading all the cs, accept;
        - ☐ Otherwise, reject;

# Design PDA

- $L(M_2)=\{ a^n b^n c^m \mid m,n \geq 0 \}$

x,y->z
x: input
y->z: the top of stack changes

$x, y \rightarrow z$ $x, y \rightarrow z$

$\varepsilon, \varepsilon \rightarrow \$$

$q_1 \longrightarrow q_2 \longrightarrow q_3 \longrightarrow q_4$

$x, y \rightarrow z$ $x, y \rightarrow z$ $x, y \rightarrow z$
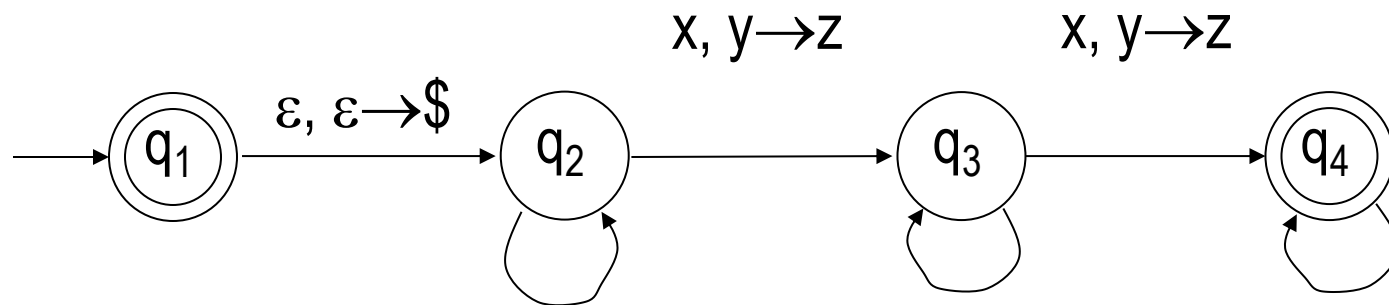
- $L(M_2)=\{ a^n b^n c^m \mid m,n \geq 0 \}$
  - Operation:
    - For an input a, and push a into stack
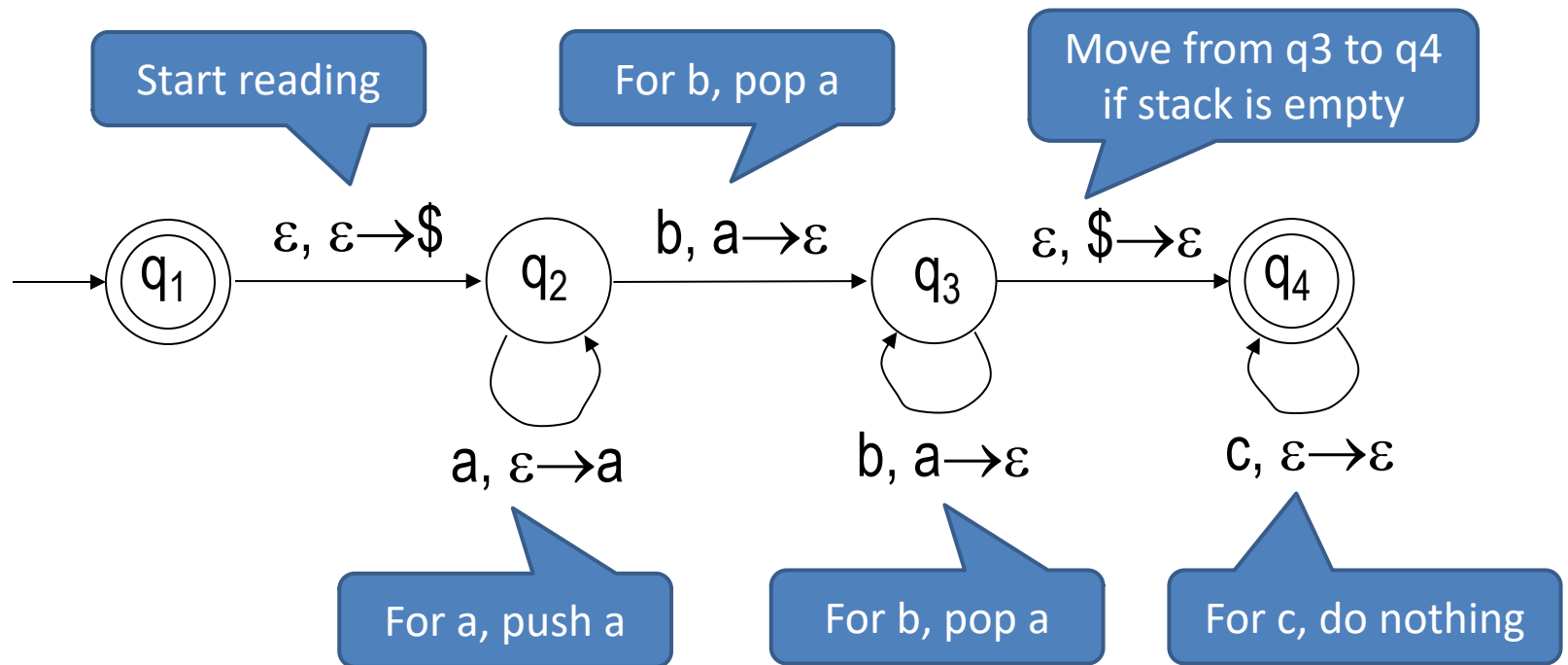    - For an input b, pop one a from the stack
  - Determine accept/reject
    - If the stack is empty when finish reading b, then after reading all the cs, accept;
    - Otherwise, reject;

# Design PDA

| $ | aaa...a | bbb...b | ccc...c |
|---|---------|---------|---------|
| $q_1$ | $q_2$ | $q_3$ | $q_4$ |

- $L(M_2) = \{\ a^n b^n c^m \mid m, n \geq 0\ \}$

Start reading

For b, pop a

Move from q3 to q4 if stack is empty

$q_1$ $\quad \varepsilon, \varepsilon \rightarrow \$ \quad$ $q_2$ $\quad b, a \rightarrow \varepsilon \quad$ $q_3$ $\quad \varepsilon, \$ \rightarrow \varepsilon \quad$ $q_4$

$a, \varepsilon \rightarrow a$

$b, a \rightarrow \varepsilon$

$c, \varepsilon \rightarrow \varepsilon$

For a, push a

For b, pop a

For c, do nothing

# Pumping lemma

Suppose A is CFL,

then there exist a number p(the pumping length) where,

if $s \in A$ and $|s| \geq p$, then s = **uvxyz**,

Satisfying the following

1) $\forall i \geq 0$, $uv^i xy^i z \in A$;

2) $|vy| > 0$;

3) $|vxy| \leq p$.

# Example: B={ $a^n b^n c^n$ | n≥0 }

- Proof:

Suppose B is CFL, p is the pumping length,

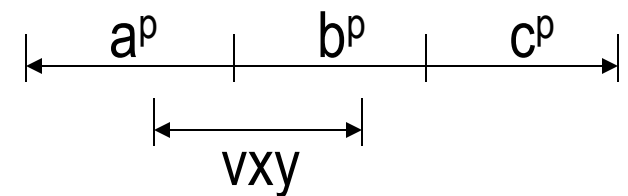let s=$a^p b^p c^p$
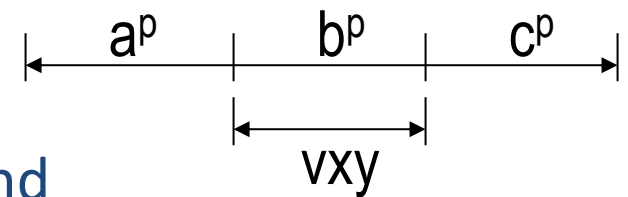
Then s = uvxyz, that

   $\forall i \geq 0,\ uv^i xy^i z \in B$;

   $|vy| > 0$, v and y have at least one kind

of symbol;

   $|vxy| \leq p$, v and y have at most two
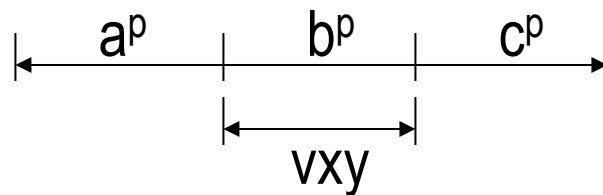
kinds of symbol;

# Example: B={ $a^n b^n c^n$ | n≥0 }
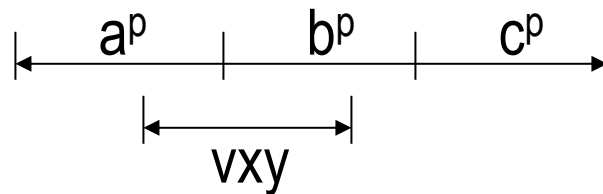
1) $\forall i \geq 0, uv^i x y^i z \in A$;

2) $|vy| > 0$;

3) $|vxy| \leq p$.

- Proof:

If v and y have one kind of symbol, then in $uv^i xy^i z$ (i>1), a/b/c has different numbers;



If v and y have two kinds of symbol, then in $uv^i xy^i z$ (i>1), a/b/c has different numbers;



Contradiction.

# Exam 2

- 10 True/False question
  - ∘ 2 points each


- 4 short answer question
  - ∘ 20 points each


- Time: check the time @D2L