Intro to Theory of Computation

CS
332

LECTURE 6

Last time:

- Pumping lemma
- Proving a language is not regular

Today:

- Pushdown automata (PDAs)
- Context-free grammars (CFGs)

On Friday: Homework 2 due
Homework 3 out

Sofya Raskhodnikova

CS 332

CS332 so far

MODEL OF A PROBLEM: LANGUAGE

MODEL OF A PROGRAM: DFA

EQUIVALENT MODELS: NFA, REGEXP

PROBLEMS THAT A DFA CAN'T SOLVE

ARE WE DONE?

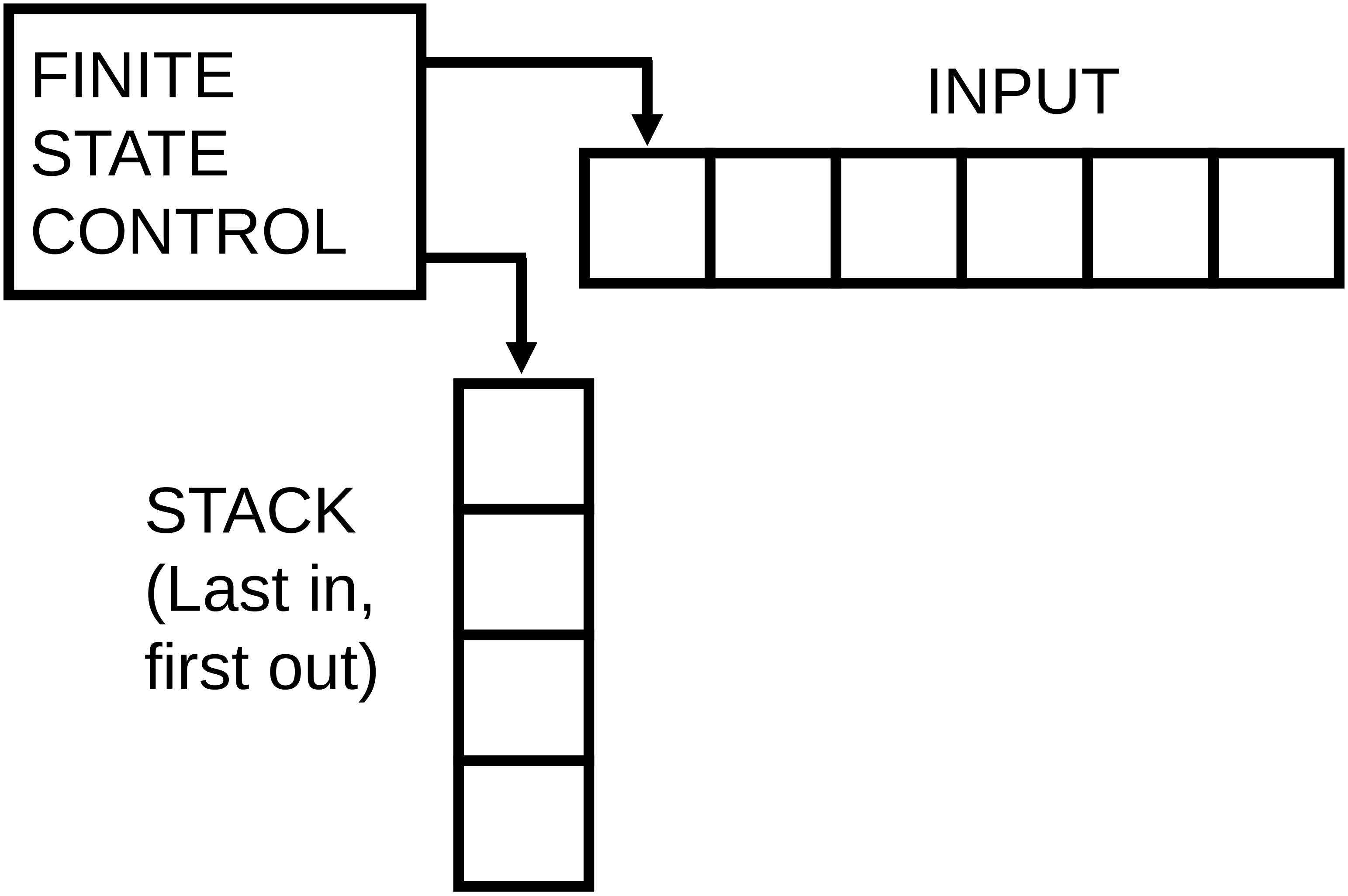
CS 332

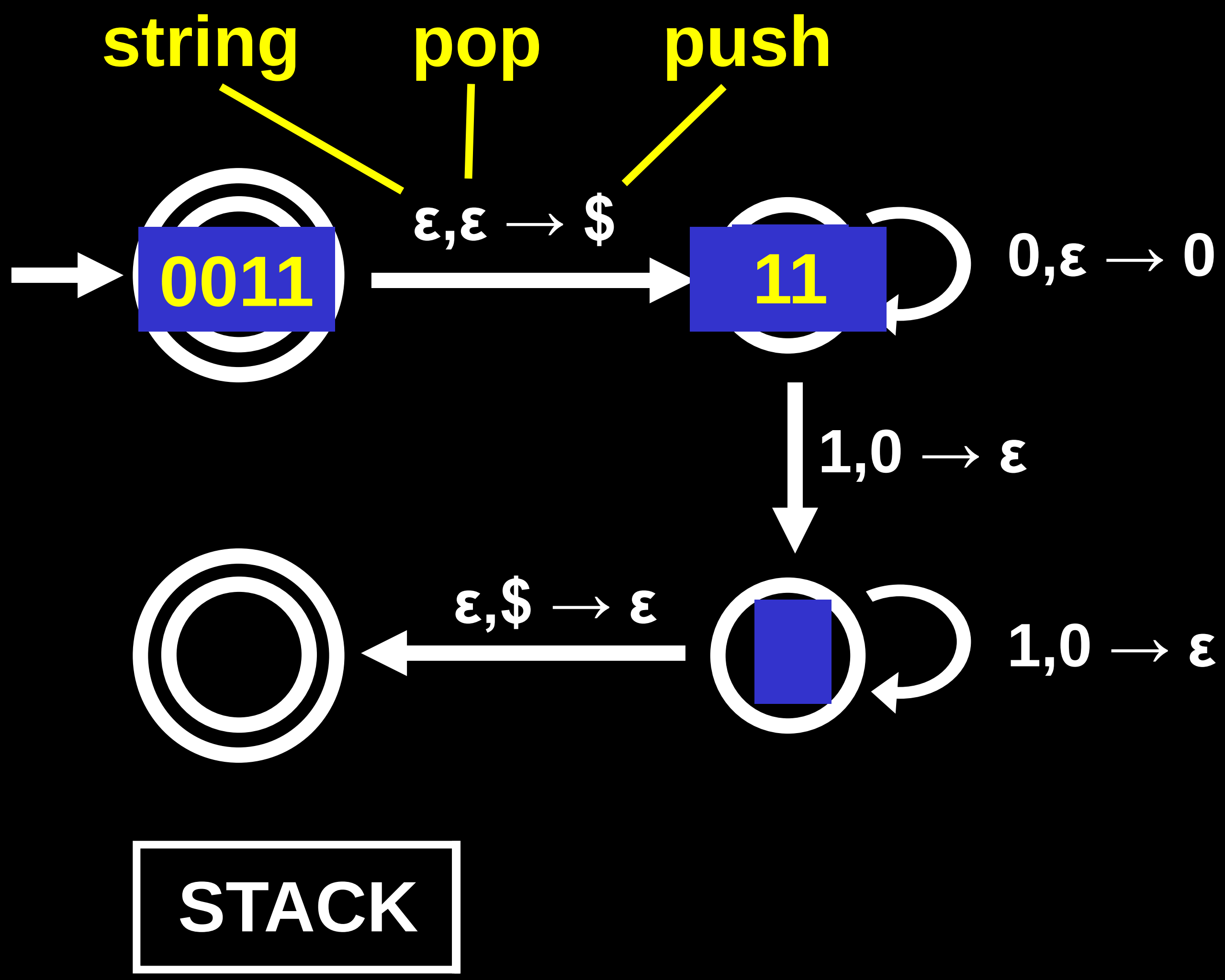
NONE OF THESE ARE REGULAR

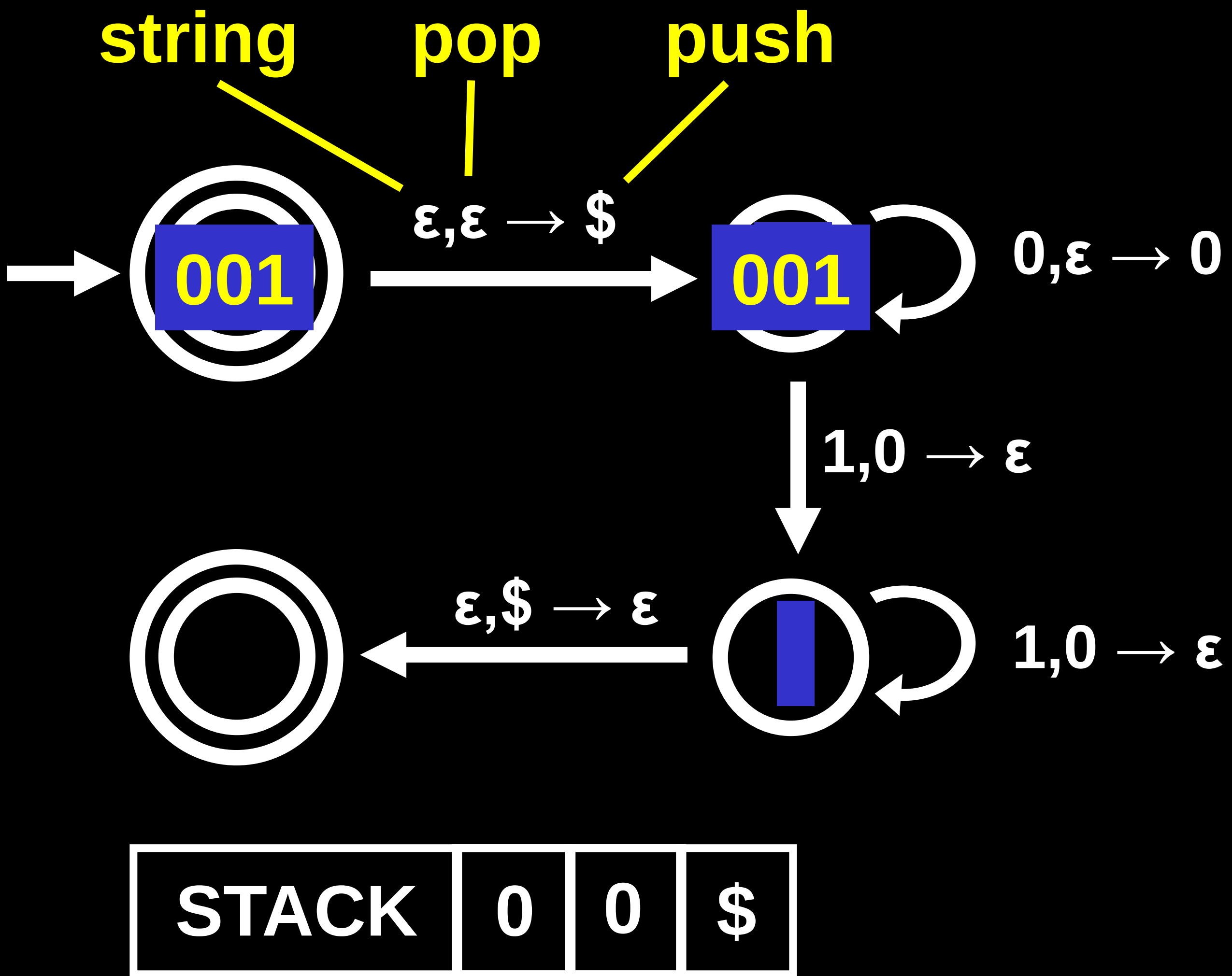
- $\Sigma = \{0, 1\}$, $L = \{ 0^n1^n \mid n \geq 0 \}$
- $\Sigma = \{a, b, c, \ldots, z\}$, $L = \{ w \mid w = w^R \}$
- $\Sigma = \{ (,) \}$, L = { balanced strings of parens }

We can write a C or JAVA program for any of them!

PUSHDOWN AUTOMATA (PDA)




string
pop
push
ε,ε → $
0011
11
0,ε → 0
1,0 → ε
ε,$ → ε
1,0 → ε
STACK



PDA to recognize $L = \{ 0^n1^n \mid n \geq 0 \}$

Formal Definition

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

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

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$ and $P(Q \times \Gamma_\varepsilon)$ is the set of subsets of $Q \times \Gamma_\varepsilon$

Note: A PDA is defined to be nondeterministic.

CS 332

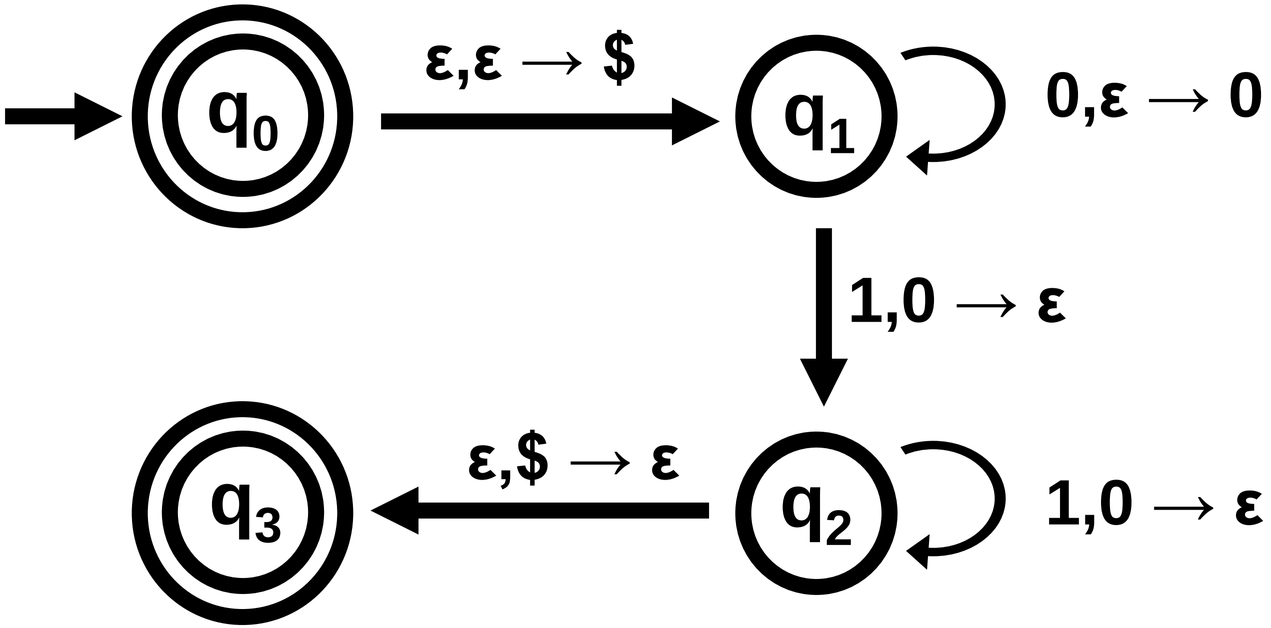
Formal Definition

A ***PDA*** is a 6-tuple $\mathbf{P = (Q, \Sigma, \Gamma, \delta, q_0, F)}$

- $\mathbf{Q}$ is a finite set of states
- $\mathbf{\Sigma}$ is the alphabet
- $\mathbf{\Gamma}$ is the stack alphabet
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow P(Q \times \Gamma_\varepsilon)$ **is the transition function**
- $\mathbf{q_0} \in \mathbf{Q}$ is the start state
- $\mathbf{F} \subseteq \mathbf{Q}$ is the set of accept states

A PDA starts with an *empty* stack.
It accepts a string if *at least one* of its computational branches reads *all the input* and gets into an *accept state* at the end of it.

An example PDA

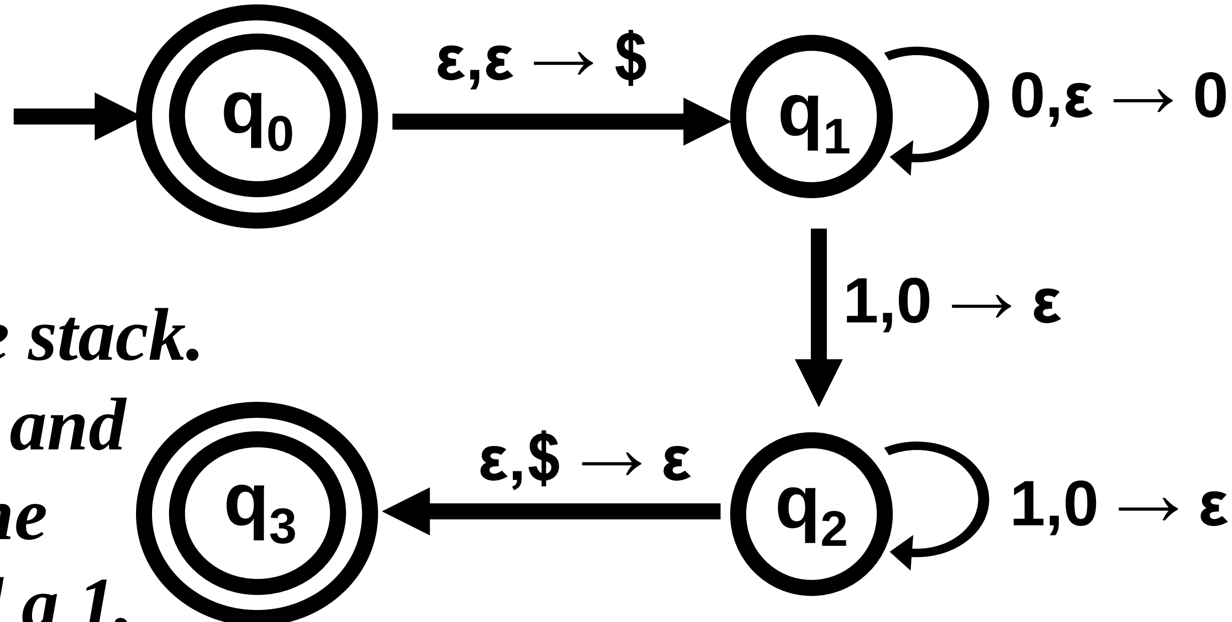


$Q = \{q_0, q_1, q_2, q_3\}$ $\Sigma = \{0,1\}$ $\Gamma = \{\$,0\}$

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

$\delta(q_1,1,0) = \{ (q_2,\varepsilon) \}$ $\delta(q_2,1,1) = \varnothing$

PDA: algorithmic description



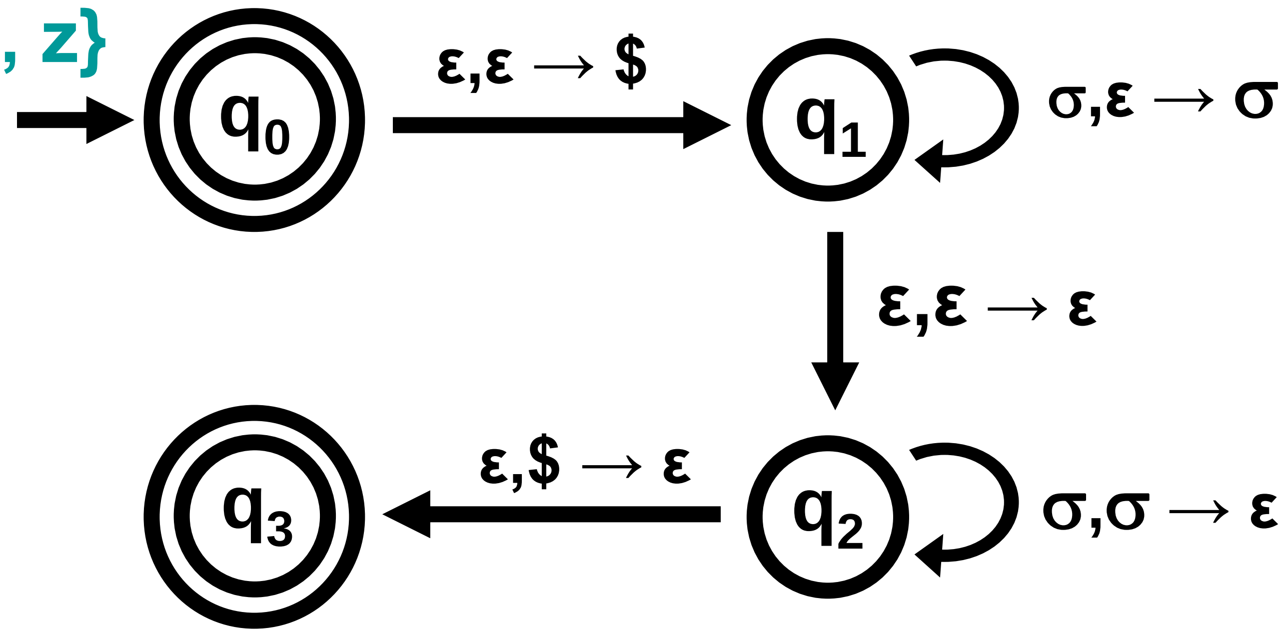
1. *Place the marker symbol $ onto the stack.*
2. *Keep reading a 0 and pushing it onto the stack. If you read a 1, pop a 0 and go to the next step.*
3. *Nondeterministically keep reading a 1 and popping a 0 or go to the next step.*
4. *If the top of the stack is $, enter the accept state. (Then PDA accepts if the input has been read).*

CS 332

Exercise

What strings are accepted by this PDA?

$\Sigma = \{a, b, c, \ldots, z\}$



A. Only ε

B. Palindromes

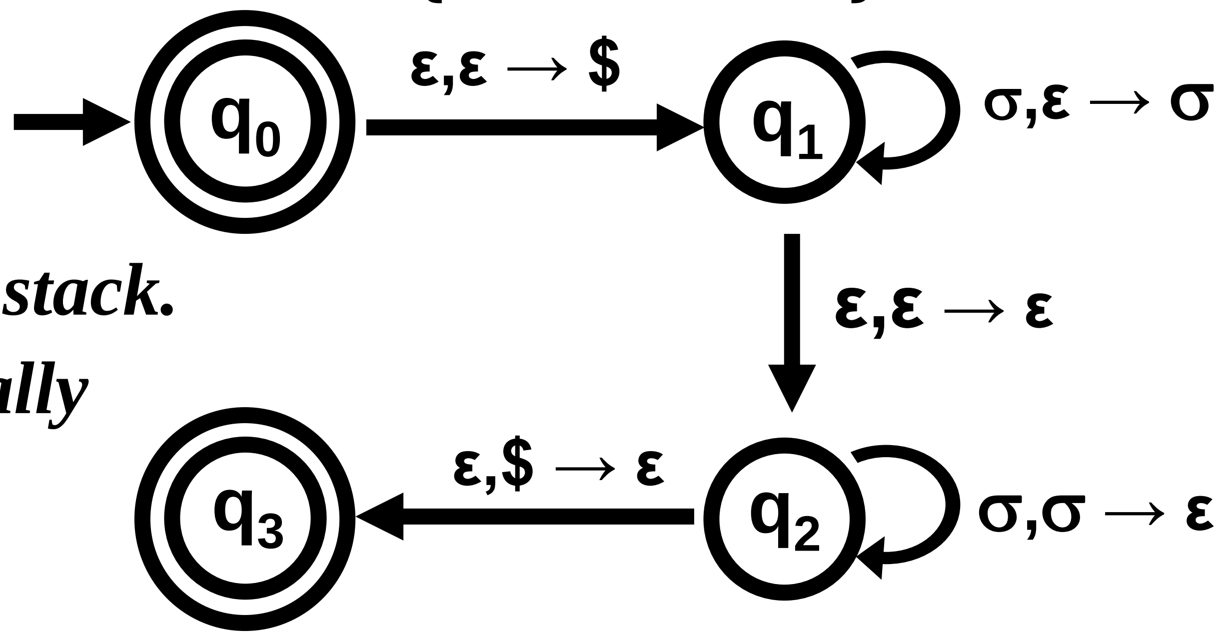
C. Even-length palindromes

D. All strings that start and end with the same letter

E. None of the above

Give an **ALGORITHMIC** description

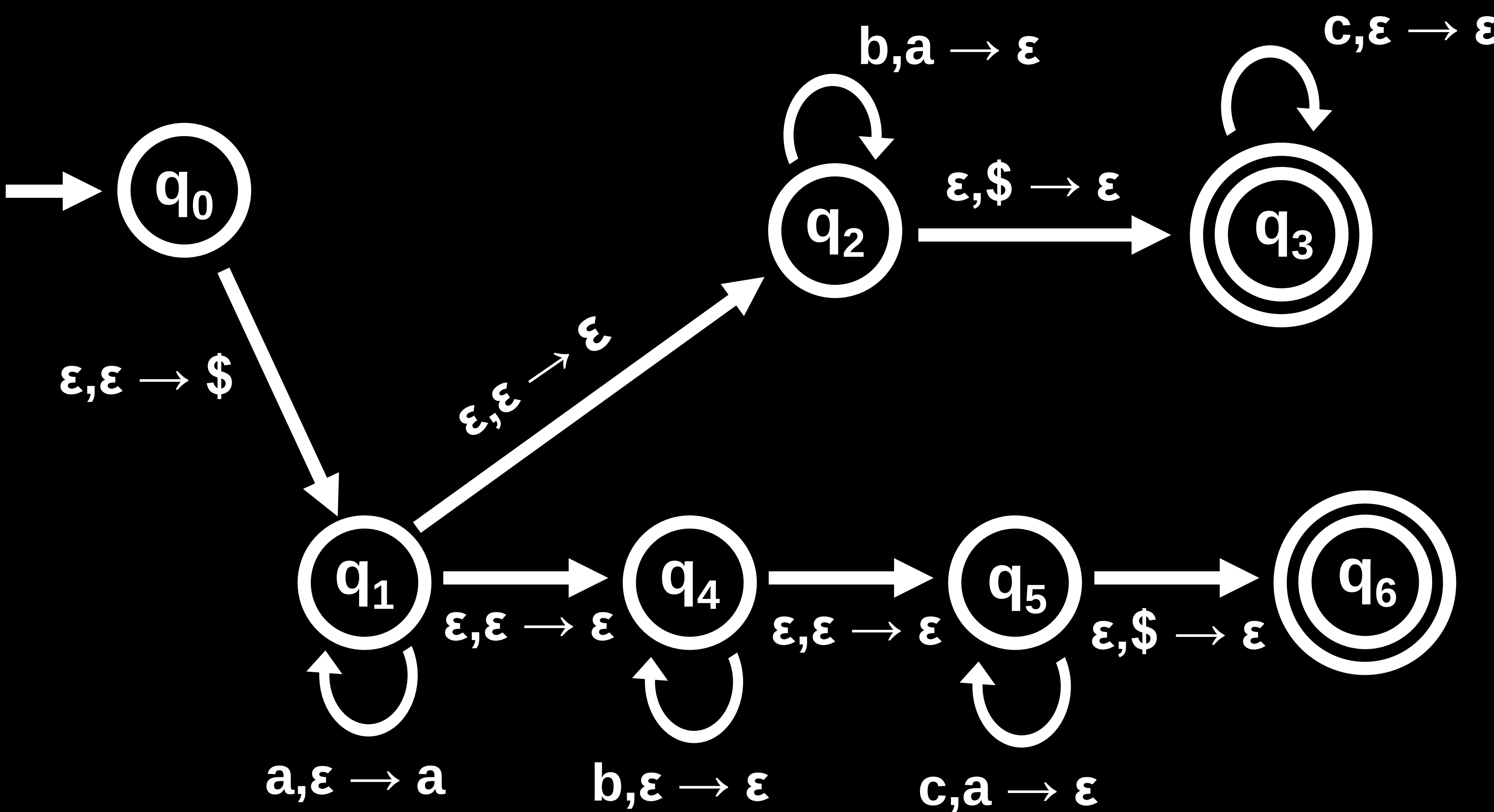
$\Sigma = \{a, b, c, \ldots, z\}$



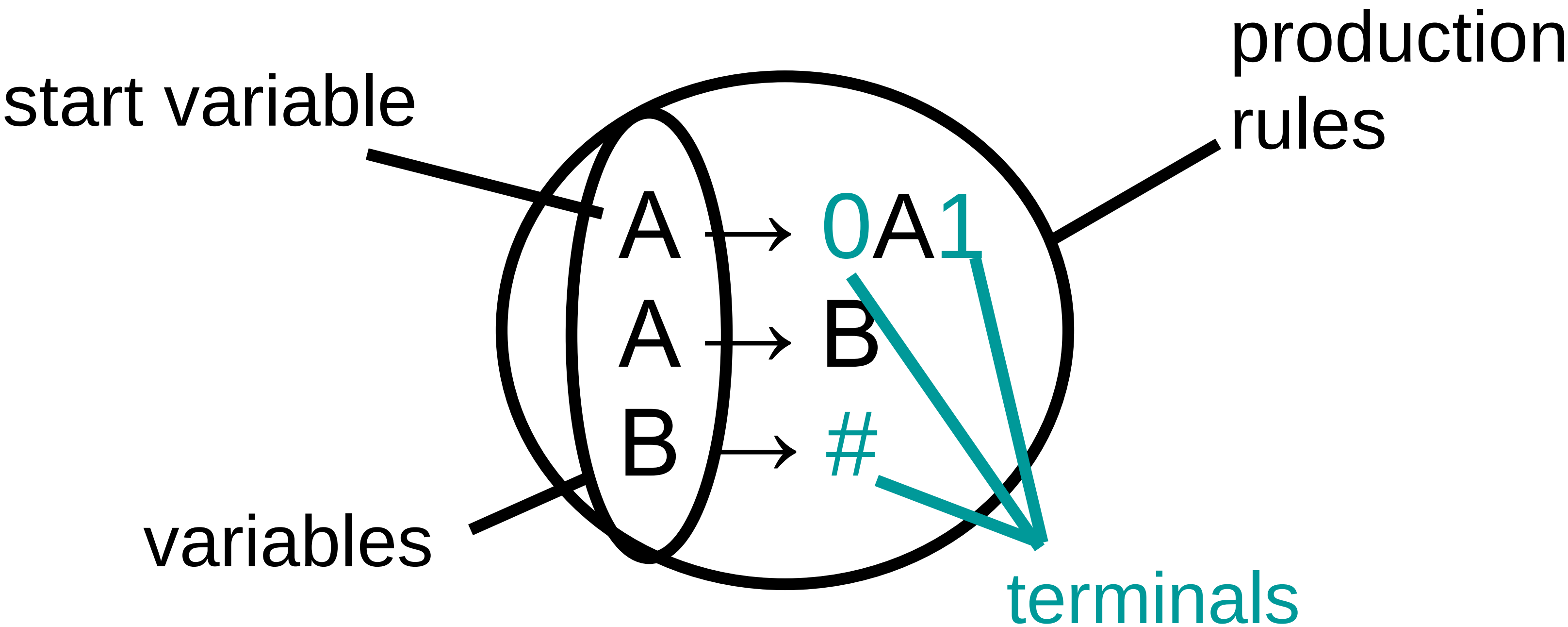
1. ***Place the marker symbol $ onto the stack.***
2. ***Nondeterministically keep reading a character and pushing it onto the stack or go to the next step.***
3. ***Nondeterministically keep reading and popping a matching character or go to the next step.***
4. ***If the top of the stack is $, enter the accept state.***

Build a PDA to recognize

$L = \{ a^i b^j c^k \mid i, j, k \geq 0$ and $(i = j$ or $i = k) \}$



CONTEXT-FREE GRAMMARS (CFGs)



$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$$

CS 332

VALLEY GIRL GRAMMAR

<PHRASE> → <FILLER><PHRASE>

<PHRASE> → <START><END>

<FILLER> → LIKE

<FILLER> → UMM

<START> → YOU KNOW

<START> → ε

<END> → GAG ME WITH A SPOON

<END> → AS IF

<END> → WHATEVER

<END> → LOSER

VALLEY GIRL GRAMMAR

<PHRASE> → <FILLER><PHRASE> | <START><END>

<FILLER> → LIKE | UMM

<START> → YOU KNOW | ε

<END> → GAG ME WITH A SPOON | AS IF | WHATEVER | LOSER

CS 332

Formal Definition

A ***CFG*** is a 4-tuple **G = (V, Σ, R, S)**

V is a finite set of variables

Σ is a finite set of terminals (disjoint from V)

R is set of production rules of the form $A \to W$, where $A \in V$ and $W \in (V \cup \Sigma)^*$

S ∈ V is the start variable

L(G) is the set of strings generated by G

A language is context-free if it is generated by some CFG.

Formal Definition

A ***CFG*** is a 4-tuple **G = (V, Σ, R, S)**

V is a finite set of variables

Σ is a finite set of terminals (disjoint from V)

R is set of production rules of the form $A \to W$, where $A \in V$ and $W \in (V \cup \Sigma)^*$

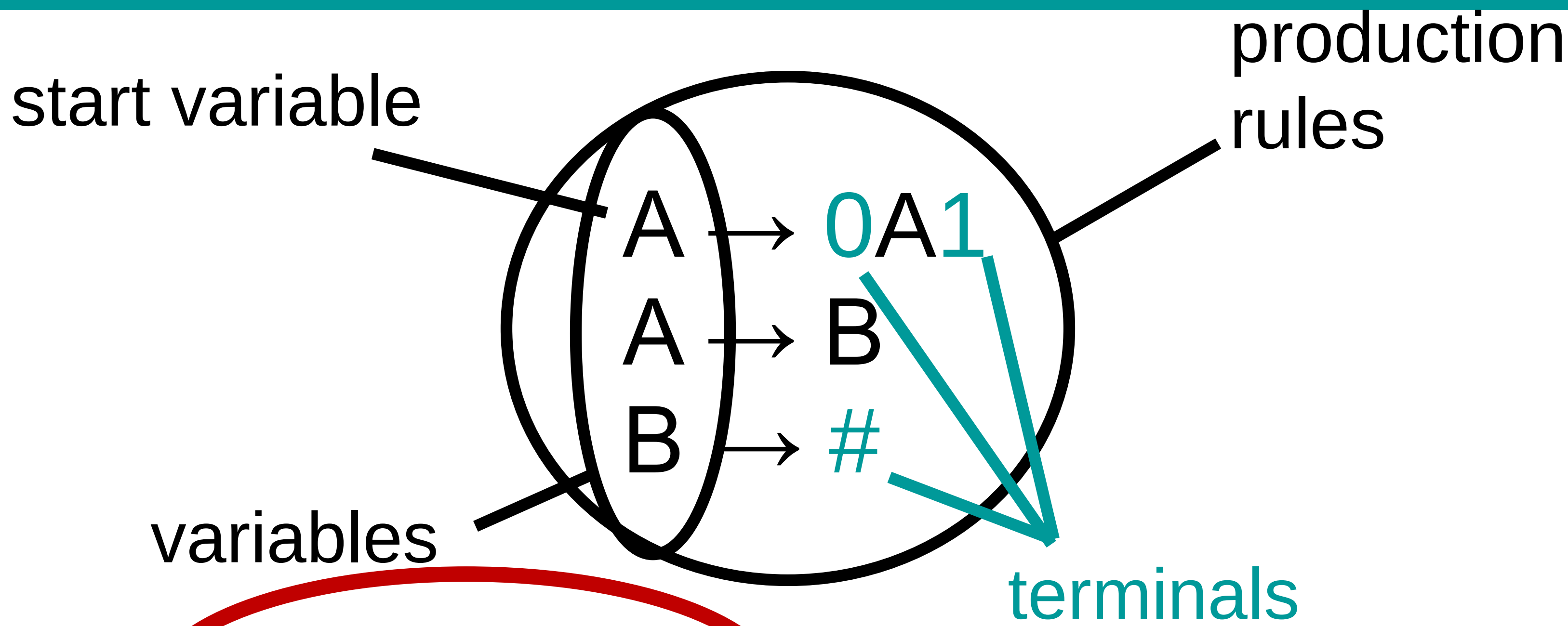
S ∈ V is the start variable

Example: G = {{S}, {0,1}, R, S}   R = { $S \to 0S1$, $S \to \varepsilon$ }

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

CS 332

CFG terminology



$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$

uVw **yields** uvw if $(V \to v) \in R$.

A **derives** 00#11 in 4 steps.

Example

GIVE A CFG FOR EVEN-LENGTH PALINDROMES

$S \to \sigma S \sigma$ for all $\sigma \in \Sigma$

$S \to \varepsilon$

CS 332

Example

GIVE A CFG FOR THE EMPTY SET

$$G = \{ \{S\}, \Sigma, \emptyset, S \}$$

GIVE A CFG FOR…

L_3 = { strings of balanced parens }

$L_4 = \{ a^i b^j c^k \mid i, j, k \geq 0 \text{ and } (i = j \text{ or } j = k) \}$

CFGs in the real world

The syntactic grammar for the Java programming language

BasicForStatement:

for *(; ;) Statement*
for *(; ; ForUpdate) Statement*
for *(; Expression ;) Statement*
for *(; Expression ; ForUpdate) Statement*
for *(ForInit ; ;) Statement*
for *(ForInit ; ; ForUpdate) Statement*
for *(ForInit ; Expression ;) Statement*
for *(ForInit ; Expression ; ForUpdate) Statement*

COMPILER MODULES

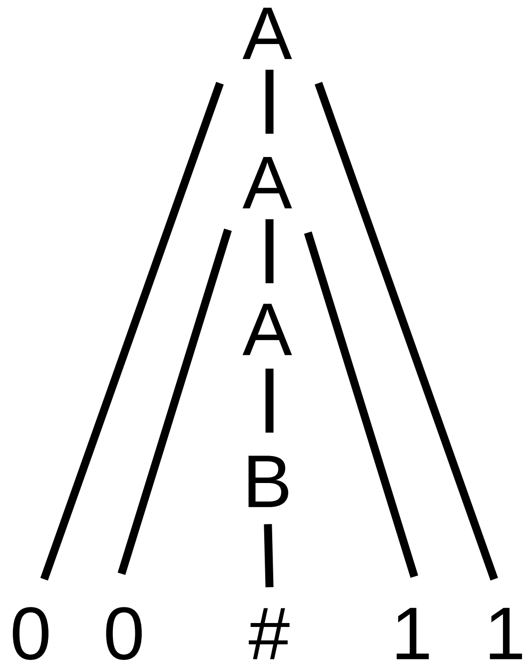
LEXER

PARSER

SEMANTIC ANALYZER

TRANSLATOR/INTERPRETER

Parse trees



$$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$$

Equivalence of CFGs & PDAs

A language is generated by a CFG

$\Leftrightarrow$

It is recognized by a PDA