

# ECE-374-B: Algorithms and Models of Computation, Fall 2024

## Midterm 1 – September 26, 2024

- 
- You can do hard things! Grades do matter, but not as much as you may think, but then life is uncertain anyway, so what.
  - **Don't cheat.** The consequence for cheating is far greater than the reward. Just try your best and you'll be fine.
  - **Please read the entire exam before writing anything.** Most problems have multiple parts. Make sure you check the front and back of all the pages!
  - This is a closed-book exam. At the end of the exam, you'll find a multi-page cheat sheet. *Do not tear out the cheatsheet!* No outside material is allowed on this exam.
  - You should write your answers legibly and in the space given for the question. Overly verbose answers will be penalized.
  - Scratch paper is available on the back of the exam. *Do not tear out the scratch paper!* It messes with the auto-scanner.
  - **You have 75 minutes (1.25 hours) for the exam.** Manage your time well. *Do not spend too much time on questions you do not understand and focus on answering as much as you can!*
  - Proofs are required only if we specifically ask for them. Even then, none of the questions require long inductive proofs. You are only required to give a short explanation of why your answer is correct.
- 

Name: \_\_\_\_\_

NetID: \_\_\_\_\_

## I Short Answer (Regular) - 24 points

Unless the question asks for it, no explanation is required for your answers for full credit. Keep any explanations of your answers to 2 sentences maximum.

- a. Write the recursive definition for the following language ( $\Sigma = \{0, 1\}$ ):

$$L_{1a} = \{w | w \in \Sigma^*, w \text{ is a palindrome ( same left to right and right to left ) } \}^1$$

**Solution:** A string  $w \in \Sigma^*$  is a palindrome if and only if:

- $\varepsilon \in L_{1a}$ , or
- $a \in L_{1a}$  for some  $a \in \Sigma$
- $axa \in L_{1a}$  for some  $a \in \Sigma$  and  $x \in L_{1a}$

■

- b. Write the regular expression for the following languages ( $\Sigma = \{0, 1\}$ ):

i  $L_{1bi} = \{w | w \in \Sigma^*, w \text{ does not contain the subsequence } 010\}$

**Solution:**  $1^*0^*1^*$

■

ii  $L_{1bii} = \{w | w \in \Sigma^*, w \text{ is any string except the string "1"}\}$

**Solution:**  $\varepsilon + 0 + (0 + 1)(0 + 1)^+$

■

- c. What is the minimum number of states a DFA would need to decide if a string belongs to the language  $L = 0^{374}1^{473}2^*$  ( $\Sigma = \{0, 1, 2\}$ )?

**Solution:** Other than "normal" states that keep track of the length of 0 and 1, we need a starting state and an extra rejecting state. So the total number of states is  $374 + 473 + 2 = 849$ .

■

<sup>1</sup> $\varepsilon$ , "0", and "1" are a part of this language.

## 2 Short Answer (Context-free) - 16 points

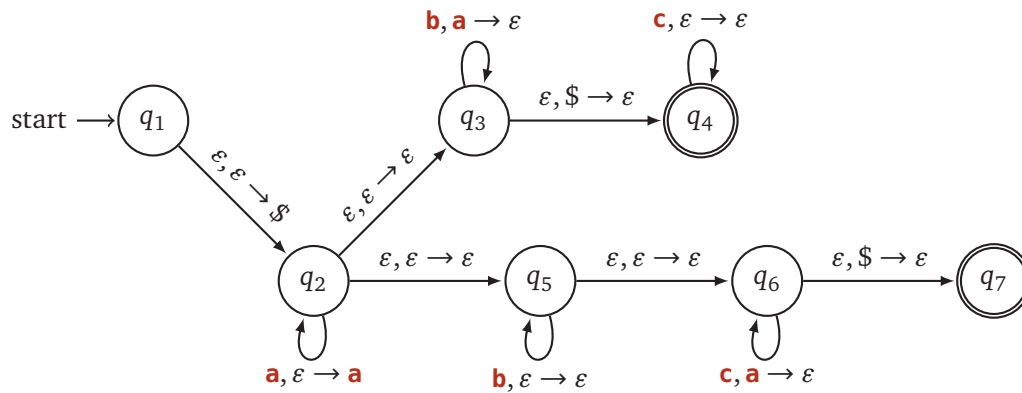
Unless the question asks for it, no explanation is required for your answers for full credit. Keep any explanations of your answers to 2 sentences maximum.

a. Provide the context-free grammar for the following language:

$$L_{2a} = \{w \mid w \in \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}^*, w = a^i b^j c^k \text{ where } k \geq i + j\}$$

**Solution:**  $V = \{S, A, B\}$ ,  $T = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ ,  $P = \{S \rightarrow \mathbf{aSc} \mid A, A \rightarrow \mathbf{bAc} \mid B, B \rightarrow \mathbf{cBc}\}$ ,  $S \rightarrow S$  ■

b. Succinctly describe the language described by the following PDA ( $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ ):



**Solution:**  $L = \{\mathbf{a}^i \mathbf{b}^j \mathbf{c}^k \mid i = j \text{ or } i = k\}$  ■

### 3 Language Transformation - 15 points

Assume  $L$  is a regular language and  $\Sigma = \{0, 1\}$ . Assume zero-indexing (first bit is at position “[0]”).

Prove that the language  $delete2\mathbf{1}'s(L) := \{xyz \mid x\mathbf{1}y\mathbf{1}z \in L\}$  is regular.

**Solution:** Let  $M = (Q, s, A, \delta)$  be a DFA that accepts  $L$ . We construct an NFA  $M' = (Q', s', A', \delta')$  with  $\varepsilon$ -transitions that accepts  $delete2\mathbf{1}'s(L)$  as follows. Intuitively,  $M'$  simulates  $M$ , but inserts two  $\mathbf{1}$ 's into  $M$ 's input string at a non-deterministically chosen location.

- The state  $(q, o)$  means (the simulation of)  $M$  is in state  $q$  and  $M'$  has not yet inserted a  $\mathbf{1}$ .
- The state  $(q, 1)$  means (the simulation of)  $M$  is in state  $q$  and  $M'$  has already inserted one  $\mathbf{1}$ .
- The state  $(q, 2)$  means (the simulation of)  $M$  is in state  $q$  and  $M'$  has already inserted two  $\mathbf{1}$ 's.

$$Q' := Q \times \{o, 1, 2\}$$

$$s' := (s, o)$$

$$A' := \{(q, 2) \mid q \in A\}$$

$$\delta'((q, o), \varepsilon) = \{(\delta(q, \mathbf{1}), 1)\}$$

$$\delta'((q, 1), \varepsilon) = \{(\delta(q, \mathbf{1}), 2)\}$$

$$\delta'((q, 2), \varepsilon) = \emptyset$$

$$\delta'((q, o), a) = \{(\delta(q, a), o)\}$$

$$\delta'((q, 1), a) = \{(\delta(q, a), 1)\}$$

$$\delta'((q, 2), a) = \{(\delta(q, a), 2)\}$$

■

## 4 Language classification I (2 parts) - 15 points

Let  $\Sigma_4 = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\}$  and each row of the string represent a binary number.

$L_4 = \{w \in \Sigma^* \mid \text{the top row of } w \text{ is twice the value of the bottom row.}\}.$

For the sake of simplicity, you may assume a binary number may (but does not have to) begin with a 0. As an example, the string  $\begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$  is in the language but the string

$\begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$  is not.

- a. Is  $L_4$  regular? Indicate whether or not by circling one of the choices below. Either way, prove it.

regular      not regular

**Solution:** The language  $L_4$  is regular. Intuitively, we notice we only have keep track of whether the top row is "shifted" to the left of the bottom row by one index, suggesting we only need a small (finite) amount of memory to verify a string. We can describe  $L_4$  with the following regular expression.

$$L_4 = \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix}^* \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right)^*$$

Alternatively, a DFA with 3 states can be constructed with appropriate transitions (the TA who wrote this solution thought of a DFA with an accepting state, an intermediate state, and a fail state). ■

- b. Is  $L_4$  context-free? Indicate whether or not by circling one of the choices below. Either way, prove it.

context-free      not context-free

**Solution:** The language  $L_4$  is context-free. This comes immediately from the fact that  $L_4$  is regular. ■

## 5 Language classification II (2 parts) - 15 points

Let  $\Sigma_5 = \{0, 1\}$  and

$$L_5 = \{x0y \mid x, y \in \Sigma_5^*, \#_1(x) \geq \#_1(y)\}^{2,3}$$

- a. Is  $L_5$  regular? Indicate whether or not by circling one of the choices below. Either way, prove it.

regular

☒ not regular

**Solution:** The language  $L_5$  is not regular. Let  $F = 1^*0$ . Let  $x = 1^i0$  and  $y = 1^j0$  be two distinct strings from  $F$ . Without loss of generality, assume that  $i > j$ . Let  $z = 1^i$ . Then  $xz = 1^i01^i$  is in  $L_5$ , while  $yz = 1^j01^i$  is not in  $L_5$ . Since  $F$  is a valid and infinite fooling set of  $L_5$ , we conclude that  $L_5$  is not regular. ■

- b. Is  $L_5$  context-free? Indicate whether or not by circling one of the choices below. Either way, prove it.

☒ context-free

not context-free

**Solution:** The language  $L_5$  is context-free, since the following CFG represents  $L_5$ .

$$S \rightarrow 1S \mid 0S \mid 1S1 \mid S0 \mid 0$$

■

<sup>2</sup> $x$  has at least as many 1's as  $y$

<sup>3</sup>The  $\#_a(w)$  operator counts the number of times character  $a$  appears in string  $w$

## 6 Language classification III (2 parts) - 15 points

Let  $\Sigma_6 = \{0, 1\}$  and

$$L_6 = \{w \in \{0, 1\}^n \mid w \text{ is a palindrome and } 0 \leq n \leq 4\}$$

- a. Is  $L_6$  regular? Indicate whether or not by circling one of the choices below. Either way, prove it.

☒ regular

☐ not regular

**Solution:** The language  $L_6$  is regular. There are a finite number of palindromes of length  $0 \leq n \leq 4$ .

$$\begin{aligned} &\epsilon \\ &+ 0 + 1 \\ &+ 00 + 11 \\ &+ 000 + 111 + 010 + 101 \\ &+ 0000 + 1111 + 0110 + 1001 \end{aligned}$$

■

- b. Is  $L_6$  context-free? Indicate whether or not by circling one of the choices below. Either way, prove it.

☒ context-free

☐ not context-free

**Solution:** By definition, every regular language is a context-free language. ■

*This page is for additional scratch work!*



# ECE 374 B Language Theory: Cheatsheet

## 1 Languages and strings

### Languages

- An *alphabet*  $\Sigma$  is a **finite** set of symbols.

**Definitions** A *string* in  $\Sigma^*$  is a **finite** sequence of symbols in  $\Sigma$ .

- A *language* is  $L$  is a set of strings over some alphabet.

All languages represent mathematical problems.  
Example: multiplication of two integers:

$$L_{MULT2} = \left\{ \begin{array}{ccc} 1 \times 1|1, & 1 \times 2|2, & 1 \times 3|3, \dots \\ 2 \times 1|2, & 2 \times 2|4, & 2 \times 3|6, \dots \\ \vdots & \vdots & \vdots \\ n \times 1|n, & n \times 2|2n, & n \times 3|3n, \dots \end{array} \right\} \quad (1)$$

#### Language operations

- For languages  $A, B$  the *concatenation* of  $A, B$  is  $AB = \{xy \mid x \in A, y \in B\}$ .
- For languages  $A, B$ , their *union* is  $A \cup B$ , *intersection* is  $A \cap B$ , and *difference* is  $A \setminus B$  (also written as  $A - B$ ).
- For language  $A \subseteq \Sigma^*$  the *complement* of  $A$  is  $\bar{A} = \Sigma^* \setminus A$ .
- $\Sigma^n$  is the set of all strings of length  $n$ .
- $\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$  is the set of all strings over  $\Sigma$ .
- $\Sigma^+ = \bigcup_{n \geq 1} \Sigma^n$  is the set of non-empty strings over  $\Sigma$ .

### Strings

- The *length* of a string  $w$  (denoted by  $|w|$ ) is the number of symbols in  $w$ .
- For integer  $n \geq 0$ ,  $\Sigma^n$  is set of all strings over  $\Sigma$  of length  $n$ .  $\Sigma^*$  is the set of all strings over  $\Sigma$ .

#### Definitions

- $\Sigma^*$  is the set of all strings of all lengths including empty string.
- $\epsilon$  is a *string* containing no symbols.
- $\emptyset$  is the *empty set*. It contains no strings.

- If  $x$  and  $y$  are strings then  $xy$  denotes their concatenation. Recursively:

- $xy = y$  if  $x = \epsilon$
- $xy = a(wy)$  if  $x = aw$

- $v$  is *substring* of  $w \iff$  there exist strings  $x, y$  such that  $w = xvy$ .

- If  $x = \epsilon$  then  $v$  is a *prefix* of  $w$
- If  $y = \epsilon$  then  $v$  is a *suffix* of  $w$

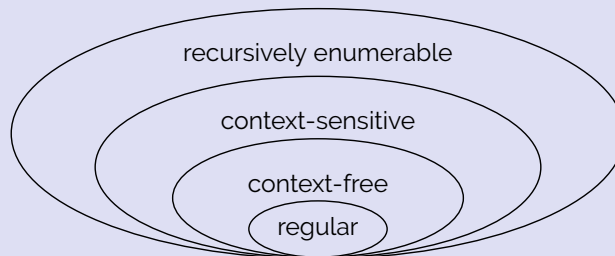
- A *subsequence* of a string  $w = w_1w_2 \dots w_n$  is either a subsequence of  $w_2 \dots w_n$  or  $w_1$  followed by a subsequence of  $w_2 \dots w_n$ .

- If  $w$  is a string then  $w^n$  is defined inductively as follows:  
 $w^n = \epsilon$  if  $n = 0$  or  $w^n = ww^{n-1}$  if  $n > 0$

#### String operations

## 2 Overview of language complexity

### Overview



Grammar	Languages	Production Rules	Automaton	Examples
Type-0	recursively enumerable	$\gamma \rightarrow \alpha$ (no constraints)	Turing machine	$L = \{w \mid w \text{ is a TM which halts}\}$
Type-1	context-sensitive	$\alpha A \beta \rightarrow \alpha \gamma \beta$	linear bounded nondeterministic Turing machine	$L = \{a^n b^n c^n \mid n > 0\}$
Type-2	context-free	$A \rightarrow \alpha$	nondeterministic pushdown automata	$L = \{a^n b^n \mid n > 0\}$
Type-3	regular	$A \rightarrow aB$	finite state machine	$L = \{a^n \mid n > 0\}$

Meaning of symbols:

- $a$  - terminal
- $A, B$  - variables
- $\alpha, \beta, \gamma$  - strings in  $\{a \cup A\}^*$  where  $\alpha, \beta$  are maybe empty,  $\gamma$  is never empty

<sup>a</sup>Table borrowed from Wikipedia: [https://en.wikipedia.org/wiki/Chomsky\\_hierarchy](https://en.wikipedia.org/wiki/Chomsky_hierarchy)

### 3 Regular languages

#### Regular language - overview

A language is regular if and only if it can be obtained from finite languages by applying

- union,
- concatenation or
- Kleene star

finitely many times. All regular languages are representable by regular grammars, DFAs, NFAs and regular expressions.

#### Regular expressions

Useful shorthand to denotes a language.

A *regular expression*  $r$  over an alphabet  $\Sigma$  is one of the following:

**Base cases:**

- $\emptyset$  the language  $\emptyset$
- $\varepsilon$  denotes the language  $\{\varepsilon\}$
- $a$  denote the language  $\{a\}$

**Inductive cases:** If  $r_1$  and  $r_2$  are regular expressions denoting languages  $L_1$  and  $L_2$  respectively (i.e.,  $L(r_1) = L_1$  and  $L(r_2) = L_2$ ) then,

- $r_1 + r_2$  denotes the language  $L_1 \cup L_2$
- $r_1 \cdot r_2$  denotes the language  $L_1 L_2$
- $r_1^*$  denotes the language  $L_1^*$

**Examples:**

- $0^*$  - the set of all strings of 0s, including the empty string
- $(00000)^*$  - set of all strings of 0s with length a multiple of 5
- $(0 + 1)^*$  - set of all binary strings

#### Nondeterministic finite automata

NFAs are similar to DFAs, but may have more than one transition destination for a given state/character pair.

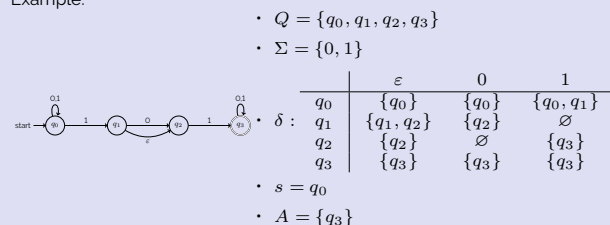
An NFA  $N$  *accepts a string*  $w$  iff some accepting state is reached by  $N$  from the start state on input  $w$ .

The *language accepted (or recognized)* by an NFA  $N$  is denoted  $L(N)$  and defined as  $L(N) = \{w \mid N \text{ accepts } w\}$ .

A *nondeterministic finite automaton (NFA)*  $N = (Q, \Sigma, s, A, \delta)$  is a five tuple where

- $Q$  is a finite set whose elements are called *states*
- $\Sigma$  is a finite set called the *input alphabet*
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \rightarrow \mathcal{P}(Q)$  is the *transition function* (here  $\mathcal{P}(Q)$  is the power set of  $Q$ )
- $s$  and  $\Sigma$  are the same as in DFAs

Example:



For NFA  $N = (Q, \Sigma, \delta, s, A)$  and  $q \in Q$ , the  $\varepsilon$ -reach( $q$ ) is the set of all states that  $q$  can reach using only  $\varepsilon$ -transitions.

Inductive definition of  $\delta^* : Q \times \Sigma^* \rightarrow \mathcal{P}(Q)$ :

- if  $w = \varepsilon$ ,  $\delta^*(q, w) = \varepsilon\text{-reach}(q)$
- if  $w = a$  for  $a \in \Sigma$ ,  $\delta^*(q, a) = \varepsilon\text{-reach}\left(\bigcup_{p \in \varepsilon\text{-reach}(q)} \delta(p, a)\right)$
- if  $w = ax$  for  $a \in \Sigma, x \in \Sigma^*$ :  $\delta^*(q, w) = \varepsilon\text{-reach}\left(\bigcup_{p \in \varepsilon\text{-reach}(q)} \left(\bigcup_{r \in \delta^*(p, a)} \delta^*(r, x)\right)\right)$

#### Regular closure

Regular languages are closed under union, intersection, complement, difference, reversal, Kleene star, concatenation, etc.

#### Deterministic finite automata

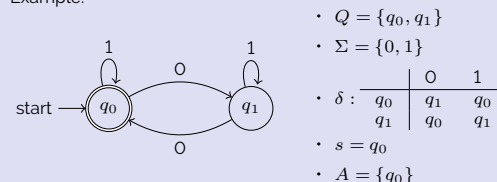
DFAs are finite state machines that can be represented as a directed graph or in terms of a tuple.

The *language accepted (or recognized)* by a DFA  $M$  is denoted by  $L(M)$  and defined as  $L(M) = \{w \mid M \text{ accepts } w\}$ .

A *deterministic finite automaton (DFA)*  $M = (Q, \Sigma, s, A, \delta)$  is a five tuple where

- $Q$  is a finite set whose elements are called *states*
- $\Sigma$  is a finite set called the *input alphabet*
- $\delta : Q \times \Sigma \rightarrow Q$  is the *transition function*
- $s \in Q$  is the *start state*
- $A \subseteq Q$  is the set of *accepting/final states*

Example:



Every string has a unique walk along a DFA. We define the extended transition function as  $\delta^* : Q \times \Sigma^* \rightarrow Q$  defined inductively as follows:

- $\delta^*(q, w) = q$  if  $w = \varepsilon$
- $\delta^*(q, w) = \delta^*(\delta(q, a), x)$  if  $w = ax$ .

Can create a larger DFA from multiple smaller DFAs. Suppose

- $L(M_0) = \{w \text{ has an even number of 0s}\}$  (pictured above) and
- $L(M_1) = \{w \text{ has an even number of 1s}\}$ .

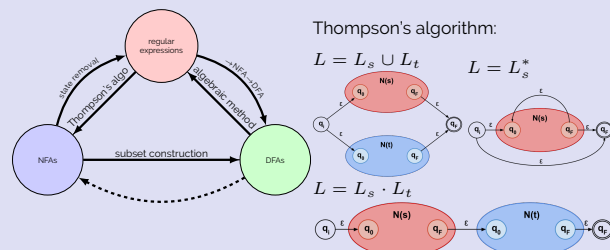
$L(M_C) = \{w \text{ has even number of 0s and 1s}\}$

Suppose  $M_0 = (Q_0, \Sigma, s_0, A_0, \delta_0)$  and  $M_1 = (Q_1, \Sigma, s_1, A_1, \delta_1)$ . Then

- $Q = Q_0 \times Q_1 = \{(q_0, q_1) \mid q_0 \in Q_0, q_1 \in Q_1\}$
- $s = (s_0, s_1)$
- $\delta : Q \times \Sigma \rightarrow Q$ , where  $\delta((q_0, q_1), a) = (\delta_0(q_0, a), \delta_1(q_1, a))$
- $A = \{(q_0, q_1) \mid q_0 \in A_0 \text{ and } q_1 \in A_1\}$

#### Regular language equivalences

A regular language can be represented by a regular expression, regular grammar, DFA and NFA.



**Arden's rule:** If  $R = Q + RP$  then  $R = QP^*$ .

#### Fooling sets

Some languages are not regular (Ex.  $L = \{0^n 1^n \mid n \geq 0\}$ ).

Two states  $p, q \in Q$  are *distinguishable* if there exists a string  $w \in \Sigma^*$ , such that

$$\delta^*(p, w) \in A \text{ and } \delta^*(q, w) \notin A.$$

or

Two states  $p, q \in Q$  are *equivalent* if for all strings  $w \in \Sigma^*$ , we have that

$$\delta^*(p, w) \in A \iff \delta^*(q, w) \in A.$$

$$\delta^*(p, w) \notin A \text{ and } \delta^*(q, w) \in A.$$

For a language  $L$  over  $\Sigma$  a set of strings  $F$  (could be infinite) is a *fooling set* or *distinguishing set* for  $L$  if every two distinct strings  $x, y \in F$  are distinguishable.

## 4 Context-free languages

### Context-free languages

A language is context-free if it can be generated by a context-free grammar. A context-free grammar is a quadruple  $G = (V, T, P, S)$

- $V$  is a finite set of *nonterminal (variable) symbols*
- $T$  is a finite set of *terminal symbols* (alphabet)
- $P$  is a finite set of *productions*, each of the form  $A \rightarrow \alpha$  where  $A \in V$  and  $\alpha$  is a string in  $(V \cup T)^*$ . Formally,  $P \subseteq V \times (V \cup T)^*$ .
- $S \in V$  is the *start symbol*

Example:  $L = \{ww^R \mid w \in \{0, 1\}^*\}$  is described by  $G = (V, T, P, S)$  where  $V, T, P$  and  $S$  are defined as follows:

- $V = \{S\}$
- $T = \{0, 1\}$
- $P = \{S \rightarrow \varepsilon \mid 0S0 \mid 1S1\}$   
(abbreviation for  $S \rightarrow \varepsilon, S \rightarrow 0S0, S \rightarrow 1S1$ )
- $S = S$

### Pushdown automata

A pushdown automaton is an NFA with a stack.

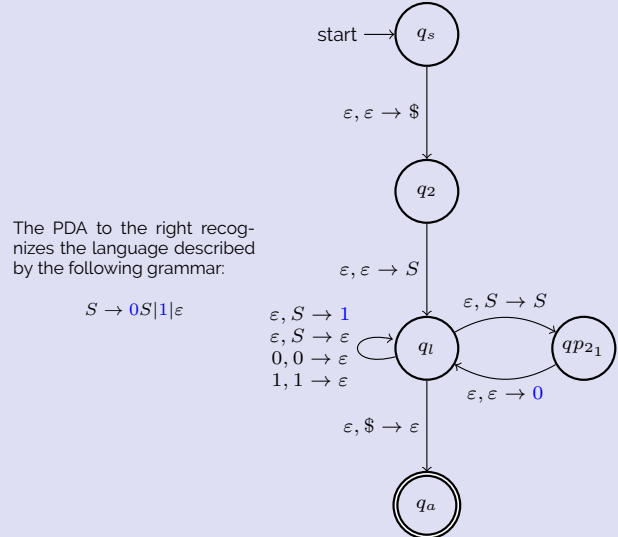
The language  $L = \{0^n 1^n \mid n \geq 0\}$  is recognized by the pushdown automaton:

A *nondeterministic pushdown automaton (PDA)*  $P = (Q, \Sigma, \Gamma, \delta, s, A)$  is a **six** tuple where

- $Q$  is a finite set whose elements are called *states*
- $\Sigma$  is a finite set called the *input alphabet*
- $\Gamma$  is a finite set called the *stack alphabet*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$  is the *transition function*
- $s$  is the start state
- $A$  is the set of accepting states

In the graphical representation of a PDA, transitions are typically written as  $\langle \text{input read} \rangle, \langle \text{stack pop} \rangle \rightarrow \langle \text{stack push} \rangle$ .

A CFG can be converted to a pushdown automaton.



### Context-free closure

Context-free languages are closed under union, concatenation, and Kleene star.

They are **not** closed under intersection or complement.