

ECE-374-B: Algorithms and Models of Computation, Fall 2023
Midterm 1 – September 21, 2023

- 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.** There are 5 problems and most have multiple parts.
 - 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: _____

Date: _____

1 Short Answer (Regular) (2 parts) - 20 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 ($\Sigma = \{0, 1\}$):

$$L_{1a} = \{w | w \in \Sigma^*, w \text{ has alternating } 0\text{'s and } 1\text{'s}\}^1$$

Solution: Very similar to the Lab and HW problem. We need to define two languages (L_{1aA} and L_{1aB}) and combine them in the end.

$$\begin{aligned} \varepsilon &\in L_{1aA} \\ \varepsilon &\in L_{1aB} \\ 1x &\in L_{1aA} \text{ for } x \in L_{1aB} \\ 0x &\in L_{1aB} \text{ for } x \in L_{1aA} \\ L_{1a} &= L_{1aA} \cup L_{1aB} \end{aligned}$$

■

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

i $L_{1bi} = \{w | w \in \Sigma^*, w \text{ does not contain the subsequence } 00\}$

Solution: This is another way of saying that the strings in the language can include at most a single 0.

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

■

ii $L_{1bii} = \{w | w \in \Sigma^*, w \text{ has alternating } 0\text{'s and } 1\text{'s}\}^2$

Solution: We have seen this in lecture. $(\varepsilon + 1)(01)^*(\varepsilon + 0)$

■

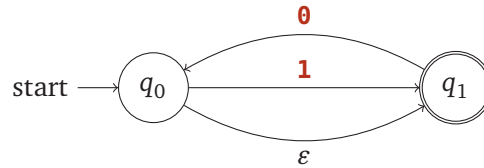
¹ ε , "0", and "1" are a part of this language.

²See previous footnote. Also note that recursive definitions and regular expressions are different things.

2 Short Answer II (2 parts) - 20 points

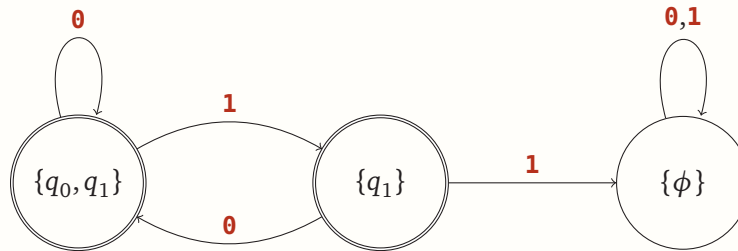
Unless the question asks for it, no explanation is required for your answers for full credit. Keep any explanations of your answers to 2 sentences maximum.

a. Consider the NFA ($\Sigma = \{0, 1\}$):



Draw the equivalent DFA:

Solution: We can draw the DFA as follows:



- b. Consider the language $(\Sigma = \{0, 1\})$ represented by the regular expression $r_{2b} = 1^*$. Show that a DFA that represents this language ($L(r_{2b})$) must have at least two states (Hint: think about fooling sets).

Solution: We went through this problem in the prelecture problem of Lecture 6. This problem was simply meant to test if you understood the idea of distinguishable strings and what fooling sets are. There are many correct answers for this. Here's one:

Consider the fooling set $F = \{\varepsilon, 0\}$. Suppose $x = \varepsilon$ and $y = 0$ and let's assume suffix $z = 1$. Therefore we know the following:

$$\begin{array}{ll}xz = 1 & \in L(r_{2b}) \\yz = 01 & \notin L(r_{2b})\end{array}$$

All the strings in the fooling set are distinguishable and therefore the DFA that represents $L(r_{2b})$ must have at least $|F|$ states: $|Q| \geq |F|$. Therefore the DFA that represents $L(r_{2b})$ must be ≥ 2 . ■

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 $FLIPEVERYTHIRDCHAR(L) := \{flipThirdChars(w) \mid w \in L\}$ is regular.

$$flipThirdChars = (\underline{000}\underline{111}\underline{100}\underline{011}) = \underline{100}\underline{011}\underline{000}\underline{111}$$

Intuitively, $flipThirdChars$ changes every third character in the input alphabet to its bit-wise complement.

Solution: We can prove that the above language is regular by doing a transformation. It is very important to state why you are doing the transformation or fooling set or grammar as I have stated many times during lecture. In the case of this problem we need to state the following:

1. We know L is regular. Therefore there is a DFA M where $L(M) \equiv L$.
2. To show $FLIPEVERYTHIRDCHAR(L)$ is regular, we will transform M into a NFA N where $L(N) \equiv FLIPEVERYTHIRDCHAR(L)$.
3. Since $FLIPEVERYTHIRDCHAR(L)$ is representable by a NFA, it is regular.

So now we just have to show the transformation. We’ve seen a similar transformations in Lab-3 so let’s follow that layering format.

Let $M = (Q, s, A, \delta)$ be a DFA that accepts L . We construct a new DFA $M' = (Q', s', A', \delta')$ that accepts $FLIPEVERYTHIRDCHAR(L)$ as follows.

To keep track of if the index is even/odd, we cross the original states Q with the set $\{0, 1, 2\}$. Then every time an input is processed we flip this third coordinate. The starts state is $(s, 0)$. Effectively this is a flag determining if it is divisible by three.

To flip the bits on third indexes, we define the transition of 0-indexed bits (i.e. (q, x)) as the transition of the original DFA with a flipped input and the 1/2-indexed bits (i.e. (q, x)) as the transition of the original DFA with the same input.

$$\begin{aligned} Q' &= Q \times \{0, 1, 2\} \\ s' &= (s, 0) \\ A' &= A \times \{0, 1, 2\} \\ \delta'((q, 0), \mathbf{0}) &= (\delta(q, \mathbf{1}), 1) \\ \delta'((q, 0), \mathbf{1}) &= (\delta(q, \mathbf{0}), 1) \\ \delta'((q, 1), \mathbf{a}) &= (\delta(q, \mathbf{a}), 2) \\ \delta'((q, 2), \mathbf{a}) &= (\delta(q, \mathbf{a}), 0) \end{aligned}$$

■

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.

$L_4 = \{w \in \Sigma^* \mid \text{the top row of } w \text{ is a larger number than is the bottom row.}\}$.

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 “ $\begin{bmatrix} 0 \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ ” 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.

Solution: regular not regular

Think about what makes one binary number larger than the other. It is whether when reading two numbers left to right, and the first difference between the two numbers you see is that 1 in the n th-digit place for binary number n_1 , but not for binary n_2 , then that means $n_1 > n_2$.

The idea here is that when reading the matrix left to right, if the first difference between the top and bottom rows is the character $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, then the top row number is larger than the bottom row number. That means that we can describe the language using the following regular expression:

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

Because this expression represents all the strings that are part of this language, this language must be regular. ■

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

Solution: context-free not context-free

All regular languages are context-free. ■

5 Language classification II (2 parts) - 15 points

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

$$L_5 = \{xy^R \mid x, y \in \Sigma_5^*, |x| = |y| \text{ but } x \neq y\}$$

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

Solution: regular not regular

Assume L_5 is regular. Then it must be representable by a DFA $((Q, \Sigma, \delta, s, A))$.

Consider the fooling set $F = \{0^{2i}1 \mid i \geq 0\}$. Let's take two strings $a = 0^{2i}1$ and $b = 0^{2j}1$ ($i \neq j$) and suffix $c = 10^{2i}$.

Then $ac = 0^{2i}110^{2i} \notin L_5$ because $x = y$.

On the other hand $bc = 0^{2j}110^{2i}$ which is in the language since the x - y split must somewhere other than in between the ones which would make either x contain the **11** substring and y be all zeros, or vice versa.

Therefore, $|Q| > |F| = \infty \dots$ Contradiction. ■

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

Solution: context-free not context-free

You can prove a language is CF by either showing that it can be represented by a PDA, or CFG. CFG way seems a bit easier so let's do that:

We can construct a context-free grammar that represents the above language. The key idea is that atleast one of the bit pairs must differ. Knowing this, the CFG can be something along the lines of:

$$S \rightarrow 0S0 \mid 1S1 \mid 0A1 \mid 1A0$$

$$A \rightarrow 0A0 \mid 0A1 \mid 1A0 \mid 1A1 \mid \varepsilon$$

You go from variable S to variable A when the n -th rightmost bit in x differs from the n -th rightmost bit in y (which is the n -th leftmost bit in y^R). Then you can fill the two strings with any other characters (while keeping $|x| = |y|$) and then you end with transforming $A \rightarrow \varepsilon$. Because the language is representable by a CFG, then it is context-free. ■

6 Language classification III (2 parts) - 15 points

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

$$L_6 = \{xy \mid x, y \text{ are unary numbers and } x > y\}^3$$

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

Solution: regular not regular

So x is a bunch of **1**'s and y is slightly fewer **1**'s meaning the two concatenated together become **1**⁺.

Another way to think about it is that y can be empty string so this language is all the runs of **1**'s. ■

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

Solution: context-free not context-free

All regular languages are context-free. ■

³Remember from Lab 0, unary numbers (also known as tick marks) are base 1 where a number n is represented by **1** ^{n} . So numbers $[0, 1, 2, 3, \dots]_{10} = [{}^{\prime}, {}^{\prime}1, {}^{\prime}11, {}^{\prime}111, \dots]_1$ respectively.

This page is for additional scratch work!

ECE 374 B Language Theory: Cheatsheet

1 Languages and strings

Languages

- An alphabet Σ is a **finite** set of symbols.

Definitions A string in Σ^* is a **finite** sequence of symbols in Σ .

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

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

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

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 $\bar{A} = \Sigma^* \setminus A$.
- Σ^n is the set of all strings of length n .
- $\Sigma^* = \cup_{n \geq 0} \Sigma^n$ is the set of all strings over Σ .
- $\Sigma^+ = \cup_{n \geq 1} \Sigma^n$ is the set of non-empty strings over Σ .

Strings

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

Definitions

- Σ^* is the set of all strings of all lengths including empty string.
- ϵ 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

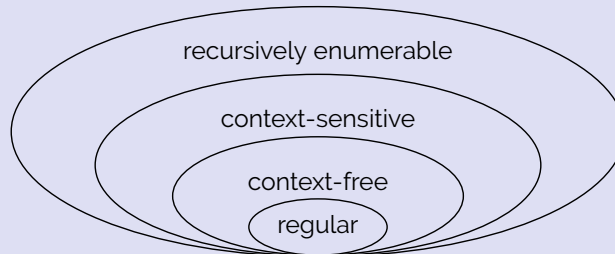
- A *subsequence* of a string $w = w_1w_2 \dots w_n$ is either a subsequence of $w_2 \dots w_n$ or w_1 followed by a subsequence of $w_2 \dots w_n$.

- 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
- α, β, γ - strings in $\{a \cup A\}^*$ where α, β are maybe empty, γ is never empty

^aTable borrowed from Wikipedia: 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 Σ is one of the following:

Base cases:

- \emptyset the language \emptyset
- ϵ denotes the language $\{\epsilon\}$
- 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.

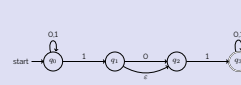
An NFA N accepts a string w iff some accepting state is reached by N from the start state on input w .

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

Example:



	ϵ	0	1
$\delta :$	q_0	$\{q_0\}$	$\{q_0\}$
	q_1	$\{q_1, q_2\}$	$\{q_2\}$
	q_2	$\{q_2\}$	$\{q_3\}$
	q_3	$\{q_3\}$	$\{q_3\}$

- $s = q_0$
- $A = \{q_3\}$

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

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

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

Regular closure

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

Deterministic finite automata

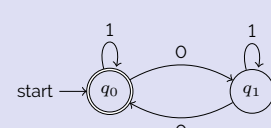
DFAs are finite state machines that can be represented as a directed graph or in terms of a tuple.

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
- Σ 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:



- $Q = \{q_0, q_1\}$
- $\Sigma = \{0, 1\}$

$\delta :$		0	1
	q_0	q_1	q_0
	q_1	q_0	q_1

- $s = q_0$
- $A = \{q_0\}$

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 $w = \epsilon$
- $\delta^*(q, w) = \delta^*(\delta(q, a), x)$ 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
- $L(M_1) = \{w \text{ has an even number of 1s}\}$.

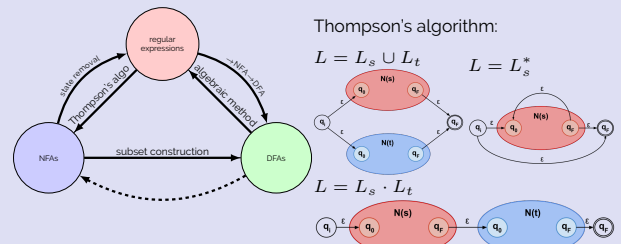
$L(M_C) = \{w \text{ has even number of 0s and 1s}\}$

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.



Thompson's algorithm:

$$L = L_s \cup L_t \quad L = L_s^*$$

$$L = L_s \cdot L_t$$

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

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

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

For a language L over Σ 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 α 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.

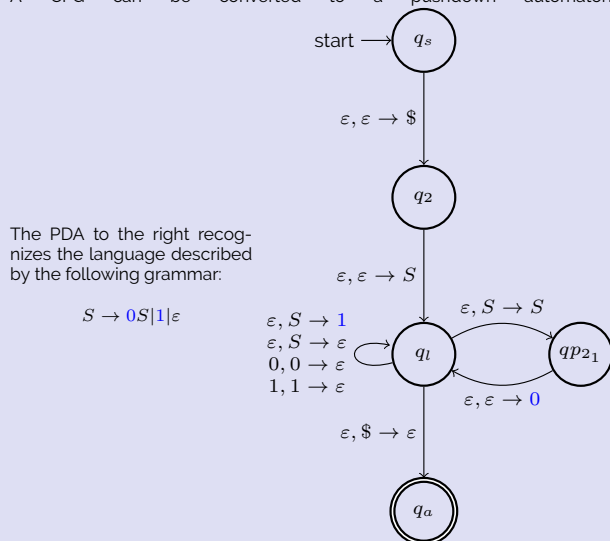
The language $L = \{0^n 1^n \mid n \geq 0\}$ is recognized by the pushdown automaton:

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*
- Σ is a finite set called the *input alphabet*
- Γ 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 (input read), (stack pop) \rightarrow (stack push).

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.