

# ECE-374-B: Algorithms and Models of Computation, Fall 2025

## Midterm 1 – September 25, 2025

- 
- 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!* It messes with the auto-scanner. 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: \_\_\_\_\_

## 1 Short Answer (Regular) - 15 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 ( $\Sigma = \{0, 1\}$ ):

$$L_{1a} = \{0^n 1^n | n \geq 0\}$$

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

i  $L_{1bi} = \{w | w \in \Sigma^*, w \text{ does not contain the subsequence } 0000\}$

ii  $L_{1bii} = \{w | w \in \Sigma^*, w \text{ all strings with alternating } 0\text{'s and } 1\text{'s}\}$ <sup>1</sup>

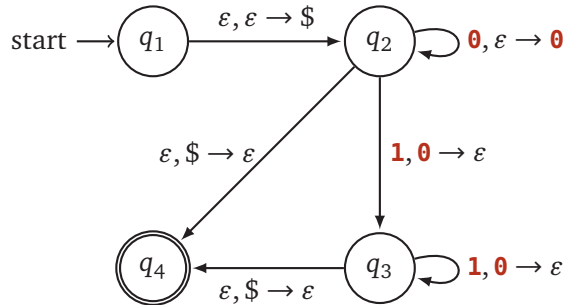
---

<sup>1</sup>strings where no adjacent symbols are the same

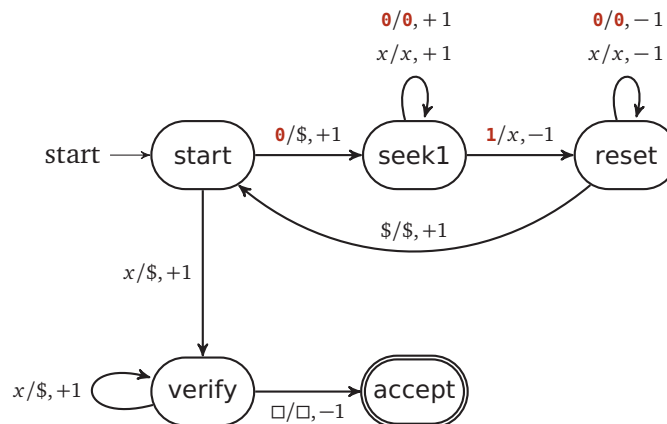
## 2 Short Answer (Automata fun) - 10 points

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. What language does the push-down automata shown below represent?<sup>2</sup>



b. What language does the Turing machine below represent?



<sup>2</sup>I don't want an English description of what the PDA does, I want a definition of a language:  $L = \{.. \}$

### 3 Checking transformer models- 15 points

I want to prove that  $L' = \{1w \mid w \in L, \}$  is regular, given  $L$  is regular. I remember seeing a similar question on the SP 22-Exam. I passed my proof request, along with the sample-exam problem and solution to two different LLMs. Letting  $M = (Q, \Sigma, \delta, s, A)$ , the LLMs produced the following language transformations:

GPT-5 Response: <https://chatgpt.com/share/68d16412-7184-800f-8569-40e03b8228d5>

$$\begin{aligned} Q' &= Q \times \{\text{pre}, \text{post}\} \\ s' &= (s, \text{pre}) \\ A' &= \{(q, \text{post}) \mid q \in A\} \\ \delta'((q, \text{post}), a) &= \{(\delta(q, a), \text{post})\}, \quad a \in \Sigma \\ \delta'((q, \text{pre}), \varepsilon) &= \{(\delta(q, 1), \text{post})\} \end{aligned}$$

Grok Response: <https://grok.com/c/d22ef5b8-4235-4743-9f28-69071d312a7a>

$$\begin{aligned} Q' &= Q \cup \{s'\} \\ s' &= s' \\ A' &= A \\ \delta'(s', 1) &= \{s\} \\ \delta'(q, a) &= \{\delta(q, a)\}, \quad q \in Q, a \in \Sigma \end{aligned}$$

**Spoiler alert: Both these solutions are correct!**

- Suppose we need to simulate the NFA resulting from one of the above transformations in a memory constrained environment. Which LLM's NFA is more memory-efficient? Succinctly (1-2 sentences) justify your answer.

- b. We decide to run incremental subset construction on each of the above NFAs. Which one leads to a smaller DFA?
- c. We now want to build an automaton for  $\overline{L'}$ . Suppose we take Grok's response and swap accepting and rejecting states. Would this work?
- d. Reread the first sentence of the problem. The goal was: "*Prove that  $L' = \{1w \mid w \in L\}$  is regular, assuming  $L$  is.*" Prove this in  $\leq 3$  lines using regular closure properties.

## 4 Language Transformation - 15 points

A run in a string  $w$  is a maximal non-empty substring of  $w$  in which all symbols are equal. For example, the string 011111100010000 consists of five runs(0, 111111, 000, 1, 0000).

For any string  $w \in \{0, 1\}^*$ , let  $compact(w)$  denote the string obtained by replacing each run with a single symbol from that run. For example,  $compact(\epsilon) = \epsilon$  and  $compact(1101111001) = 10101$

Let  $L$  be an arbitrary regular language.

Prove that the language  $compact(L) = \{compact(w) | w \in L\}$  is regular. <sup>3</sup>

---

<sup>3</sup>Hint: Consider the string  $w$  described above. We can break  $w$  down into runs of 0's and 1's as described in the problem. So we to construct a new machine for  $compact(L)$  that accepts the first 0, then  $\epsilon$ 's in the places of any subsequent 0's, then accepts a 1, then  $\epsilon$ 's instead of the following 1's, then a 0, then  $\epsilon$ 's instead of the following 0's, etc.

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

Let  $\Sigma = \{0, 1\}$ , and define the following languages:

$$L_x = \{w \in \Sigma^* \mid \text{every } 0 \text{ is immediately followed by at least two } 1\text{'s}\}$$

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

Let  $L_5 = L_x \cap L_y$ .

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

regular      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

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

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

$$L_6 = \{0^n 1^n w \mid w \in \Sigma^* \text{ and } n \geq 1\}^4$$

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

regular      not regular

- 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

---

<sup>4</sup>Yes this is very *similar* to a lab problem, but it is different. Regurgitating a memorized answer will get you in trouble. Take a moment and *think*.



## 7 Language classification III (3 parts) - 15 points

During the language classification portion of the exam, when Sumedh wasn't looking, a student took out his cellphone asked GPT-5 to help classify the following language as regular, irregular, context-free, or not-context-free.

$$L_7 = \{w \in \{0, 1\}^* \mid |w| > 3 \text{ and every run of 1s in } w \text{ has odd length}\}$$

- a. After looking at the paper to the left of him, the student guesses that the language is regular, so he tries to come up with a regex to prove it! He asks Chat-GPT the following.

Regex Prompt

Write a regular expression for strings where every run of 1's is odd and is  $> 3$ .

GPT-5 Regex: <https://chatgpt.com/c/68cdda9d-d5eo-832f-88df-c9d12778a754>

$0^*((111(11)^+)0^*)$

**Is this regular expression correct or incorrect.** If correct, succinctly explain why. Otherwise, provide a single counterexample.

- b. The student suddenly gets a suspicion that the language might be incorrect and asks GPT-4o for a fooling set proof for irregularity.

Regex Prompt

Consider the set of binary strings where every run of 1's is odd, but is also strictly greater than 3. Prove that this language is irregular via fooling sets. Give me the prefix and suffixes necessary.

Fooling Set Proof: <https://chatgpt.com/c/68ce1216-95d4-8327-ab8c-19c3f1313488>

$F = \{x_n = 1^{2n+1} \mid n = 0, 1, 2, 3, \dots\}.$

Consider  $x_i = 1^{2i+1}$ ,  $x_j = 1^{2j+1}$ , where  $i \neq j$ .

Use the suffix  $y = 0$ . If  $i \geq 2$ , then  $x_i y \in L$

If  $j < 2$ , then  $x_j y \notin L$

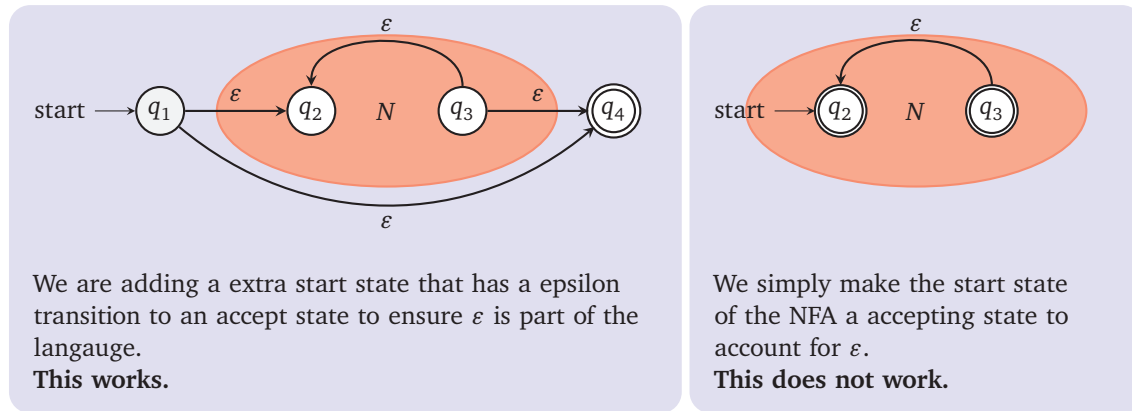
Thus, we have constructed an infinite fooling for L. L is not regular.

Is the following proof correct or incorrect? If incorrect, identify the error.

- c. After all this, really there is only one question left: Is  $L_7$  regular or not? Either way, prove it!

## EXTRA CREDIT - 5 POINTS

In our lecture (and cheatsheet) we constructed a transformation for  $L^*$  from the NFA of  $L$  to show that regular languages are closed under Kleene star. But in this transformation I specified that a new start state was required. Can you show me why?



In other words, **show me an example** of a NFA (which represents a language  $L$ ) where the left transform represents  $L^*$  but the right transformation does not represent  $L^*$ .

*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  $w = xvy$ .

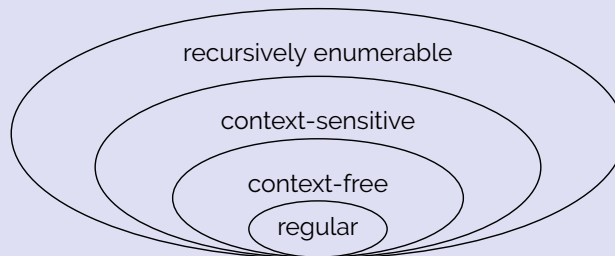
- If  $x = \epsilon$  then  $v$  is a *prefix* of  $w$
- If  $y = \epsilon$  then  $v$  is a *suffix* of  $w$

- 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.

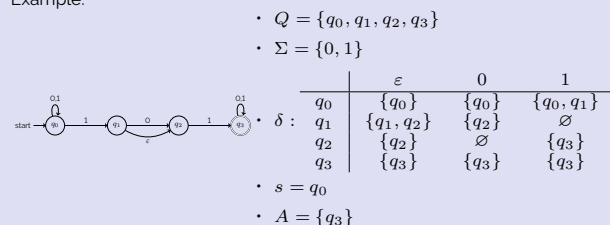
An NFA  $N$  *accepts a string*  $w$  iff some accepting state is reached by  $N$  from the start state on input  $w$ .

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

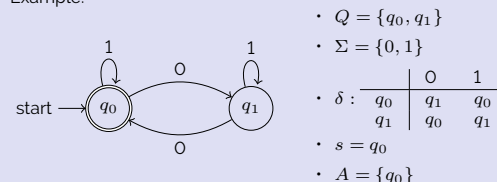
DFAs are finite state machines that can be represented as a directed graph or in terms of a tuple.

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

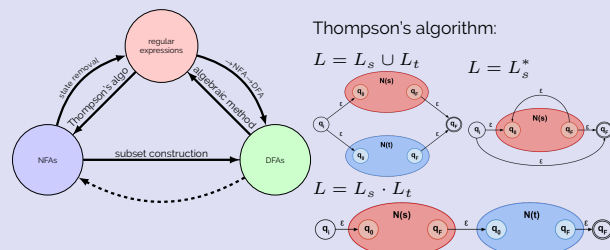
$L(M_C) = \{w \text{ has even number of 0s and 1s}\}$

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

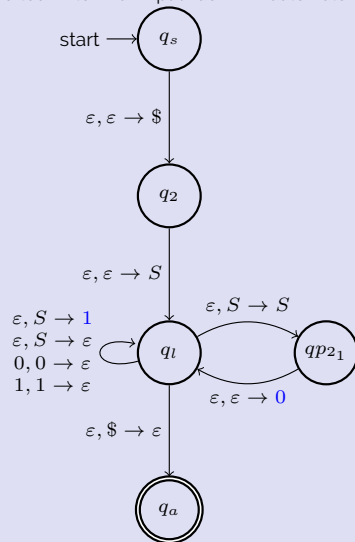
- $Q$  is a finite set whose elements are called *states*
- $\Sigma$  is a finite set called the *input alphabet*
- $\Gamma$  is a finite set called the *stack alphabet*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$  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.

The PDA to the right recognizes the language described by the following grammar:

$$S \rightarrow 0S1 \mid \varepsilon$$



### Context-free closure

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

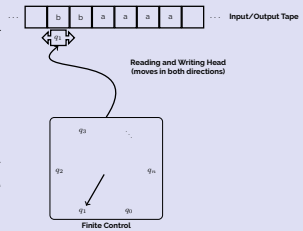
They are **not** closed under intersection or complement.

## 5 Recursively enumerable languages

### Turing Machines

Turing machine is the simplest model of computation.

- Input written on (infinite) one sided tape.
- Special blank characters.
- Finite state control (similar to DFA).
- Every step: Read character under head, write character out, move the head right or left (or stay).
- Every TM  $M$  can be encoded as a string  $\langle M \rangle$



Transition Function:  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{\leftarrow, \rightarrow, \square\}$

$\delta(q, c) = (p, d, \leftarrow)$

- $q$ : current state.
- $c$ : character under tape head.
- $p$ : new state.
- $d$ : character to write under tape head
- $\leftarrow$ : Move tape head left.

