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: ______________________________

1 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 ($\Sigma = \{\mathtt{0}, \mathtt{1}\}$):

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

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

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

■

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

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

Solution: $\mathtt{1}^*\mathtt{0}^*\mathtt{1}^*$ ■

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

Solution: $\varepsilon + \mathtt{0} + (\mathtt{0}+\mathtt{1})(\mathtt{0}+\mathtt{1})^+$ ■

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

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 $374 + 473 + 2 = 849$. ■

[1] ε, "$\mathtt{0}$", and "$\mathtt{1}$" are a part of this language.

2 Short Answer (Context-free) - 16 points

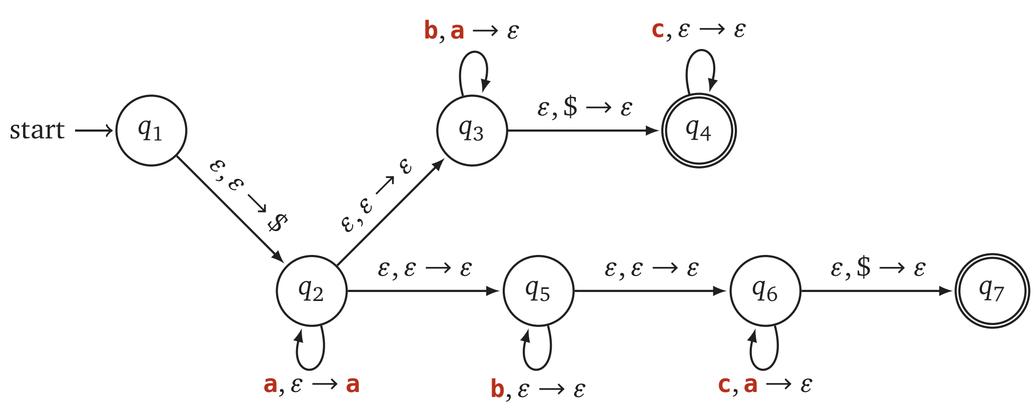
Unless the question asks for it, no explanation is required for your answers for full credit. Keep any explanations of your answers to 2 sentences maximum.

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

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

Solution: $V = \{S, A, B\}$, $T = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$, $P = \{S \rightarrow \mathbf{a}S\mathbf{c} | A, A \rightarrow \mathbf{b}A\mathbf{c} | B, B \rightarrow \mathbf{c}B | \varepsilon\}$, $S \rightarrow S$ ■

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



Solution: $L = \left\{ \mathbf{a}^i \mathbf{b}^j \mathbf{c}^k | i = j \text{ or } i = k \right\}$ ■

3 Language Transformation - 15 points

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

Prove that the language $delete2\mathtt{1}'s(L) := \{xyz \mid x\mathtt{1}y\mathtt{1}z \in L\}$ 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 ε-transitions that accepts $delete2\mathtt{1}'s(L)$ as follows. Intuitively, M' simulates M, but inserts two $\mathtt{1}$'s into M's input string at a non-deterministically chosen location.

- The state $(q, 0)$ means (the simulation of) M is in state q and M' has not yet inserted a $\mathtt{1}$.
- The state $(q, 1)$ means (the simulation of) M is in state q and M' has already inserted one $\mathtt{1}$.
- The state $(q, 2)$ means (the simulation of) M is in state q and M' has already inserted two $\mathtt{1}$'s.

$$\begin{aligned} Q' &:= Q \times \{0, 1, 2\} \\ s' &:= (s, 0) \\ A' &:= \big\{(q, 2) \bigm| q \in A\big\} \end{aligned}$$

$$\begin{aligned} \delta'((q,0),\varepsilon) &= \big\{(\delta(q,\mathtt{1}),1)\big\} \\ \delta'((q,1),\varepsilon) &= \big\{(\delta(q,\mathtt{1}),2)\big\} \\ \delta'((q,2),\varepsilon) &= \varnothing \end{aligned}$$

$$\begin{aligned} \delta'((q,0),a) &= \big\{(\delta(q,a),0)\big\} \\ \delta'((q,1),a) &= \big\{(\delta(q,a),1)\big\} \\ \delta'((q,2),a) &= \big\{(\delta(q,a),2)\big\} \end{aligned}$$

■

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.

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

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.

> **Solution:** regular   not regular
>
> 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:** context-free   not context-free
>
> 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 $\Sigma_5 = \{\mathtt{0}, \mathtt{1}\}$ and

$$L_5 = \left\{x\mathtt{0}y \middle| x, y \in \Sigma_5^*, \#_{\mathtt{1}}(x) \geq \#_{\mathtt{1}}(y)\right\}^{23}$$

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

Solution: regular   not regular

The language L_5 is not regular. Let $F = \mathtt{1}^*\mathtt{0}$. Let $x = \mathtt{1}^i\mathtt{0}$ and $y = \mathtt{1}^j\mathtt{0}$ be two distinct strings from F. Without loss of generality, assume that $i > j$. Let $z = \mathtt{1}^i$. Then $xz = \mathtt{1}^i\mathtt{01}^i$ is in L_5, while $yz = \mathtt{1}^j\mathtt{01}^i$ 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.

Solution: context-free   not context-free

The language L_5 is context-free, since the following CFG represents L_5.

$$S \to \mathtt{1}S \mid \mathtt{0}S \mid \mathtt{1}S\mathtt{1} \mid S\mathtt{0} \mid \mathtt{0}$$

∎

[2] x has *at least* as many $\mathtt{1}$'s as y

[3] The $\#_{\mathtt{a}}(w)$ operator counters the number of times character $\mathtt{a}$ appears in string w

6 Language classification III (2 parts) - 15 points

Let $\Sigma_6 = \{\mathtt{0}, \mathtt{1}\}$ 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.

Solution: regular not regular

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

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

■

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

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

Solution: context-free not context-free

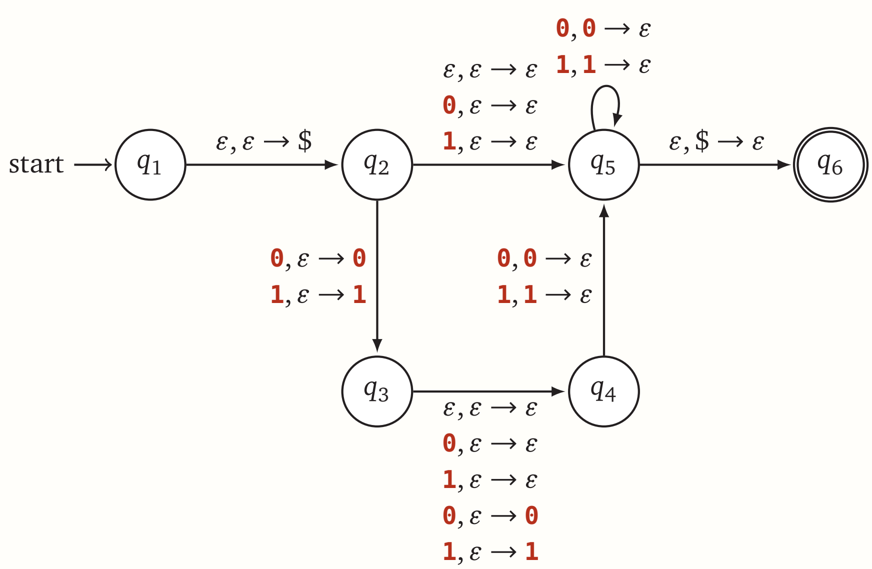
I'm adding in an extra solution because there appears to be a impression that if you didn't get the first part of the question right, you wouldn't get the second part right either. This is totally untrue. One o the things I like about these classification problems is that they can be answered in multiple different ways for full credit.

So while you could have answered this part by saying "I'm confident in my regular proof and this language is context-free because it is regular" you could have also proved context-free-ness in multiple different ways too.

You could have provided a context-free-grammar:

$$
\begin{aligned}
S &\to A|B\\
A &\to \mathtt{0}B\mathtt{0}|\mathtt{1}B\mathtt{1}\\
B &\to \varepsilon|\mathtt{0}|\mathtt{1}|\mathtt{00}|\mathtt{11}
\end{aligned}
$$

or you could have provided a push-down-automata:



The point is that there are numerous ways to answer these questions. The solutions in the solution packet are just one way. Feel free to for to OHs or post on Piazza is you want to discuss other potential solutions to any problem. ■

This page is for additional scratch work!

ECE 374 B Language Theory: Cheatsheet

1 Languages and strings

Languages

Definitions

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

$$L_{MULT2} = \left\{ \begin{matrix} 1\times 1|1, & 1\times 2|2, & 1\times 3|3,\dots \\ 2\times 1|2, & 2\times 2|4, & 2\times 3|6,\dots \\ \vdots & \vdots & \vdots \\ n\times 1|n, & n\times 2|2n, & n\times 3|3n,\dots \end{matrix} \right\} \tag{1}$$

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

Definitions

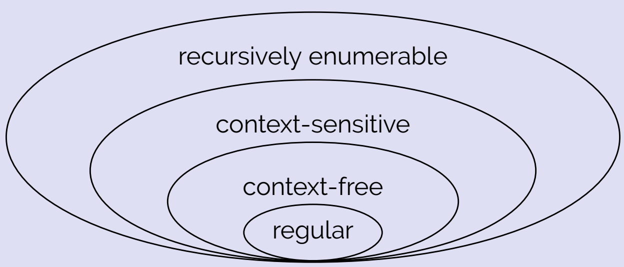
- The *length* of a string w (denoted by $|w|$) is the number of symbols in w.
- For integer $n \geq 0$, Σ^n is set of all strings over Σ of length n. Σ^* is the set of all strings over Σ.
- Σ^* is the set of all strings of all lengths including empty string.
- ε is a *string* containing no symbols.
- $\varnothing$ is the *empty set*. It contains no strings.

String operations

- If x and y are strings then xy denotes their concatenation. Recursively:
  - $xy = y$ if $x = \varepsilon$
  - $xy = a(wy)$ if $x = aw$
- v is *substring* of w $\iff$ there exist strings x, y such that $w = xvy$.
  - If $x = \varepsilon$ then v is a *prefix* of w
  - If $y = \varepsilon$ then v is a *suffix* of w
- A *subsequence* of a string $w = w_1 w_2 \dots w_n$ 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: $w^n = \varepsilon$ if $n = 0$ or $w^n = ww^{n-1}$ if $n > 0$

2 Overview of language complexity

Overview



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

[a]

[a] Table 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* $\mathbf{r}$ over an alphabet Σ is one of the following:
Base cases:

- $\varnothing$ the language $\varnothing$
- ε denotes the language $\{\varepsilon\}$
- a denote the language $\{a\}$

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

- $\mathbf{r_1} + \mathbf{r_2}$ denotes the language $L_1 \cup L_2$
- $\mathbf{r_1} \cdot \mathbf{r_2}$ denotes the language $L_1 L_2$
- $\mathbf{r_1^*}$ 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.

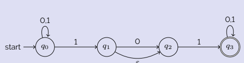
An NFA N *accepts a string* w iff some accepting state is reached by N from the start state on input w.

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*
- $\delta : Q \times \Sigma \cup \{\varepsilon\} \to \mathcal{P}(Q)$ is the *transition function* (here $\mathcal{P}(Q)$ is the power set of Q)
- s and Σ are the same as in DFAs

Example:



- $Q = \{q_0, q_1, q_2, q_3\}$
- $\Sigma = \{0, 1\}$
- δ :

	ε	0	1
q_0	$\{q_0\}$	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\{q_1, q_2\}$	$\{q_2\}$	$\varnothing$
q_2	$\{q_2\}$	$\varnothing$	$\{q_3\}$
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 $\delta^* : Q \times \Sigma^* \to \mathcal{P}(Q)$:

- 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

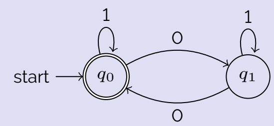
DFAs are finite state machines that can be represented as a directed graph or in terms of a tuple.

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

Example:



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

	0	1
q_0	q_1	q_0
q_1	q_0	q_1

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

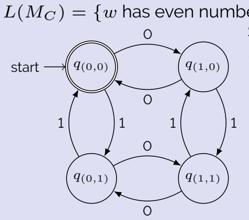
Every string has a unique walk along a DFA. We define the extended transition function as $\delta^* : Q \times \Sigma^* \to Q$ defined inductively as follows:

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

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

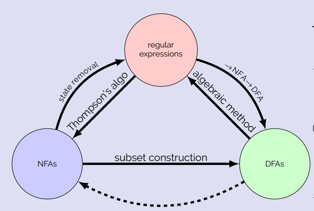


Suppose $M_0 = (Q_0, \Sigma, s_0, A_0, \delta_0)$ and $M_1 = (Q_1, \Sigma, s_1, A_1, \delta_1)$. Then

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

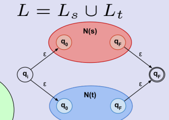
Regular language equivalences

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

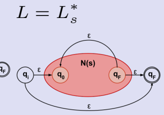


Thompson's algorithm:

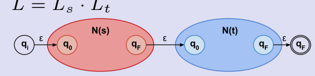
$L = L_s \cup L_t$



$L = L_s^*$



$L = L_s \cdot L_t$



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

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

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

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

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

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

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

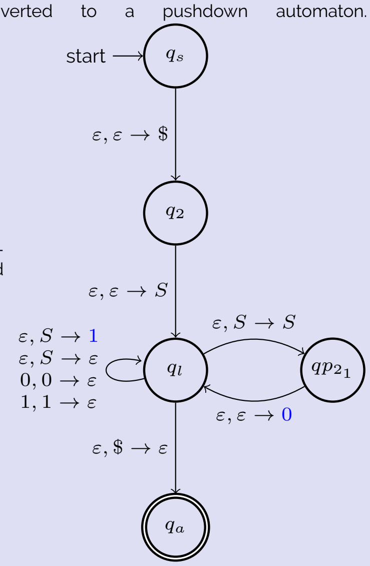
- Q is a finite set whose elements are called *states*
- Σ is a finite set called the *input alphabet*
- Γ is a finite set called the *stack alphabet*
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times (\Gamma \cup \{\varepsilon\}) \to \mathcal{P}(Q \times (\Gamma \cup \{\varepsilon\}))$ is the *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 $\langle$input read$\rangle$, $\langle$stack pop$\rangle \to \langle$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:

$S \to 0S|1|\varepsilon$



Context-free closure

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

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