COMPUTATIONAL MODELS
~ EXERCISES SOLUTION ~

TAU SPRING 2022

Author

Saar Barak

LaTeX Documents Available on Git-Hub

Contents

1 ∼ Assignment 1 ∼

1.1 Exercise 1

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and a boolean circuit C of size $|C| = s$ that computes f, Lets proof that there existences of a boolean circuit C' that's stand with the question property.

claim 1.1. *Any simple logic statement that uses $\vee$ or $\wedge$ before $\neg$ can be re-form to $\neg$ before $\vee$ or $\wedge$ using one additional gate*

Proof. using De-Morgan law for $x, y \in \{0,1\} : \neg(x \vee y) = \neg x \wedge \neg y$ and on the same way for $\neg(x \wedge y) = \neg x \vee \neg y$, we apply the same statement using extra one gate in total. □

using the claim above we can intuitive say that we need to "roll down the tree" all the NOT gates, and from quick indication I can claim the following.

claim 1.2. *if P have some cycle C size K that compute it $\Rightarrow$ three is some cycle $|C'| = |2K-1|$ that can computes P where all the $\neg$ gate's before any $\wedge$ or $\vee$ gates*

Proof. the indication base case is given from claim0.1, so for some Atomic P where $P_i \in \{0,1\} \Rightarrow$ in total $|K|$ simple gates we get:

$$\neg(P_1 \vee P_2 \vee \dots P_n) \Rightarrow \neg P_1 \wedge \neg P_2 \wedge \dots \neg P_n$$

we can notice that now we need $|K|$ NOT gates and$|K-1|$ of $\wedge$ gates, and in total $|K+K-1|$ gates. and all the $\neg$ gate's before any $\wedge$ gate's.
Now lets P_k be any boolean function that stand with the induction property,for some $x, y \in \{0,1\}$ lets look at:

$$X = \neg(P_k \vee x) = \neg P_K \wedge \neg x, Y = \neg(P_k \wedge y) = \neg P_K \vee \neg y$$

so in any case X, Y can be written in $|2k-1|$ gates of P_k, additional NOT gate and one $\wedge$ or $\vee$ gate, in total $|2k-1+3| = |2k+2|$ under the induction assumption we know that P have some $|C| = K$ that compute it ,so C including the following $\neg, \vee$ witch is size $|K+3|$ and can compute X □

on the same way we proof claim 0.2 we can get the same property for the $\wedge$ gate, so in total for some f we can find for any appearance of $\neg$ gate $\in C$, and each time apply De-Morgan law. based on claim 2 this new $|C'|$ hold the property (a)(b)(c).

1.2 Exercise 2

For f that has a circuit C of size m over B_ℓ, C can be written as a binary tree with m internal nodes. Each internal node is labelled by a gate, and each leaf is labelled by a variable, for each $v \in C$ lets $L_v : \{0,1\}^k \to \{0,1\}$be the circut that accept language , according to Lupanov '58 theorem its can be expressed as language over DeMorgan basis ,using the result of exercise 2 at Recitation 1 ,we know that for such L_v language there is some C that can accept it when $|C| = O(2^k)$ hence for $\forall v \in C \Rightarrow M|L_v|$ can be decided with some C_m size $O(m2^\ell)$.

1.3 Exercise 3

$\Longrightarrow$ Lets M be an DFA such as $M = (Q, \sum, \delta, q_0, F)$ and assume that M accept some $w \in \sum^*$ i.e $\hat{\delta}_m(q_0, w) \in F$. using $\hat{\delta}_m$ definition ,lets k_i be the state such as

$$\hat{\delta}_m(q_0, w) = k_n$$

$$\hat{\delta}_m(q_0, w) = \delta(\hat{\delta}_m(q_0, w_1, w_2 \dots w_n - 1), w_n) \Rightarrow k_{n-1} = \hat{\delta}_m(q_0, w_1, w_2 \dots w_n - 1)$$

follow same proses for $n-2, n-3, \dots$ so for general k_i on the "back-word" path

$$\hat{\delta}_m(q_0, w_j) = \delta(\hat{\delta}_m(q_0, w_1, w_2 \dots w_{j-1}), w_j) \Rightarrow k_{j-1} = \hat{\delta}_m(q_0, w_1, w_2 \dots w_{j-1})$$

hence $\forall k_i \in Q$, $k_n \in F$, $k_0 = q_0$ and $\delta_m(k_{i-1}, w_i) = k_i$ for $1 \leq i \leq n$
so we can claim that the sequence $k_1, k_2, \dots k_n$ stand with the property of the second equivalent DFA definition.

$\Longleftarrow$ Lets M be an DFA such as $M = (Q, \sum, \delta, q_0, F)$ and we assume that
$\forall q_i \in Q$, $q_n \in F$,$w_0 = q_0$ and $\delta_m(q_{i-1}, w_i) = q_i$ for some $w = w_1, w_2, \dots w_n$ and $q_1, q_2, \dots q_n$ with the same way decribed above

$$\delta(q_0, w_1) = q_1, \hat{\delta_m}(q_0, w_1, w_2) = \delta(\delta(q_0, w_1), w_2) = \delta(q_1, w_2) = q_2,$$

in general

$$\hat{\delta_m}(q_0, w_1, \dots, w_j) = \delta(\hat{\delta_m}(q_0, w_1, \dots, w_{j-1}), w_j) = q_j$$

$$\hat{\delta_m}(q_0, w_1, \dots, w_n) = \delta(\hat{\delta_m}(q_0, w_1, \dots, w_{n-1}), w_n) = q_n \in F$$

and we proof that both definition are equivalent

1.4 Exercise 4

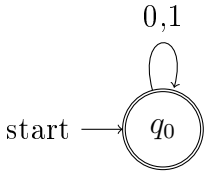


Figure 1: DFA (a) Σ^*

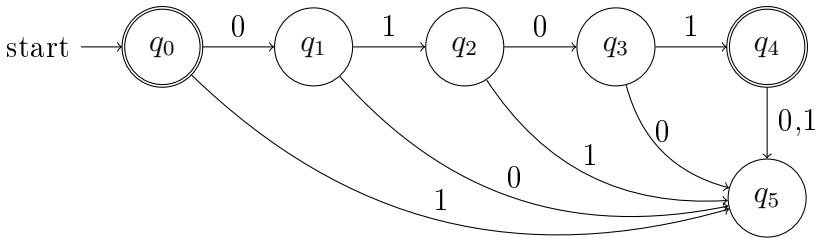


Figure 2: DFA (b) $\{\epsilon, 0101\}$

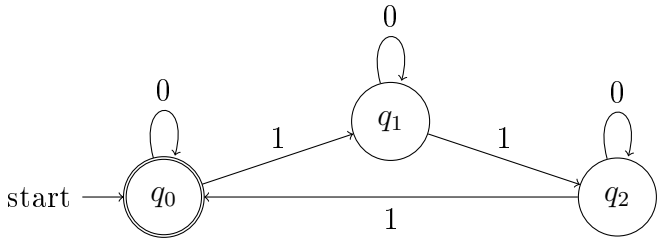


Figure 3: DFA (c) $\{w|\#_1 w \bmod 3 \equiv 0\}$

1.5 Exercise 5

Given n, Lets formalize the language

$$L_n = \{w0w_n | w \in \{0,1\}^* \wedge w_n \in \{0,1\}^{n-1}\}$$

lets present NFA $N = (Q, \Sigma, \delta, S, F)$ that accept L_n , we know that any NFA can be transform to DFA .

$$Q = \{q_0, q_1 \ldots, q_n\}, \Sigma = \{0,1\}, S = \{q_0\}, F = \{q_n\}$$

$$\delta(q_i, \sigma) = \begin{cases} \{q_0, q_1\} & \text{if } q_i = q_0 \wedge \sigma = 0 \\ \{q_0\} & \text{if } q_i = q_0 \wedge \sigma = 1 \\ \{q_{i+1}\} & \text{if } 0 < q_i < q_n \forall \sigma \in \Sigma \\ \phi & \text{if } q_i = q_n \forall \sigma \in \Sigma \end{cases}$$

claim 1.3. *N accept* L_n

Proof. $w \in L_n \Leftrightarrow$

$$w \in \{w0w_n | w \in \{0,1\}^* \wedge w_n \in \{0,1\}^{n-1}\} \Leftrightarrow \exists \hat{w} \in \Sigma^*. \exists \sigma \in \{0\}. w_n \in \{0,1\}^{n-1}$$

$$\Leftrightarrow \exists \hat{w} \in \Sigma^*. \exists \sigma \in \{0\}. \hat{\delta}_w(q_1, w_n) \in Q \Leftrightarrow \exists \hat{w} \in \Sigma^*. \exists \exists \delta(q_0, q_1), \hat{\delta}_w(q_1, w_n)) \in Q$$

$$\Leftrightarrow \exists \hat{w} \in \Sigma^*. \exists \hat{\delta}(q_0, q_n) \in Q \Leftrightarrow \exists \exists \ldots q_i, q_j \in Q. \exists \hat{\delta}(q_0, q_n) \in F$$

$$\Leftrightarrow \delta(\hat{\delta}_w(q', q_n), \sigma) \exists q' \in Q \forall \sigma \in \Sigma. \hat{\delta}(q_0, q_n) \in F \Leftrightarrow \phi.(q_0, q_n) \in F \Leftrightarrow (q_0, q_n) \in L(N)$$

to be honest i'm not relay sure about the last few equivalence :) □

Now lets show N as DFA

$$Q_m = \{[R]|R \in Q\}, \Sigma = \{0,1\}, \delta_0 = q_0, F_m = \{[R]|R \cap Q \neq \phi\}, \delta_m([R],\sigma) = \cup_{q\in R}\delta(q,\sigma)$$

Question 5 (b)

claim 1.4. *All regular languages is closed under $\setminus$ with a any regular language.*

Proof. Lets L_1 and L_2 be regular, using the regular opertor we know already, we can immediately claim that following holds

$$L_1 \cap \overline{L_2} \Rightarrow L_1 \setminus L_2 \text{ is regular language}$$

□

1.6 Exercise 6

Let $A = (Q, \sum, \delta, q_0, F)$ be a DFA, and let $w_1, w_2 \in \Sigma^*$ be words such that $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2)$. Lets proof that for every word $w \in \Sigma^*$ it holds that $w_1 w \in L(A) \Leftrightarrow w_2 w \in L(A)$ now we set some q_k such as
$\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2) = q_k \in Q$. now we construct new A_1 and A_2 in the following way:

$$A_1 = (Q, \Sigma, \delta, q_0, q_k = F_1), A_2 = (Q, \Sigma, \delta, q_k, F_2 = F)$$

we can notice that $\hat{\delta}(q_0, w_1) = \hat{\delta}(q_0, w_2) \in L(A_1)$,and for some w such as $w \in L(A_2)$ we get

$$w \in L(A_2) \Leftrightarrow \hat{\delta}(q_k, w) \in F_2 \Leftrightarrow \hat{\delta}(q_k, w) \in F$$

$$\Leftrightarrow q_k \in F_1, \hat{\delta}(q_k, w) \in F$$

$$\Leftrightarrow \forall w' \in L(A_1).\hat{\delta}(\hat{\delta}(q_0, w')), w) \in F$$

$$\hat{\delta}(\hat{\delta}(q_0, w_1)), w) \in F \Leftrightarrow \hat{\delta}(\hat{\delta}(q_0, w_2)), w) \in F$$

$$w_1 w \in L(A) \Leftrightarrow w_2 w \in L(A)$$

Question 6 (b)
Let n be a number and let A be a DFA such that $L(A) = \{0^i 1^i : 0 \leq i \leq n\}$. now follow the hint

claim 1.5. $\hat{\delta}(q_0, 0^i) \neq \hat{\delta}(q_0, 0^i)$ *for any* $i \neq j$

Proof. lets assume that $\hat{q} = \hat{\delta}(q_0, 0^i) = \hat{\delta}(q_0, 0^j)$ and without any loss of generality $i < j$ .from A definition we know that $w_i := 0^i 1^i, w_i := 0^j 1^j \in L(A)$ i.e for $\hat{\delta}(\hat{\delta}(q_0, 0^j), 1^i)$ we can get to for some $q \in F$. now lets look at

$$\hat{\delta}(q_0, w) \in F \Rightarrow \ \hat{\delta}(\hat{\delta}(q_0, 0^i), 1^j) = \hat{\delta}(\hat{q}, 1^i)$$

under the assuming we lets look at

$$\hat{\delta}(\hat{q}, 1^i) = \hat{\delta}(\hat{\delta}(q_0, 0^j), 1^i) = \hat{\delta}(q_0, 0^j 1^i) \in F$$

and we got an contradiction.

□

Hence the claim above hold for any $i \neq j$ and in total we find that A has at least n accepting state.

2 ∼ Assignment 2 ∼

2.1 Exercise 1

Define the language L_n over alphabet $\sum_n = \{1, 2, ..., n\}$ to be the set of words which do not contain all the letters from $\sum_n$, now lets describe DFA $M = (Q, \sum, \delta, q_0, F)$ s.t M accept L_n.

$$Q = \{q_k | k \subseteq \{\Sigma_n\}\}, \Sigma = \Sigma_n, q_0 = q_\phi, F = \{Q/\{1, 2, 3 \dots n\}\}$$

$$\delta(q_k, \sigma) = \begin{cases} q_{k \cup \sigma} & \text{if } \sigma \notin \{k\} \\ q_k & \text{if } \sigma \in \{k\} \end{cases}$$

its sufficient to see that the following claim hold to proof M accept L_n

claim 2.1. *the DFA M **not** accept w iff* $\{\sigma : \sigma \in w\} = \{k_n\}$ *when* k_n *is the set of all the letters from* Σ_n

Proof. First lets notice that Q contain all the subset from Σ_n including the empty set witch correspond the empty world i.e $\{\sigma\}$ so in total we looking at $O(2^n)$ state for finite alphabet size n.
$\Rightarrow w \notin L_m$ if $w \in \Sigma_n^*$ since M have only one state lets call it q_n which is not accepting state, hence $\hat{\delta}(q_0, w) = q_n$ we must go through at least n state to achieve q_n.
now using reduction on the number of district steps we done for $\hat{w} = \phi$ $\delta(q_0, \phi) = q_0$ and for some $\hat{w}, \hat{\sigma}$ s.t

$$\hat{\sigma} \notin |\{\sigma : \sigma \in \hat{w}\}| = |k|, \hat{\delta}(q_0, \hat{w}) = q_k \Rightarrow \delta(q_k, \hat{\sigma}) = q_{k \cup \hat{\sigma}} = \hat{\delta}(q_0, (\hat{w}||\hat{\sigma})), \{\sigma : \sigma \in \hat{w}||\hat{\sigma}\}| = |k{+}1|$$

hence we done n district steps so $|\{\sigma : \sigma \in w\}| = n$ since M is DAG (except the self loops) w contain all the letters from the alphabet
$\Leftarrow w := \forall \sigma \in \Sigma_n | \sigma \in \{w\}$ now lets look at the first $\sigma \in w$, so $\delta(q_0, \sigma) = q_\sigma \in Q$ if the next letter in w hold $\hat{\sigma} \neq \sigma, \delta(q_\sigma, \hat{\sigma}) = q_\sigma$ but we know that w have n district letters so in total we get

$$\hat{\delta}(q_0, w) = \hat{\delta}(q_{\sigma_1}, \underbrace{\hat{w}}_{\{\hat{w}\}=\{w/\sigma_1\}}) = \hat{\delta}(q_{\{\sigma_1, \sigma_2 \dots \sigma_k\}}, \underbrace{\hat{w}}_{\{\hat{w}\}=\{w/\sigma_1, \sigma_2 \dots \sigma_k\}}) = \delta(q_{\{w/\sigma_n\}}, \sigma_n) = q_n \notin F$$

hence $w \notin L_m$.

□

2.2 Exercise 2

When A,B regular languages over same alphabet. $A \diamond B = \{xy : x \in A \wedge y \in B \wedge |x| = |y|\}$ lets define

$$\Sigma = \{0, 1\}, A = L(0^*), B = L(1^*)$$

by assuming that $A \diamond B$ is regular language, using the property of regular language operation we get.

$$(A \diamond B) \cap \{0^*, 1^*\} = \{0^i, 1^i | i \geq 0\}$$

which lead to contradiction since $\{0^i, 1^i | i \geq 0\}$ is not regular language

2.3 Exercise 3

(a) $L_1 = \{w : \#a(w) \geq \#b(w)\}$ over $\Sigma = \{a, b, c\}$
L_1 is not regular while assuming it is. for fix ℓ lets $w = b^\ell a^\ell$,$w \in L_1$ based on Pumping Lemma we can notice $|w| > \ell$, and for any patrion of $w = xyz$ such that $|y| > 0, |xy| \leq \ell$ hence y is in the form of b^k now lets look at

$$b^{\ell+k}a^\ell \Rightarrow \#_b(b^{\ell+k}a^\ell) > \#_a(b^{\ell+k}a^\ell) \Rightarrow b^{\ell+k}a^\ell \notin L_1$$

we got an contradiction hence L_1 is not regular

(b) $L_2 = \{w : |w| \in \mathbb{N}_{even} \wedge w = w^R)\}$ over $\Sigma = \{0, 1\}$
L_2 is not regular while assuming it is. for fix ℓ lets $w = 1^\ell 001^\ell$,$w \in L_2$ based on Pumping Lemma we can notice $|w| > \ell$, and for any patrion of $w = xyz$ such that $|y| > 0, |xy| \leq \ell$ hence y is in the form of 1^k now lets look at

$$(1^{k+\ell}001^\ell) \neq (1^{k+\ell}001^\ell)^R \Rightarrow 1^{k+\ell}001^\ell \notin L_2$$

we got an contradiction hence L_2 is not regular

(c) $L_3 = \{w : |w| \in \mathbb{N}_{even} \wedge w = w^R)\}$ over $\Sigma = \{0\}$
L_3 is regular language since we can be written as regular expression

$$L_3 = L((00)^*) = \{00\}^* = \{\epsilon, 00, 0000, \dots\}$$

(d) $L_4 = \{w : |w| \in \mathbb{N} \text{ s.t } |w| = n^3\}$ over $\Sigma = \{0, 1\}$
L_4 is not regular while assuming it is. for fix ℓ lets $w = 0^{\ell^3}$,$w \in L_4$ based on Pumping Lemma we can notice $|xyz| > \ell$, and for any patrion of $w = xyz$ such that $|y| > 0, |xy| \leq \ell$. hence for some k,m such that $k + m \leq \ell$ this means that :

$$x = 0^k, y = 0^m, z = 0^{\ell^3 - k - m} \Rightarrow, xyz = 0^{\ell^3}$$

by our assumption for any $n \in \mathbb{N}$, $xy^n z = 0^{\ell^3 + m(n-1)} \in L_4$, lets choose n=2

$$\text{since } 1 \geq m \geq \ell \text{ we will get for some } t \Rightarrow \underline{t^3 = \ell^3 + m}$$

$$\text{but } t^3 \geq (\ell + 1)^3 = \ell^3 + 3\ell^2 + 3\ell + 1 > \ell^3 + \ell \geq \ell^3 + m \Rightarrow \underline{t^3 > \ell^3 + m}$$

we got an contradiction hence L_4 is not regular

2.4 Exercise 4

claim 2.2. *For language L s.t L satisfies pumping lemma with pumping constant ℓ , $L||L$ satisfies pumping lemma with pumping constant 2ℓ*

Proof. Lets $w_1, w_2 \in L'$ i.e $w_1 \in L \wedge w_2 \in L$ and lets assume $|w_1 w_2| \geq 2\ell$ to imply the pumping lemma, now we can notice that there is 2 possible scenario

- $|w_1| \geq \ell$ hence we can write w_1 as $w_1 = xyz \Rightarrow$ and we can write w_1w_2 as $w_1w_2 = xyz||w_2$ which stand with the lemma property
- $|w_1| < \ell$ so $|w_2| > \ell$ can write w_2 as $w_2 = xyz$ and $|xy| \leq \ell \Rightarrow |w_1xy| \leq 2\ell, |y| > 0$ we can write w_1w_2 as $\underbrace{w_1x}_{x'}yz$ which stand with the claim property

□

2.5 Exercise 5

(a)

Lets L be a regular language over alphabet Σ. and we will proof the language L' define $L' = \{xyz : xy^Rz \in L\}$ is regular. since L is regular $\Rightarrow$ exists some DFA that accept L $M = (Q, \Sigma, \delta, q_0, F)$ now lets define few new DFA's s.t:

- $M_{q_0,q_k} = (Q, \Sigma, \delta, q_0, q_k)$ for any $q_k \in Q$ this DFA will cover any path that accessible from q_0
- now lets look at $M_{q_k,q_j} = (Q, \sum, \delta, q_k, q_j)$lets the language of M_{q_k,q_j} is define by $L(M_{q_k,q_j}) = \{w : \exists q_k, q_j \in Q|$ exists path s.t $q_k \rightsquigarrow q_j\}$,since $L(M_{q_k,q_j})$ is regular $\Rightarrow$ $rev(L(M_{q_k,q_j}))$ is regular (Recitation 3 ex.1). for any $q_k, q_j \in Q$ lets define the following language $L(M_{q_k,q_j})^R$
- $M_{q_k,F} = (Q, \Sigma, \delta, q_k, F)$ for any $q_k \in Q$ this DFA will cover any path that end in F

claim 2.3. $L' = \cup_{q_i,q_j} L(M_{q_0,q_i})||L(M_{q_i,q_j})^R||L(M_{q_j,F})$

Proof.

$$w = xyz \in L' \Leftrightarrow xy^Rz \in L \Leftrightarrow \exists\hat{\delta}(\hat{\delta}(\hat{\delta}(q_0, x), y^R), z) \in F_m \Leftrightarrow$$

$$\exists M_{q_0,q_i} : \hat{\delta}(q_0, x) \in F_{M_{q_0,q_i}} \wedge \exists L(M_{q_i,q_j})^R : y \in L(M_{q_i,q_j})^R \wedge \exists M_{q_j,F} : \hat{\delta}(q_j, z) \in F_{M_{q_j,F}} = F_m$$

$$\Leftrightarrow xyz \in \cup_{q_i,q_j} L(M_{q_0,q_i})||L(M_{q_i,q_j})^R||L(M_{q_j,F})$$

□

(b)

Lets L be a regular language over alphabet Σ. and we will proof the language L' define $L' = \{xy \in \Sigma^* : (x \in L)$ XOR $(y \in L)\}$ is regular. using the closure property of regular language

- $L = \{x \in \Sigma^* : x \in L\}$ and $\overline{L} = \{x \in \Sigma^* : x \notin L\}$ are regular.
- $L||\overline{L} = \{xy \in \Sigma^* : x \in L \wedge y \notin L\}$ and $\overline{L}||L = \{xy \in \Sigma^* : x \notin L \wedge y \in L\}$ are regular.
- $(L||\overline{L}) \cup (\overline{L}||L) = \{xy \in \Sigma^* : (x \in L \wedge y \notin L) \vee (x \notin L \wedge y \in L) = (x \in L)$ XOR $(y \in L)\}$ is regular

Exercise 6

(a) $\{w \in \Sigma^* : \#_0(w) \leq 3\}$
lets express the following as regular expression

$$\underbrace{1^*}_{I} \cup \underbrace{1^*01^*}_{II} \cup \underbrace{1^*01^*01^*}_{III} \cup \underbrace{1^*01^*01^*01^*}_{IV}$$

(I) is the regular expression of the language that contain non zeros at all.
(II) is the regular expression of the language that contain exactly 1 zero.
(III) is the regular expression of the language that contain exactly 2 zero.
(IV) is the regular expression of the language that contain exactly 3 zero.
hence combine them all together will hold regular expression of the language that contain at most 3 zeros.

(b) $\{w \in \Sigma^* : |w| \bmod 4 = 2\}$
lets express the following as regular expression

$$\underbrace{(0 \cup 1)(0 \cup 1)}_{(I)} \underbrace{((0 \cup 1)(0 \cup 1)(0 \cup 1)(0 \cup 1))^*}_{II}$$

(I) is the regular expression of the language that size is exactly 2.
(II) is the regular expression of the language from size 0,4,8..
hence the concatenate of (I) and (II) will hold the $|w| \bmod 4 = 2$ property

3 $\sim$ Assignment 3 $\sim$

3.1 Exercise 1

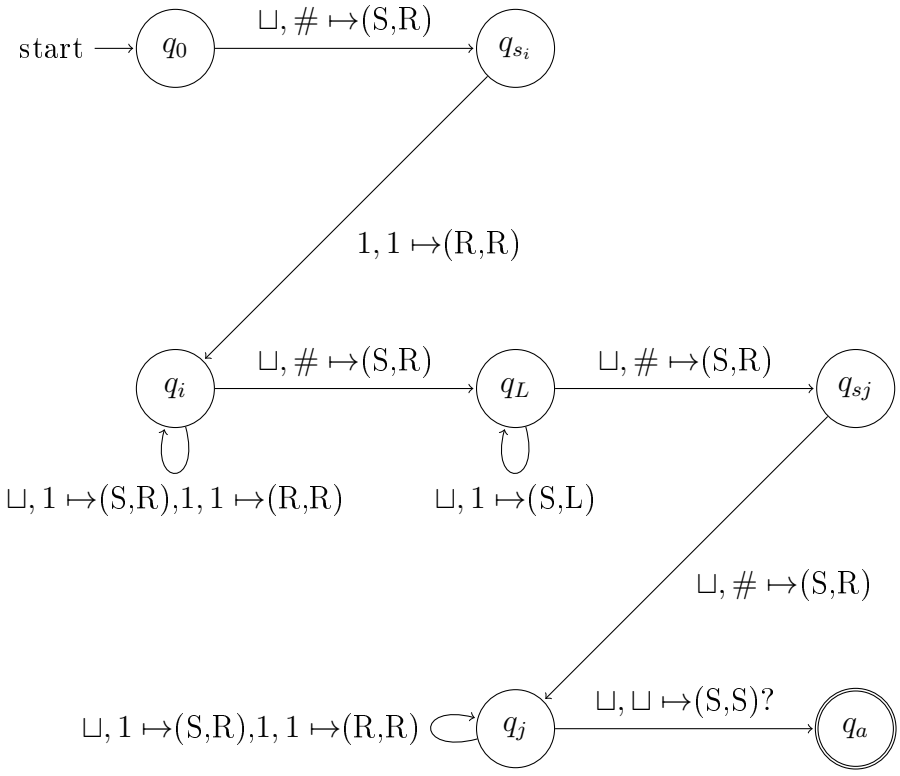
Considering the following language

$$\mathcal{L} = \{\#1^n\#x_1 \dots x^n : \exists i \neq j \text{ s.t } x_i \neq x_j\}$$

Lets describe an 2-tape NTM $\mathcal{M}$ that allow the head to stay at the same place, that recive input $\#1^n\#(n > 1)$ and implements the first part of decide $\mathcal{L}$

$$Q = \{q_0, q_a, q_r\}, \Sigma = \{1, \#\}, \Gamma = \{1, \#, \sqcup\}$$

and define δ :



The idea behind the way I generated δ is first to verify we start "legal" sequence i.e $i \neq j$ and $i, j > 0$ and split non-deterministic for any i, j the NTM run

3.2 Exercise 2

(a)

For DFA $A = (Q, \Sigma, \delta, q_0, F)$. lets describe a 2-tape TM M that accept $\mathcal{L}(A)$

$$Q = \{q_m, q_a, q_r\}, \Sigma = \Sigma, \Gamma = \Sigma \cup \{\{q \in Q\}, \sqcup\}, q_0 = q_m$$

$$\delta(q, (\sigma_1, \sigma_2)) = \begin{cases} (q_m, (\sigma_1, \delta(q_0, \sigma_1), (R, L)) & \text{if } \sigma_1 \in \Sigma \wedge \sigma_2 = \sqcup \\ (q_m, (\sigma_1, \delta(\sigma_1, \sigma_2)), (R, L)) & \text{if } \sigma_1 \in \Sigma \wedge \sigma_2 = q \in Q/F \\ (q_a, (\sigma_1, \sigma_2), (R, L)) & \text{if } \sigma_1 = \sqcup \wedge \sigma_2 = q \in F \\ (q_r, (\sigma_1, \sigma_2), (R, L)) & \text{if } \sigma_1 = \sqcup \wedge \sigma_2 = q \notin F \end{cases}$$

The idea is to save the last visted state on of A.

(b)

For TM $M = (Q, \Sigma, \delta, q_0, F)$. lets describe a 2-tape TM M_2 that accept $\mathcal{L}(M)$

$$Q = \{q_{m2}, q_a, q_r\}, \Sigma = \Sigma, \Gamma = \Sigma \cup \{q \in Q\}, q_0 = q_{m2}$$

Lets define $\hat{q}, \hat{\sigma}, \hat{D}$ to be the triplet such that $\delta(q, \sigma) \longmapsto (\hat{q}, \hat{\sigma}, \hat{D})$

$$\delta(q, (\sigma_1, \sigma_2)) = \begin{cases} (q_m, (\hat{\sigma}, \hat{q}), (\hat{D}, L)) & \text{if } \sigma_1 \in \Sigma \wedge \sigma_2 = \sqcup \\ (q_m, (\hat{\sigma}, \hat{q}), (\hat{D}, L)) & \text{if } \sigma_1 \in \Sigma \wedge \sigma_2 = q \in Q \\ (q_a, (\sigma_1, q_a), (R, L)) & \text{if } \sigma_2 = q_a \wedge \forall \sigma_1 \\ (q_r, (\sigma_1, q_r), (R, L)) & \text{if } \sigma_2 = q_r \wedge \forall \sigma_1 \end{cases}$$

The idea is kind of same as above but now we track the direction of the head of M

3.3 Exercise 3

Proposition 1. *Disprove $\mathcal{RE}$ is not closed under complement.*

For $L, \overline{L}$, if both are semi-decidable $\Rightarrow$ both are decidable, (can just flip the nation reject accept) $\Rightarrow$ if we assume $\mathcal{RE}$ closed under complement, $\forall L \in \mathcal{RE}, \overline{L} \in \mathcal{RE} \Rightarrow$ we get that $\mathcal{RE} = \mathcal{R} \Rightarrow \clubsuit$

Proposition 2. *co$\mathcal{RE}$ is closed under intersection.*

By detention $L \in co\mathcal{RE}$ if $\overline{L} \in \mathcal{RE}$. using the fact that $\mathcal{RE}$ Closures under union we can apply De-Morgan law for some $\overline{L_1}, \overline{L_2} \in co\mathcal{RE}$ which lead us to :

$$L_1, L_2 \in \mathcal{RE} \Rightarrow L_1 \cup L_2 \in \mathcal{RE} \Rightarrow \overline{L_1 \cup L_2} \in co\mathcal{RE} \Rightarrow \overline{L_1} \cap \overline{L_2} \in co\mathcal{RE}$$

Proposition 3. *$\mathcal{R}$ is closed under Kleene star.*

w.l.o.g $\mathcal{L} \in \mathcal{R}$ such that exist TM T that decide $\mathcal{L}$.now we can build an TM M^* that accept L^* for input x .

- if $x = \epsilon$ Accept.
- partition $|x| = n$ to any combination possible ways lets say $x_{power} := (\{x_1\}, \{x_2\} \dots \{x_{2^{n-1}}\})$
- run L for any $\forall w \in \{x_i\}$, if for some x_i L accept **all** $w \in \{x_i\}$ Accept.
- else Reject.

Proposition 4. *$\mathcal{RE}$ is closed under Prefix, but $\mathcal{R}$ is not closed under Prefix,*

i. Since $\mathcal{L} \in \mathcal{RE}$, there exists an enumerator $f_{\mathcal{L}}$. Its will be sufficient to describe f_{prefix} to proof that $\mathcal{L} \in \mathcal{RE} \Rightarrow \text{Prefix}(\mathcal{L}) \in \mathcal{RE}$ Now lets built new f_{prefix}.

$$f_{prefix} = w := \{\sigma_1 \dots \sigma_k\} \forall k : 0 < k < |w| \text{ for any } f_{\mathcal{L}} = w$$

the idea is to use $f_{\mathcal{L}}$ output and apply Prefix on any world to create f_{prefix}

ii. Lets assume $\mathcal{R}$ closed under Prefix. and lets look at my favourite TM M_F for $L_f \in \mathcal{RE}/\mathcal{R}$. now lets construct new $\hat{L}$ consisting of strings from the encode of M_F and # i.e $M_F\#$ now I claim that $\hat{L}$ Accept only when its see # ,otherwise its Reject. hence exist some deterministic TM that Reject/Accept. $\hat{L} \in \mathcal{R}$, but $\text{Prefix}(\hat{L}) = L_f \notin \mathcal{R} \Rightarrow \clubsuit$

3.4 Exercise 4

Define Size$(O(n))$:

$$\text{Size}(O(n)) = \{\mathcal{L} : \exists \mathcal{C} := \{C_n\}_{n\in N} \text{ s.t } \mathcal{L}(\mathcal{C}) = \mathcal{L} \wedge |C_n| \in O(n) \forall n \in N\}$$

i. Lets look at the unary language $\mathcal{L} \subseteq \{1^n : n \in N\}$, its immediate that $\mathcal{L} \in \text{Size}(O(n))$ since its needs to have a single "hardwired" bit for indicating 1^n. now lets look at the language $\mathcal{L}_{\mathcal{U}}$

$$\mathcal{L}_{\mathcal{U}} := \{1^n | \text{ The Turing machine encode to n halts on } \epsilon.\}$$

Hence its immediate from Rice's Theorem, or the fact that $H_{TM\epsilon} \leq_m \mathcal{L}_{\mathcal{U}}$ that $\mathcal{L}_{\mathcal{U}} \notin \mathcal{R}$ and $\mathcal{L}_{\mathcal{U}} \in \text{Size}(O(n))$

ii. We can look at some recursively define $\mathcal{L} \in \mathcal{R}$ for example:

$$\mathcal{L} = \{1^{2^n} : n \geq 0\}$$

Since we will need circuit for each possible input length, we may need exponential size circuit (in terms of n) to accept words in the language. and the following holds that $\mathcal{L} \notin \text{Size}(O(n))$

3.5 Exercise 5

(A) **Prove:** If $\mathcal{L}_1 \leq_m \mathcal{L}_2$ and $\mathcal{L}_2 \leq_m \mathcal{L}_3$, then $\mathcal{L}_1 \leq_m \mathcal{L}_3$.
Define $f_{1\to2}, f_{2\to3}$ to be a computable mapping redaction functions. now lets $w \in \mathcal{L}_1$, and by dentition we get :

$$w \in \mathcal{L}_1 \Leftrightarrow f_{1\to2}(w) \in \mathcal{L}_2 \Leftrightarrow f_{2\to3}(f_{1\to2}(w)) \in \mathcal{L}_3$$

(B) **Disprove:** If $\mathcal{L}_1 \leq_m \mathcal{L}_2$ and $\mathcal{L}_2 \leq_m \mathcal{L}_1$, then $\mathcal{L}_1 = \mathcal{L}_2$.
for H_{TM}, A_{TM} we know that $H_{TM} \leq_m A_{TM}$ and $A_{TM} \leq_m H_{TM}$ but $A_{TM} \neq H_{TM} \Rightarrow \clubsuit$

(C) **Disprove:** If $\mathcal{L}_1 \subseteq \mathcal{L}_2$ then $\mathcal{L}_1 \leq_m \mathcal{L}_2$.
Lets look at $\mathcal{L} = \Sigma^*$,$\mathcal{L} \in \mathcal{R}$ since any other $\mathcal{L}' \subseteq \mathcal{L}$. if we assume that $\forall \mathcal{L}' \leq_m \mathcal{L}$ its will lead to $\forall \mathcal{L}' \in \mathcal{RE} \Rightarrow \clubsuit$

(D) **Disprove:** For every $\mathcal{L}_1, \mathcal{L}_2$, then $\mathcal{L}_1 \leq_m \mathcal{L}_2$ or $\mathcal{L}_2 \leq_m \mathcal{L}_1$
Lets look at $A_{TM}, \overline{A_{TM}}$. If we assume $\overline{A_{TM}} \leq_m A_{TM}$ Since $\overline{A_{TM}} \notin \mathcal{RE}$ its will lead to $A_{TM} \in \mathcal{R} \Rightarrow \clubsuit$,
and if $A_{TM} \leq_m \overline{A_{TM}}$ Since $A_{TM} \notin$ co-$\mathcal{RE} \Rightarrow \overline{A_{TM}} \in \mathcal{R} \Rightarrow \clubsuit$

(E) **Prove:** If $\mathcal{L}$ is regular, then $\mathcal{L} \leq_m HALT$
Lets A be an DFA s.t $L(A) = \mathcal{L}$ when $\mathcal{L}$ is regular. based on 2(a) we define some TM M that accept $L(A)$ and Let M_{loop} be TM that loop forever for any input. hence we can define computable f :

$$f(w) = \begin{cases} \langle M, w\rangle & \text{if } w \in M \\ \langle M_{loop}, w\rangle & \text{if } w \notin M \end{cases}$$

Which imply $\mathcal{L} \leq_m HALT$

3.6 Exercise 6

(a) $\mathcal{L} = \{\langle M \rangle : M \text{ is a TM and } |L(M)| > 10\} \in \mathcal{RE}/\mathcal{R}$

Lets describe algorithm A_{10} that accept $\mathcal{L}$

Algorithm 1 A_{10} on input $\langle M \rangle$

Require: $\langle M \rangle$ is a valid encode of TM
else **Reject**
$k \leftarrow 1$
while $k < \infty$ **do**
counter $\leftarrow 0$
for $i \leftarrow 1$ **to** k **do**
simulate M on w_i for k steps ▷ w_i generated from Σ of M
if M accept **then**
counter + 1
end if
end for
if counter > 10 **then Accept**
end if
end while

Its implement that we cover any optional input on M and while discovered more then 10 worlds in $\mathcal{L}$ we accept. hence $\mathcal{L} \in \mathcal{RE}$
Now lets proof $\mathcal{L} \notin \mathcal{R}$ using reduction from HALT$\leq_m \mathcal{L}$. now lets define $\mathcal{M}$

Algorithm 2 $\mathcal{M}$ on input $\langle M, \hat{w} \rangle$.

Require: $\langle M, \hat{w} \rangle$ is a valid encode of TM and word
else **Reject**
Ignore $\hat{w}$
Write M and w on the tape
run M on w **Accept** if M halt
else **Reject**

I claim that exists maping function from H_{TM} to $\mathcal{L}$ s.t:

- If $\langle M, \hat{w} \rangle \in H_{TM} \Rightarrow M(\hat{w})$ halt $\Rightarrow \mathcal{M}$ accept any input $\Rightarrow |L(\mathcal{M})| > 10 \Rightarrow \mathcal{M} \in \mathcal{L}$
- If $\langle M, \hat{w} \rangle \notin H_{TM} \Rightarrow M(\hat{w})$ not halt $\Rightarrow \mathcal{M}$ dont accept any input $\Rightarrow |L(\mathcal{M})| < 10 \Rightarrow \mathcal{M} \notin \mathcal{L}$

$$f(\langle M, w \rangle) = \begin{cases} \langle \mathcal{M} \rangle & \text{if } \langle M, w \rangle \text{ is valid input of TM and word} \\ \langle \mathcal{M}_{favourite} \rangle & \text{otherwise} \end{cases}$$

Where $\mathcal{M}_{favourite}$ is my favourite TM that hold $|L(\mathcal{M}_{favourite})| = 10$

(B) $\mathcal{L} = \{M : \langle M \rangle \text{ is a TM that accepts 1 but does not accept 0}\} \in \overline{\mathcal{RE} \cup \text{co-}\mathcal{RE}}$

Lets look at

$$\mathcal{L}_{1\wedge\neg 2} = \{\langle M_1, w_1, M_2, w_2\rangle | w_1 \in L(M_1) \wedge w_2 \notin L(M_2)\}$$

since we know that $\mathcal{L}_{1\wedge\neg 2} \in \overline{\mathcal{RE} \cup \text{co-}\mathcal{RE}}$ we can can apply mapping reduction. Now lets construct a TM $\mathcal{M}_{1\wedge\neg 2}$ that work as follow:

Algorithm 3 $\mathcal{M}_{1\wedge\neg 2}$ on input w.

If $w = 0$ **Then** simulate M_2 on w_2
If $w = 1$ **Then** simulate M_1 on w_1
  If $w \neq 0 \wedge w \neq 1$ **Accept**

I claim that exists maping function f from $\mathcal{L}_{1\wedge\neg 2}$ to $\mathcal{L}$ s.t:

$$f(\langle M_1, w_1, M_2, w_2\rangle) = \begin{cases} \langle \mathcal{M}_{1\wedge\neg 2}\rangle & \text{if } \langle M_1, w_1, M_2, w_2\rangle \text{ is valid input of 2 TMs and 2 words} \\ \langle \mathcal{M}_0\rangle & \text{otherwise} \end{cases}$$

where $\mathcal{M}_0$ is some TM that accept 0

- If $\langle M_1, w_1, M_2, w_2\rangle \in \mathcal{L}_{1\wedge\neg 2} \Rightarrow w_1 \in L(M_1) \wedge w_2 \notin L(M_2)$
  $\Rightarrow 1 \in L(\mathcal{M}_{1\wedge\neg 2}) \wedge 0 \notin L(\mathcal{M}_{1\wedge\neg 2}) \Rightarrow \langle \mathcal{M}_{1\wedge\neg 2}\rangle \in \mathcal{L}$
- If$\langle M_1, w_1, M_2, w_2\rangle \notin \mathcal{L}_{1\wedge\neg 2} \Rightarrow w_1 \in L(M_1) \downarrow w_2 \notin L(M_2)$
  $\Rightarrow 1 \in L(\mathcal{M}_{1\wedge\neg 2}) \downarrow 0 \notin L(\mathcal{M}_{1\wedge\neg 2}) \Rightarrow \langle \mathcal{M}_{1\wedge\neg 2}\rangle \notin \mathcal{L}$

(c) $\mathcal{L} = \{\langle M_1, M_2\rangle : M_1, M_2 \text{ are TMs and } \mathcal{L}(M_1) \cap \mathcal{L}(M_2) = \emptyset\} \in \text{co-}\mathcal{RE}/\mathcal{R}$

Lets describe an algorithm $\mathcal{A}$ that reject for $\mathcal{L}$

Algorithm 4 $\mathcal{A}$ on input $\langle M_1, M_2\rangle$

$k \leftarrow 1$
while $k < \infty$ **do**
  for $i \leftarrow 1$ **to** k **do**
    simulate M_1 and M_2 on w_i for k steps
    If both accept then **Reject**
  end for
end while

Hence its hold that if both M_1, M_2 accept same word then $\mathcal{L}(M_1) \cap \mathcal{L}(M_2) \neq \emptyset$, $\mathcal{L} \in \text{co-}\mathcal{RE}$. Now lets proof $\mathcal{L} \notin \mathcal{R}$ using reduction from $E_{TM} \leq_m \mathcal{L}$. and its will be sufficient to see that $\overline{E_{TM}} \leq_m \overline{\mathcal{L}}$ now lets define ND-TM $\mathcal{M}_E$

Algorithm 5 $\mathcal{M}_E$ on input $\langle \hat{M} \rangle$.

Guess w from Σ^*
Simulate non-deterministic M on w

And let define f s.t:

$$f(\langle M \rangle) = \begin{cases} \langle \mathcal{M}_E, \mathcal{M}_{ACC} \rangle & \text{if } \langle M \rangle \text{ is valid input} \\ \langle \mathcal{M}_E, \mathcal{M}_{\emptyset} \rangle & \text{otherwise} \end{cases}$$

Where $\mathcal{M}_{ACC}$ is TM that accept any input, and $\mathcal{M}_{\emptyset}$ is TM that don't accept any word.

- If $\langle M \rangle \in \overline{E_{TM}} \Rightarrow \mathcal{L}(M) \neq \emptyset \Rightarrow \mathcal{L}(\mathcal{M}_E) \cap \mathcal{L}(\mathcal{M}_{ACC}) \neq \emptyset \Rightarrow \langle \mathcal{M}_E, \mathcal{M}_{ACC} \rangle \in \overline{\mathcal{L}}$
- If $\langle M \rangle \notin \overline{E_{TM}} \Rightarrow \mathcal{L}(M) = \emptyset \Rightarrow \mathcal{L}(\mathcal{M}_E) \cap \mathcal{L}(\mathcal{M}_{ACC}) = \emptyset \Rightarrow \langle \mathcal{M}_E, \mathcal{M}_{ACC} \rangle \notin \overline{\mathcal{L}}$.

(d) $\mathcal{L} = \{\langle M_1, M_2 \rangle : M_1, M_2 \text{ are TMs and } \mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)\} \in \overline{\mathcal{RE}} \cup \text{co-}\overline{\mathcal{RE}}$

Lets look at

$$EQ = \{\langle M_1, M_2 \rangle : \text{are TMs and } \mathcal{L}(M_1) = \mathcal{L}(M_1)\}$$

Since we know that $EQ \in \overline{\mathcal{RE}} \cup \text{co-}\overline{\mathcal{RE}}$ we can can apply mapping reduction $EQ \leq_m \mathcal{L}$. Now lets construct a TM $\mathcal{M}_{EQ}$ that work as follow:

Algorithm 6 $\mathcal{M}_{EQ}$ on input $\langle M_1, M_2 \rangle$.

Run $M_{\mathcal{L}}\langle M_1, M_2 \rangle$ ▷ we assume that exist some TM $M_{\mathcal{L}}$ that halt
if $M_{\mathcal{L}}$ reject **then Reject**
end if
if $M_{\mathcal{L}}$ accept **then** simulate $M_{\mathcal{L}}\langle M_2, M_1 \rangle$
  Answer like $M_{\mathcal{L}}\langle M_2, M_1 \rangle$
end if

We can see that $\mathcal{M}_{EQ}$ decides EQ, since $\mathcal{M}_{EQ}$ accept $\Leftrightarrow \mathcal{L}(M_1) \subseteq \mathcal{L}(M_2) \wedge \mathcal{L}(M_2) \subseteq \mathcal{L}(M_1)$. from the way we construct $\mathcal{M}_{EQ}$ its guaranteed that will act like any $\mathcal{M}_{\mathcal{L}}$ we plug in (halt/loop). Hence if we assume $\mathcal{M}_{EQ}$ halt and accept $\Rightarrow EQ \in \mathcal{R}$. on the other hand if we assume $\mathcal{M}_{EQ}$ halt and reject $\Rightarrow EQ \in \text{co-}\mathcal{RE}$
both lead to contradiction ♣

4 ∼ Assignment 4 (partial) ∼

4.1 Exercise 1

(a)

Definition I. for every $x \in \Sigma^*$

- If $x \in \mathcal{L}$, then there exists $c \in \Sigma^*$ such that V accept (x, c)
- If $x \notin \mathcal{L}$, then V rejects (x, c) for every $c \in \Sigma^*$

Definition II. for every $x \in \Sigma^*$

- If $x \in \mathcal{L}$, then there exists $c \in \Sigma^*$ such that V accept (x, c)
- If $x \notin \mathcal{L}$, then V *does not accept* (x, c) for every $c \in \Sigma^*$

I $\Rightarrow$ **II** first property hold since its identical. and the second since when V reject $(x, c) \Rightarrow$ V *does not accept* (x, c).
II $\Rightarrow$ **I** Let $V_{\mathbf{II}}$ define a verifier. to prove that for some $\mathcal{L}$ exists a verifier by **I**, its sufficient[1] to prove that $\mathcal{L} \in \mathcal{RE}$. now lets define TM that run as follow.

Algorithm 7 $\mathcal{M}_{\mathcal{L}}$ on input x

Let $c_1, c_2 \ldots$ denote lexicographic order for $\forall c \in \Sigma^*$
$i \leftarrow 1$
while $i < \infty$ **do**
  for $j \leftarrow 1$ **to** i **do**
    simulate $V_{\mathbf{II}}(x, c_j)$ for i steps
    Accept if $V_{\mathbf{II}}(x, c_j)$
  end for
end while

Let $x \in \mathcal{L}$ exists V s.t V accept $(x, c) \Rightarrow \mathcal{M}_{\mathcal{L}}$ halt and accept at some point. and If $x \notin \mathcal{L}$ then $\mathcal{M}_{\mathcal{L}}$ *does not accept* matter how many steps we run. Hence $L(\mathcal{M}_{\mathcal{L}}) = \mathcal{L}$

(b) Definition III

- If $x \in \mathcal{L}$, then there exists $c \in \Sigma^*$ such that V accept (x, c)
- If $x \notin \mathcal{L}$, then V rejects (x, c) for every $c \in \Sigma^*$
- V *always halt*

III $\Rightarrow$ **I** is trivial, since its weaker condition.

[1] Recitation 8 ex 3

I $\Rightarrow$ III Let V define a verifier for some $\mathcal{L}$,same as above by **Def 1** its following that $\mathcal{L} \in \mathcal{RE}$, and let $M_{\mathcal{L}}$ define some TM that accept it. we can write the following:

$$\mathcal{L} = \{x : \exists c \text{ such that } (x, c) \in L(V)\}$$

where V has the property that given input $(x, c), V$ reject any input if $x \notin \mathcal{L}$.

Algorithm 8 V_{III} on input (x, c)

If (x,c) is invalid input then **Reject**
Emulate in parallel $M_{\mathcal{L}}(x)$ and $V(x, c)$
Accept if $M_{\mathcal{L}}$ accept
Reject if V Reject

By the we define the verifier if $x \in \mathcal{L}$ $M_{\mathcal{L}}$ halt and accept. and if $x \notin \mathcal{L}$ then V reject any input i.e halt and reject . Hence V *always halt*

4.2 Exercise 2

Let $\mathcal{C} \subset \mathcal{RE}$ be such that $\emptyset \in C$. then

$$\mathcal{L}_{\mathcal{C}} = \{\langle M \rangle : M \text{ is a TM and } \mathcal{L}(M) \in \mathcal{C}\} \notin \mathcal{RE}$$

Since $\mathcal{C} \subset \mathcal{RE}$ then exist some $\mathcal{L}' \in \mathcal{RE}$ such that $\mathcal{L}' \notin \mathcal{C}$. Let $M_{\mathcal{L}'}$ denote TM that accept $\mathcal{L}'$

5 $\sim$ Assignment 5 $\sim$

5.1 Exercise 1

Assuming $P \neq NP$

(a) Input: Sets $A_1, ..., A_n$, and a number k.
Question: do there exist k mutually disjoint sets $A_{i1}, ..., A_{ik}$
not in $\mathcal{P}$

Proof. we can reduce IS $\leq_P$ MDS as follow. let $G(V, E)$ denote an instant of IS problem, for any $v_i \in V$ generate set A_i that contain all edges that have an end in v_i i.e $v_i v_j \in A_i \Leftrightarrow v_i v_j \in E$. Its immediate that there is IND-SET $v_{i1}, ..., v_{ik}$ size k in G iff there exist k mutually disjoint sets $A_{i1}, ..., A_{ik}$. The reduction f is poly-time computable function in $O(|V| + |E|)$, since we just need travel on G and add the elements to the sets as described above. Additionally, given a sets, we can verify if they are disjoint and there are at least k of them in poly time. Hence MDS$\notin P$ $\square$

(b) Input: Sets $A_1, ..., A_n$.
Question: do there exist 3 mutually disjoint sets A_i, A_j, A_k
in $\mathcal{P}$

claim. $3IND^2 \in \mathcal{P}$

Proof. First notice 3IND$\in NP$ using same verifier of IS. Given $G(V,E)$ denote $|V| = n, |E| = m$ we can label its vertex $1 \dots n$. run STCON on $\langle G, v_i, v_j \rangle$ for any i, j .now by choose any triplet from $\{1 \dots n\}^3$ and check if STCON$\langle G, v_i, v_j \rangle$ return 0 for any $i, j \in \{i, j, l\}$ in total we run $\binom{n}{2}$ times STCON save result on the tape size $\binom{n}{2}$ and check the nation for any triplet in total $\binom{n}{3}$. all the following is can be compute in poly time. hence 3-IND$\in$P

□

Hence using same reduction described in (a) its holds that $3-\text{IND} \leq_P 3-\text{MDS}$.

(c) Input: Sets $A_1, ..., A_n$, and a number k.
Question: do there exist k sets such that $A_{i1}, ..., A_{ik}$ such that $A_i \cap A_j \neq \emptyset$
not in $\mathcal{P}$

Proof. we can reduce CLIC $\leq_P$ $Q_{(c)}$ as follow. let $G = (V,E)$ denote an instant of CLIC problem. using same proses as described at (a) we generate the sets $A_{vi}, ..., A_{vn}$. Hence if G have clic size k then exists $v_1 \dots v_k$ s.t $(v_1v_2)(v_2, v_3) \dots (v_{k-1}v_k) \in E \Leftrightarrow$ for any $e = v_iv_j$ exists $x \in A_i, A_J \Leftrightarrow A_i \cap A_j$ for any $i \neq j$. □

(d) CNF_{50v}
Input: a CNF formula ψ with at most 50 variables.
Question: does there exist an assignment that satisfies ψ
in $\mathcal{P}$

Proof. Let $\mathcal{A}$ denote algorithm that get as input CNF formula ψ. If ψ contain more then 50 variables then $\mathcal{A}$ immediately reject, otherwise its "brute force" by set any possible combination of boolean assignment for the given variables, If one of them satisfy ψ its ACCEPT, and if none of them satisfy ψ then REJECT. Since from 50 variables there are total of 2^{50} possible assignment, $\mathcal{A}$ is polynomial algorithms that decide our problem. □

(e) Input: a CNF formula ψ with at most 50 clauses.
Question: does there exist an assignment that satisfies ψ
in $\mathcal{P}$

Proof. Let ψ be a CNF formula with at most 50 clauses on n literals. Consider some algorithm $\mathcal{A}(\psi)$ that try all possible ways to choose one literal from each clause whenever $x_i, \neg x_i$ does **Not** appear together $\forall i$. If $\mathcal{A}$ find such an assignment $\Rightarrow \psi$ satisfies, otherwise it is not .Since there are at most $(2n)^{50}$ ways to choose such an assignment, $\mathcal{A}$ is polynomial algorithms that decide our problem. □

[2]decide if given graph have independence set size 3

(f) Input: a 3CNF formula ψ with even number of variables.
Question: does there exist an assignment that satisfies ψ and gives $True$ for exactly one half of the variables?
not in $\mathcal{P}$

Proof. we can reduce 3SAT $\leq_P$ 3CNF$_{half}$ as follow. Given 3SAT formula ψ that have n variable $x_1 \dots x_n$, now construct new n variable $y_1 \dots y_n$ such that each y_i correspond to the opposite value of x_i, its can done by adding to ψ 2-CNF clauses i.e

$$(x_1 \vee y_1)(\overline{x_1} \vee \overline{y_1}) \dots (x_n \vee y_n)(\overline{x_n} \vee \overline{y_n})$$

Any solutions to the formula will have half the variables with true values and half false. the following can done in poly-time with computable function just duplicate ψ variables and add the 2-CNF clauses. □

(g) Input: undirected graph G, a number k.
Question: does there exist in G a clique of size at least k or an independent set of size at least k?
not in $\mathcal{P}$

Proof. we can reduce IS $\leq_P$ CLIC$\vee$IS as follow. let $G = (V, E)$ instant of IS problem, we can construct new graph G' by adding set of additional $|V'| = |V|$ isolated vertexes. now define

$$f(\langle G = (V, E), k \rangle) = \langle G' = (V + V', E), k + |V| \rangle$$

f is polynomial. Since $k + |V| > |V|$ then G' don't have CLIC size $k + |V|$ for sure. and exist IS size k in $G \Leftrightarrow$ exist IS size $k + |V|$ in G'. Hence CLIC$\vee$IS $\notin$P □

5.2 Exercise 2

(a)

For every nontrivial $L_1, L_2 \in P, L_1 \leq L_2$
TRUE

Let $L_1, L_2 \in P$ W.L.O.G we can generate arbitrary f that yield $L_1 \leq_P L_2$

Algorithm 9 f on input w .

Decide $w \in L_1$
If $w \in L_1$ Return any $w' \in L_2$
If $w \notin L_1$ Return any $w'' \notin L_2$

First line can done in poly-time since $L_1 \in P$, 2$^{\text{nd}}$ holds because $L_2 \neq \emptyset$ and 3$^{\text{nd}}$ since $L_2 \neq \Sigma^*$. All the following can done in poly-time hence f poly-time computable reduction function from $L_1 \leq_P L_2$

(b) For every nontrivial $L_1, L_2 \in NP, L_1 \leq_P L_2$
Equivalent to an Open Problem

claim. *$P = NP \Leftrightarrow$ claim 2(b) is True*

Proof. $\Rightarrow$ if $P = NP$ any $L_1, L_2 \in NP \Rightarrow L_1, L_2 \in P \Rightarrow$ Using 2(a) any non-trivial $L_1, L_2 \leq P$ can reduce to any other $\Rightarrow L_1 \leq_p L_2$ claim 2(b) Is true.

$\Leftarrow$ Consider some non trivial $L_2 \in P$, and assume that exist non trivial $L_1 \in NP/P$. Since $P \subseteq NP$ then $L_2 \in NP \Rightarrow$ claim 2(b) yield that exists reduction s.t $L_1 \leq_P L_2$ then $L_1 \in P$ but $L_1 \in NP/P$ and we get an contradiction $\Rightarrow$ there is no exist such $L_1 \Rightarrow NP/P = \emptyset \Rightarrow NP \subseteq P \Rightarrow P = NP$ □

(c) There exists a language in $\mathcal{RE}$ that is complete w.r.t polynomial-time reductions.
TRUE.

claim. *A_{TM} is complete w.r.t polynomial-time reductions.*

Proof. consider $f(w) = \langle \mathcal{M}, w \rangle$ for any $L \in \mathcal{RE}$ and TM $\mathcal{M}$ that accept L. f computable, and able to write the encode of $\mathcal{M}, w$ in poly time. Since $w \in L \Leftrightarrow \langle \mathcal{M}, w \rangle \in A_{TM}$, f define poly-time reduction for any arbitrary $L \in \mathcal{RE}$. A_{TM} is complete w.r.t polynomial-time reductions □

(d) If there exists a deterministic TM that decides SAT in time $n^{O(\log n)}$
Then every $L \in NP$ is decidable by a deterministic TM in time $n^{O(\log n)}$.
TRUE.

Proof. Let $L \in \mathcal{NP}$ and $\mathcal{M}_{SAT}$ denote D-TM that decide SAT. Since $SAT \in \mathcal{NPC}$ then exist some poly-time reduction f s.t $L \leq_P SAT$ decide SAT. Let look at d-TM $\mathcal{M}'$.

Algorithm 10 $\mathcal{M}'$ on input w .
Computes $f(w)$
Simulate $\mathcal{M}_{SAT}(f(w))$ and **Answer** like $\mathcal{M}_{SAT}$

$w \in L \Leftrightarrow \mathcal{M}_{SAT}$ accept $f(w) \Leftrightarrow f(w) \in SAT$. Since $f \in \mathcal{O}(n) \Rightarrow |f(w)| = n^k$ and $\mathcal{M}_{SAT} \in \mathcal{O}(n^{O(\log n)})$ its following that

$$\mathcal{M}'(w) = \mathcal{M}_{SAT}(f(w)) \in {}^{3}\mathcal{O}(\mathcal{M}_{SAT}(n^k)) = \mathcal{O}(n^k)^{O(\log n^k)} = \mathcal{O}(n^{O(\log n)})$$

□

[3] Not realy sure if its the proper way to write it.

5.3 Exercise 3

claim. *if $P = NP$, there exists a polynomial-time TM, that given a $3CNF$ formula ψ , outputs a satisfying assignment for ψ or indicates one does not exists.*

Proof. first notice that if $P = NP$ then $NP \subseteq P$, since $3CNF \in NP$ then $3CNF \in P$. Its following that exist polynomial-time TM $\mathcal{M}_{3CNF}$ that decide 3CNF. Now let us define TM $\mathcal{M}'$ that given a formula $\psi = \alpha(x_1, x_2 \dots x_N)$ work such that:

Algorithm 11 $\mathcal{M}'$ on input ψ .

---

check if ψ is valid 3CNF formula, otherwise **Reject**
Simulate $\mathcal{M}_{3CNF}\langle\psi\rangle$ and **Reject** if $\mathcal{M}_{3CNF}\langle\psi\rangle$ **Reject**
$\alpha(x_1, x_2 \dots x_N) \leftarrow \psi$ ▷ α represent the logic relation w.r.t ψ
$y_j \leftarrow 0 \qquad 1 \leq j \leq N$
$i \leftarrow 1$
while $i \leq N$ **do**
  Simulate $\mathcal{M}_{3CNF}\langle\alpha(y_1, y_2 \dots y_{i-1}, T, x_{i+1} \dots x_N)\rangle$
  if $\mathcal{M}_{3CNF}$ Accept **then**
    $y_i \leftarrow T$
  else ▷ $\mathcal{M}_{3CNF}$ reject
    $y_i \leftarrow F$
  end if
  $\alpha(x_i) \leftarrow y_i$
end while
output $y_i \dots y_N$

If ψ does not have an boolean satisfy assignment then $\mathcal{M}'$ indicate that at step 2. The correctness of $\mathcal{M}'$ following from the "greedy" posses, i.e before any iteration α can be satisfied. Hence by assign each time one of the veritable to T, we check if $\mathcal{M}_{3CNF}$ accept. if its accept the assignment its satisfied, and if its reject then the value F is the valid assignment. $\mathcal{M}'$ run $N + 1 \times \mathcal{M}_{3CNF}$ that is poly-time TM that output a satisfying assignment for ψ or indicates one does not exists.

□

5.4 Exercise 4

We say that a polynomial reduction f is a *shrinking reduction* if there exists n_0 such that for every $x \in \Sigma^*$ such that $n_0 \leq x, |f(x)| \leq |x| - 1$. Assuming $P \neq NP$

(a) For every two nontrivial languages $A, B \in P$ there exists a *shrinking reduction* from A to B.
Prove

Proof. using Q2(a) between any non trivial $A, B \in P$ exists mapping reduction. Since both non trivial, then exists some $b \in B, \bar{b} \notin B$. Consider f s.t

$$f(w) = \begin{cases} b & \text{if } w \in A \\ \bar{b} & \text{if } w \notin A \end{cases}$$

Its immediate that $f(w) \in B \Leftrightarrow f(w) = b \Leftrightarrow w \in A$. and f define valid poly-reduction. Let us define $n_0 = \max\{|\bar{b}|, |b|\} + 1$, we can notice that f define *shrinking reduction* from A to B since

$$x \in \Sigma^* \text{ s.t } n_0 \leq |x| \qquad |f(x)| \leq \max\{|\bar{b}|, |b|\} \leq n_0 - 1 \leq |x| - 1$$

□

(b)

For every two nontrivial languages $A, B \in NPC$ there exists a *shrinking reduction* from A to B.
Disprove

Proof. let A be any non trivial language s.t $A \in NPC$. By consider the following claim is true, W.L.O.G we can choose $A = B$. then exist $shrinking - reduction$ f with some n_0 such that $A \leq_P A$. First notice that there is only finite many w s.t $|w| < n_0$. We can encode them all correct answers to an algorithm $\mathcal{A}$. For given $x \in \Sigma^*$ if $|x| < n_0$ then its encoded already to $\mathcal{A}$. Since $f(x) < |x|$, when $|x| \geq n_0$ we can apply finite time of $ff\ldots(x)$ until we achieve some $|w| < n_0$. All can done in poly-time since f is poly-reduction. Hence $\mathcal{A}$ decide any non trivial $A \in NPC$ in polynomial time, its following that $P = NP$, Contradiction. □

5.5 Exercise 5

The following languages are NPC:

(a)

$$EXACT3SAT = \{\varphi \in 3SAT : \text{every clause of } \varphi \text{ has exactly 3 distinct variables}\}$$

The verifier for SAT is valid poly-time verifier for EX-3SAT$\in NP$. We can reduce 3SAT$\leq_P$EX-3SAT. Given instant of SAT $\psi = C_1 \wedge C_2 \ldots C_n$ for any clause C_i, define f work as follows

- If $C = \emptyset$ i.e ψ have empty clause then it is unsatisfiable. then f return any possible combination that can generate using 3 distinct variables i.e 2^3 clauses of 3EX-SAT

$$f(\emptyset) = (x \vee y \vee z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee z) \wedge (x \vee \neg y \vee \neg z) \wedge (\neg x \vee y \vee z) \wedge (\neg x \vee y \vee \neg z) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$$

$\emptyset \equiv f(\emptyset)$

- $C = (x)$ when C have just one literal. Let us define f such that

$$f((x)) = (x \vee y \vee z) \wedge (x \vee \neg y \vee z) \wedge (x \vee y \vee \neg z) \wedge (x \vee \neg y \vee \neg z)$$

  $(x) \equiv f(x)$
- $C = (x \vee y)$ when C have just 2 literal. Let us define f such that

$$f((x \vee y)) = (x \vee y \vee z) \wedge (x \vee y \vee \neg z)$$

  $(x \vee y) \equiv f((x \vee y))$
- $C = (x \vee y \vee z)$ then $f(C) = C$

Hence let $f'(\psi) = f'(C_1 \wedge C_2 \dots C_n) = f(C_1) \wedge f(C_2) \dots f(C_n)$ define poly-time reduction from 3SAT to EX-3SAT. Following that EX-3SAT$\in NPC$

(b)

$$L_2 = \{\langle M, 1^n \rangle : \text{M is a TM and there exists a string that M accepts in n steps}\}$$

Proof. First consider some $w = \langle M, 1^n \rangle$ we can verify that $w \in L_2$, by simulate M on any input size n that is[4] $n^{|\Sigma^*|} \in \mathcal{O}(n)$, and in total $\mathcal{O}(n^{|\Sigma^*|}|M|^3 n \log n) \Rightarrow L_2 \in NP$
Let L be any $L \in NP$, and V_L denote its polynomial verifier, and assume its runtime is $p(|x|)$. Any such L can reduce $L \leq_P L_2$ as follows

$$x \in L \Leftrightarrow \exists c \quad \text{s.t } (x, c) \in V_L \Leftrightarrow f(x) = \langle V_L, 1^{p(|x|)} \rangle \in L_2$$

The correctness followed by the verifier definition, and f define computable poly-reduction from any $L \in NP$, Hence $L_2 \in NPC$ □

6 ∼ Assignment 6 (Partial) ∼

6.1 Exercise 1

Let proof that

$$\overline{2SAT} \leq_P PCON$$

(a) We can allocate space on the working tape for counters of the variables and the assignment, and all can done in $O(\log n)$ space.

(b) If $\langle G, P \rangle \in PCON \Rightarrow$ exist some $(u, v) \in P$ s.t $u \rightsquigarrow v$ and $v \rightsquigarrow u \Rightarrow$ exist cycle $C \subseteq G$ such that $u, v\neg, u, \neg v \in C$ and $v \rightsquigarrow \neg v \rightsquigarrow v \in G \Rightarrow$ any $x_1 \dots x_v \dots x_n$ that satisfies φ following that $x_1 \dots \neg x_v \dots x_n$ satisfies φ as well $\Rightarrow v \equiv \neg v$ contradiction.

[4] I assumed a finite alphabet

(c) [5] If $\langle G, P\rangle \notin PCON$ then t is well defined, since if $v \rightsquigarrow \neg u, v \rightsquigarrow u$ then by f definition $v \rightsquigarrow u \rightsquigarrow \neg v$. and when we looking at closure size 1 (u) then both $u, \neg u$ reachable as well. Hence if $\langle G, P\rangle \notin PCON$ then $P \not\subseteq G \Rightarrow$ for any u, v exist e s.t $e \in (V \times V), e \notin E$ $t(v) = 1$ then $v \not\rightsquigarrow \neg v$ for any $v \in G$. then $\forall \ell_1 \in \varphi$ we get that $t(\ell) \bigoplus t(\neg \ell) = 1$. and we can consider it as

$$t = \left(t(\ell_1) \bigoplus t(\neg \ell_1)\right) \vee \left(t(\ell_2) \bigoplus t(\neg \ell_2)\right) \dots$$

And its immediate that t satisfies φ

6.2 Exercise 3

$ZPP \subseteq RP \cap coRP$

Proof. Let $\mathcal{L} \in ZPP \Rightarrow$ there exists an expected poly-time TM $\mathcal{M}$ that decides $\mathcal{L}$ using Markov's inequality $\Pr(x \geq a) \leq \frac{E[x]}{a} \leq \frac{p(|x|)}{a} \Rightarrow \Pr(x \geq 2p(|x|)) \leq \frac{1}{2}$
Let describe an algorithm A_{RP} on input x

Algorithm 12 A_{RP} on input x

Simulate $\mathcal{M}(x)$ for at most $2p(x)$ steps.
If $\mathcal{M}(x)$ halt answer like $\mathcal{M}(x)$
Otherwise **Reject**

Its immediate that A_{RP} always reject for $x \notin L$, and by the ineq above A_{RP} accept $x \in \mathcal{L}$ with probability $\geq 1/2$, hence $ZPP \subseteq RP$. using same algorithm described above with flip the nation in line 3 from Reject to Accept, will give us same result for $coRP$. □

[5] t refer v in the original exercise. since i used v already