CS340

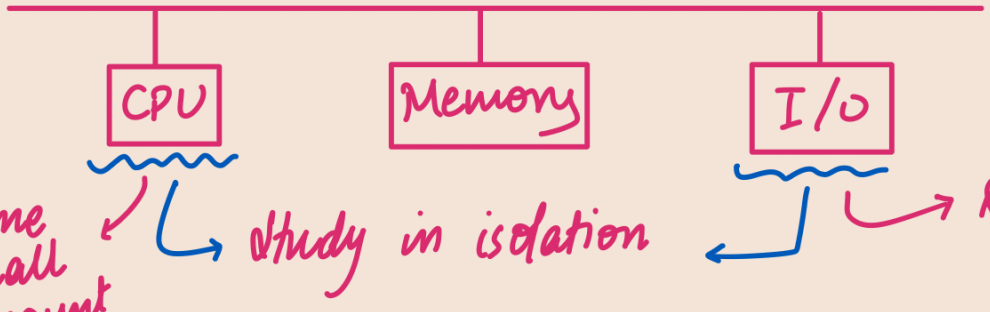
Course Logistics

Assgn (4): 25%, MidSem: 25%, EndSem: 50%

Grading: 80% + ⇒ A, 20% + ⇒ D+

Textbooks: By Michael Sipser (Intro. to the T.o.C.)
By Dexter Kozen (Automata & Computability)

What is a computer?



Some small amount of memory

Study in isolation

Rudimentary

CPU + I/O → CPU + I/O + Limited Memory (Stack) → CPU + I/O + Memory {Analysis}

Formalism

- A computer computes functions.
  - Interested in knowing the class of functions
  - $f: D \rightarrow R$ (Domain) (Range)
- Domain / Range: A sequence of bits
- Sequence of bits
  - Alphabet: A set of symbols. Typically denoted as Σ

Ex: $\Sigma = \{0,1\}$, $\Sigma = \{a, b, c, \ldots, x, y, z\}$,
$\Sigma = \{0, 1, \ldots, 8, 9\}$, etc.

- **Strings**: A finite sequence of symbols from an alphabet.
  Denoted as Σ^*
  $\Sigma^* = \{ a_1 a_2 \ldots a_m \mid a_i \in \Sigma \}$

  Ex: $\{0,1\}^*$: Set of all binary strings
  $\{0,1,\ldots,9\}^*$: Set of all non-negative numbers
  $\{a, b, \ldots, y, z\}^*$: Set of all words using the English Alphabet.

- $D : \{0,1\}^*$, $R : \{0,1\}^*$
- Let F be the class of functions computed by a computer.
  For any function $f : \Sigma^* \to \Sigma^*$, define
  $A_f = \{ (x, y) \mid f(x) = y \}$
- A **set** is a special function: $f : \Sigma^* \to \{0,1\}$
  ↳ $\{ x \mid f(x) = 1 \}$
- A string y is a **prefix** of string z if $z = yz'$ for some string z'.

Ex. Define $A_f = \{(x, y) \mid f(x) = yz, \text{ for some string } z\}$
→ y is a prefix of $f(x)$
→ Computing f from A_f
- Given x, check if $(x, 0) \in A_f$ OR $(x, 1) \in A_f$
- If $(x, 0) \in A_f$, check for $(x, 00) \in A_f$ OR $(x, 01) \in A_f$, etc.

- Order of Sets with (x, y) is the same as Order of Real Numbers.
  But. Order of Computable sets (= Programs) is the same as the order of Natural Numbers.

Consider CPU at a specific stage of computation.
Contents of its registers define the state of CPU at that stage.
CPU has finitely many states independent of input. (Input is also finite)

Q: The set of states of CPU

q_0 / s_0: Starting state of CPU.

δ: $\delta : Q \times \Sigma \cup \{\epsilon\} \longrightarrow Q$, Transitions

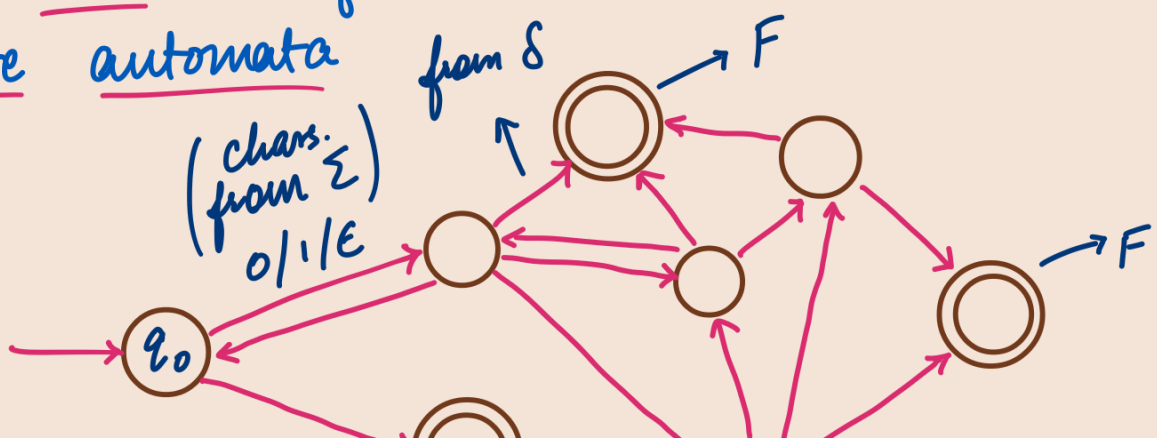
(ϵ: Null or empty string)

F: $F \subseteq Q$, Final states

Σ: Alphabet

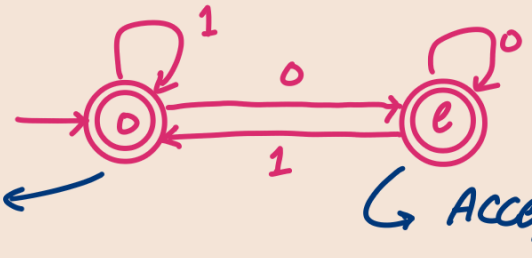
Def. A finite automata is described by 5-tuple $(Q, q_0, \Sigma, \delta, F)$.

Representation of finite automata



Ex. Set of even numbers written in binary

$A = \{ s0 \mid s \in \{0,1\}^* \}$



- We have to label the states "reject" and "accept": What if we didn't need to?
  ⇝ Only "accept"s included in final state and then if automata lands on them, it is accepted.

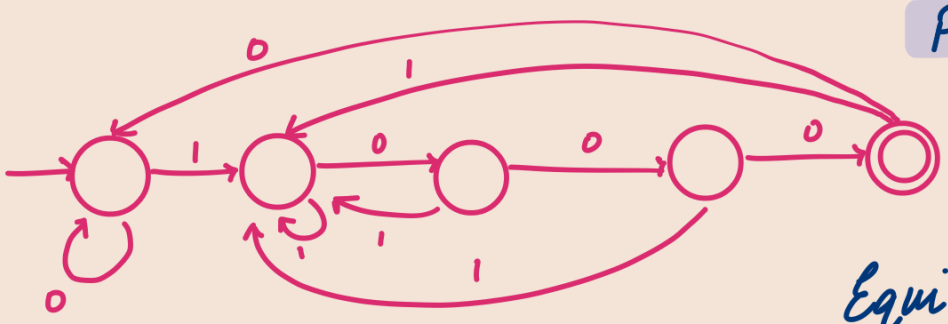
Lemma: A set accepted by a finite automata is also accepted by a finite automata in which every state is either accepting or rejecting.

Pf:

- From now on, F: Only accepting states

Ex. $A = \{ s1000 \mid s \in \{0,1\}^* \}$

Deterministic Finite Automata (DFA)



Equivalent in terms of inputs accepted

Non-deterministic Finite Automata (NFA)

☆ A deterministic finite automata is a 5-tuple $(Q, q_0, \Sigma, \delta, F)$ where

1) $F \subseteq Q$ is a set of accepting states
2) $\delta: Q \times \Sigma \longrightarrow Q$

DFA accepts an input x if starting from q_0, it ends up in a state of F.

A nondeterministic finite automata is a 5-tuple $(Q, q_0, \Sigma, \delta, F)$ such that

1) $F \subseteq Q$ is accepting states
2) $\delta: Q \times \Sigma \cup \{\epsilon\} \longrightarrow 2^Q$

NFA accepts an input x if starting from q_0, it is possible to end up in a state of F.

Ex.

Suppose F_1, F_2 accept sets A_1 & A_2
Design an automata to accept $A_1 \cup A_2$.

Let $F_1 = (Q_1, q_0', \Sigma, \delta_1, F_1)$ &

$F_2 = (Q_2, q_0^2, \Sigma, \delta_2, F_2)$

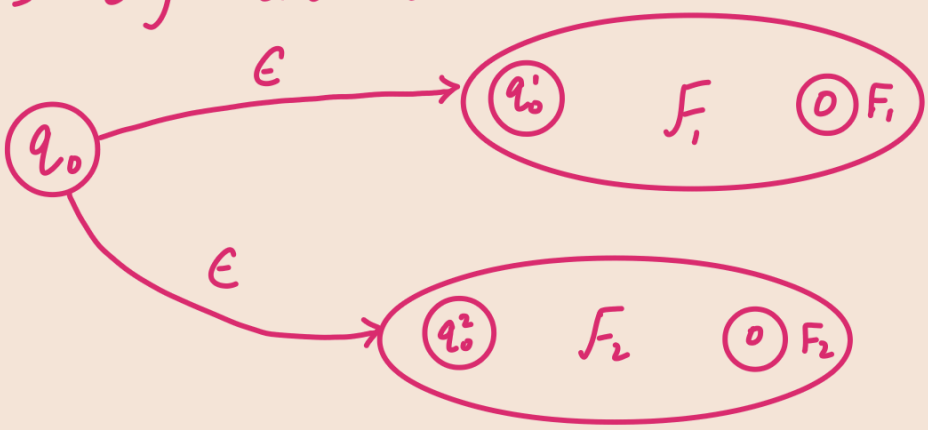
$F = (Q, q_0, \Sigma, \delta, F)$

$F = (Q_1 \times Q_2, \langle q_0', q_0^2 \rangle, \Sigma, \langle \delta_1, \delta_2 \rangle, F_1 \times Q_2 \cup Q_1 \times F_2)$

$\langle \delta_1, \delta_2 \rangle := \langle q_1, q_2 \rangle \xrightarrow{a} \langle \delta_1(q_1, a), \delta(q_2, a) \rangle$

For $A_1 \cap A_2$ $\quad F = F_1 \times F_2$

→ Let's try the same with an NFA



For $A_1 \cap A_2$, NFA is not much better than a DFA

Theorem: A set accepted by an NFA is also accepted by a DFA

Pf: Let $F = (Q, q_0, \Sigma, \delta, F)$ be an NFA.

Define $F_D = (2^Q, \{q_0\}, \Sigma, \delta_D, F_D)$

$F_D = \{ H \mid H \subseteq Q \text{ & } H \cap F \neq \phi \}$

↳ At least one state from F should be in H/ be reachable

$\delta_D(H, a) := \{ \delta(q, a) \mid q = \delta(p, \epsilon_0)$ for $p \in H$ & $\epsilon_0 \in \{\epsilon\}^* \}$

OR $:= \{ \delta(q, a) \mid q$ is reachable through ϵ transitions $\}$

Def. Let F be a finite automata. The set of strings accepted by F is called a regular set.

Properties of Regular Set

1) Closed under union and intersection

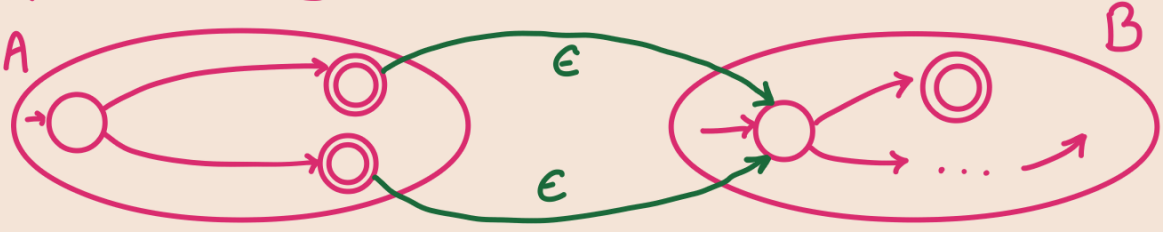
Construction of suitable FSM was seen earlier

2) Closed under **complement**:

If $A \subseteq \Sigma^*$ is a regular set, then $\Sigma^* - A$ is also a regular set

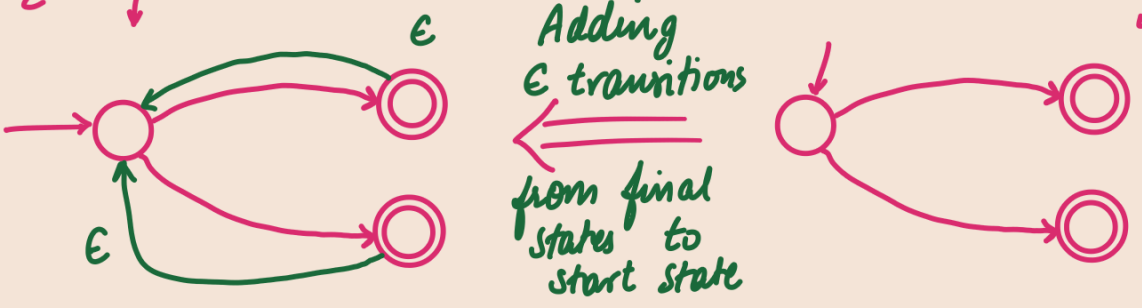
3) Closed under $\cdot$ operation (**Concatenation**)

$A \cdot B = \{ xy \mid x \in A, y \in B \}$



4) Closed under * **operation**

$\Sigma^* \leftarrow A^* = \{ y \mid y = x_1 x_2 \dots x_k, x_i \in A \}$ ← Σ



Regular Expressions (Reg Ex)

Regular Expression over alphabet Σ is defined as:

(i) $a \in \Sigma$ is a regex

(ii) ϵ is ...

(ii) ε is r.e.

(iii) If r_1, r_2 are r.e., so is $r_1 + r_2$

(iv) " " " " , " " $r_1 \cdot r_2$

(v) If r is " , " " r^*

A regex r defines a set **L(r)** in Σ^* as follows

(i) $r = a \in \Sigma \longmapsto L(r) = \{a\}$

(ii) $r = \epsilon \longmapsto L(r) = \{\epsilon\}$

(iii) $r = r_1 + r_2 \longmapsto L(r) = L(r_1) \cup L(r_2)$

(iv) $r = r_1 \cdot r_2 \longmapsto L(r) = L(r_1) \cdot L(r_2)$

(v) $r = r_1^* \longmapsto L(r) = L(r_1)^*$

Ex. (i) $(0+1)^*$: $\{0,1\}^*$ {All binary strings}

(ii) $1(0+1)^*0$: All strings starting with 1 and ending with 0

- Reg Ex — Made by $\Sigma, +, \cdot, *$
- Regular Set — Accepted by FA

Theorem: Set A is regular iff $A = L(r)$ for some regular expression r

Pf:

Suppose A is accepted by DFA F.

A **generalized** **NFA** **(GNFA)** is a finite automata where transitions are done on regular expressions.

Formally,

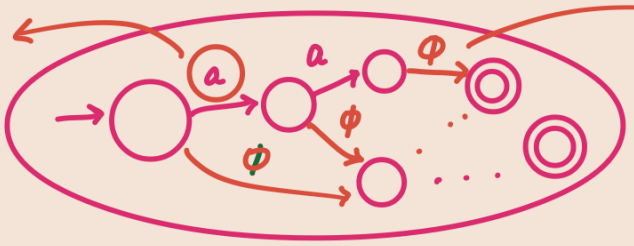
$$GNFA = (Q, \Sigma, q_0, \delta, F)$$

with $\delta : Q \times R \to Q$

where R = Set of all regex over Σ and also includes ϕ

Observation: We can easily convert F to a GNFA

View it as a regex

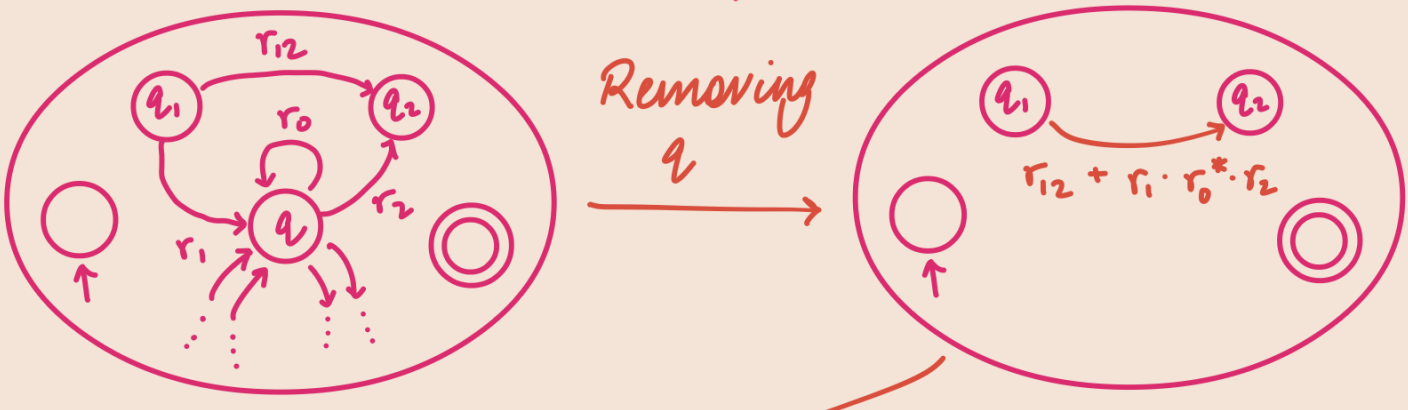


Replace : no transition : by : transition on ϕ :

$F \longrightarrow$ GNFA

We intend to remove states & modify transitions so that eventually, we are only left w/

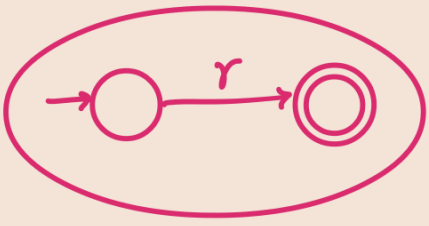
$$q_0 \xrightarrow{\text{RegEx}} q_f$$



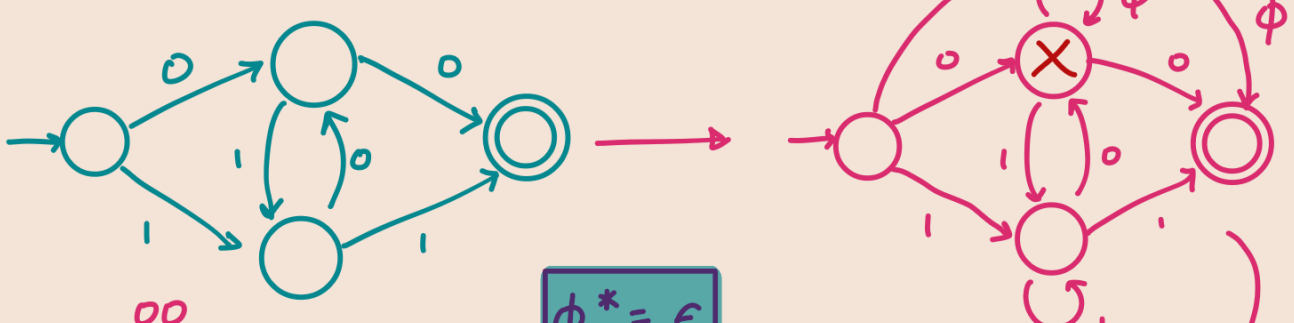
Repeat this & keep removing intermediate states and terminate at

Thus, A being a regular set

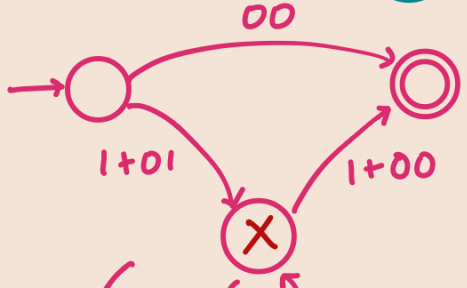
$\Rightarrow \exists\ r \mid L(r) = A$



Ex.

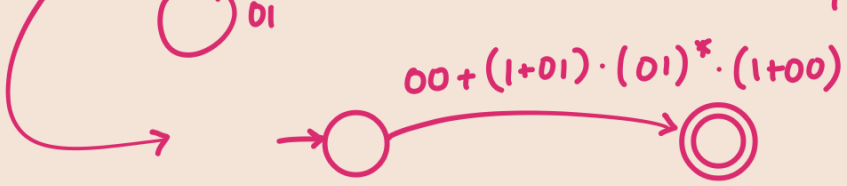


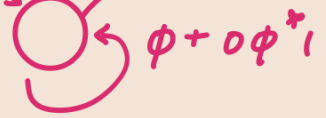
$\phi^* = \epsilon$



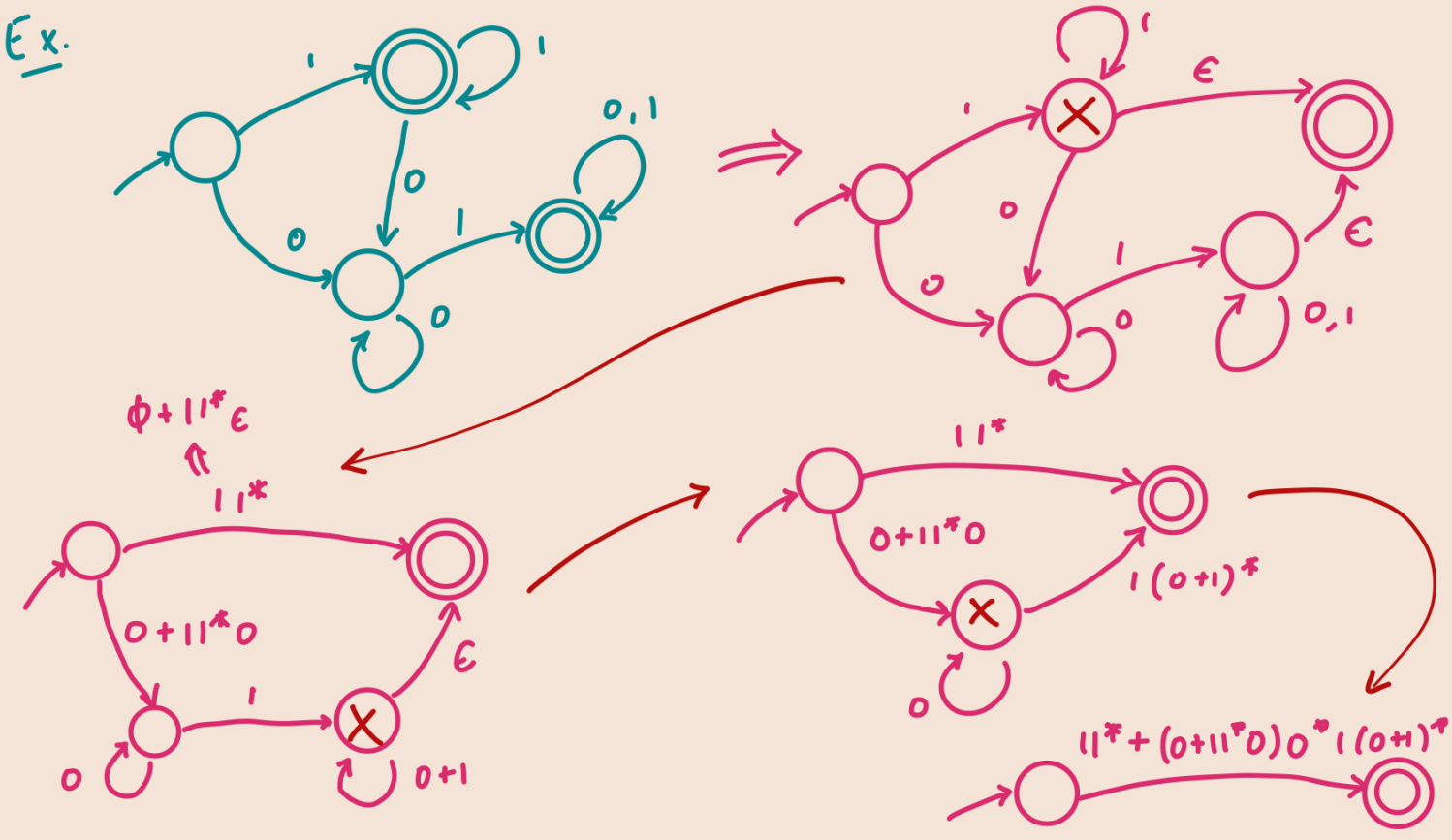
$\phi + r = r$

$\phi + 0\phi^*0$





Ex.



Myhill - Nerode Theorem

For any regular set A, there is a unique deterministic automata with minimum number of states accepting it

Pf: Let A be a regular set.
Define a relation R on it as follows:

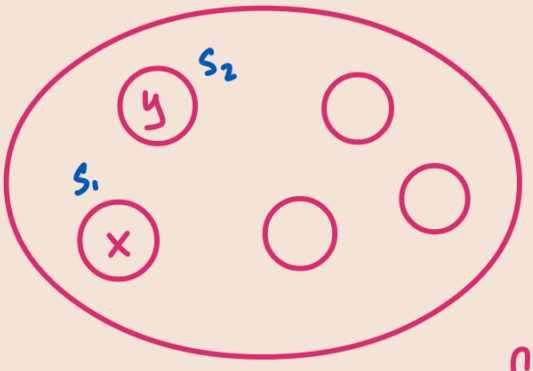
$x \, R \, y$ iff for every z, $xz \in A$ iff $yz \in A$

R is an equivalence relation:

- (Reflexive) $x \, R \, x$
- (Symmetric) $x \, R \, y \Rightarrow y \, R \, x$
- (Transitive) $x \, R \, y \;\&\; y \, R \, v \Rightarrow x \, R \, v$

↳ Consider any w & xw, zw, then,
$xw \in A \Rightarrow yw \in A$; $yw \in A \Rightarrow zw \in A$
$\therefore$ $xw \in A \Rightarrow zw \in A$; Similarly $zw \in A \Rightarrow xw \in A$

Consider equivalence classes formed by R on Σ^*



Can some input bring x & y to the same state? $[x \not R y]$

No.

Suppose $\exists$ w s.t. either
$xw \in A$ & $yw \notin A$ or $xw \notin A$ & $yw \in A$ ⟶ xw will be taken to a final (= accept) state but yw won't or vice-versa

⇒ Different eq. classes will take a DFA accepting A to different states
$\therefore$ Label the equivalence classes with:
S_i: Set of states that DFA ends up in on reaching the i^{th} equivalence class

Also,

$$S_i \cap S_j = \phi \quad (\text{for } i \neq j)$$
& $$\bigcup S_i = Q \quad (\text{the set of states in the DFA})$$

A new DFA based on equivalence classes:
Let $E =$ Set of all equivalence classes, each represented as $[x]$ where x belongs to the class

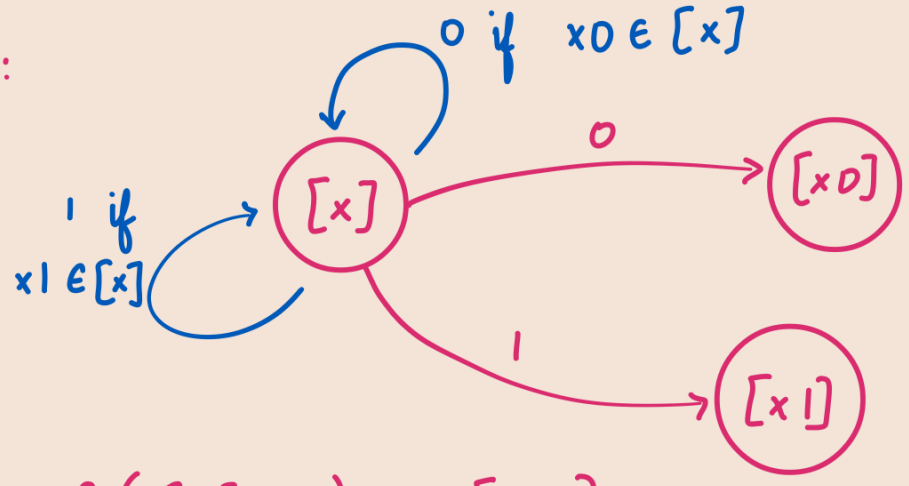
Define

$$F = (E, [\epsilon], \delta, F, \Sigma)$$

F: Set of final states ≡ Set of equivalence classes which contain strings accepted by DFA

$$F = \{ [x] \mid [x] \cap A \neq \phi \}$$

δ:



$$\delta([x], a) = [xa]$$

Lemma: F on input x ends up in state [x]

Pf: By induction on $|x|$ = length of x

BS: $|x| = 0 \iff x = \epsilon$,
automata is in state $[\epsilon]$

IS: Assume for $|x| < n$,
consider x, $|x| = n$. Let $x = ya$,
$|y| = n-1$; On reading y, F ends up in $[y] \Rightarrow \delta([y], a) = [ya] = [x]$

∴

on reading $x \in A$, F ends up in $[x]$ & $[x] \in F$.

On reading $x \notin A$, F ends up in $[x]$ & $[x] \notin F$.

Wkt any deterministic automata accepting A, will have number of states $\geq |E|$

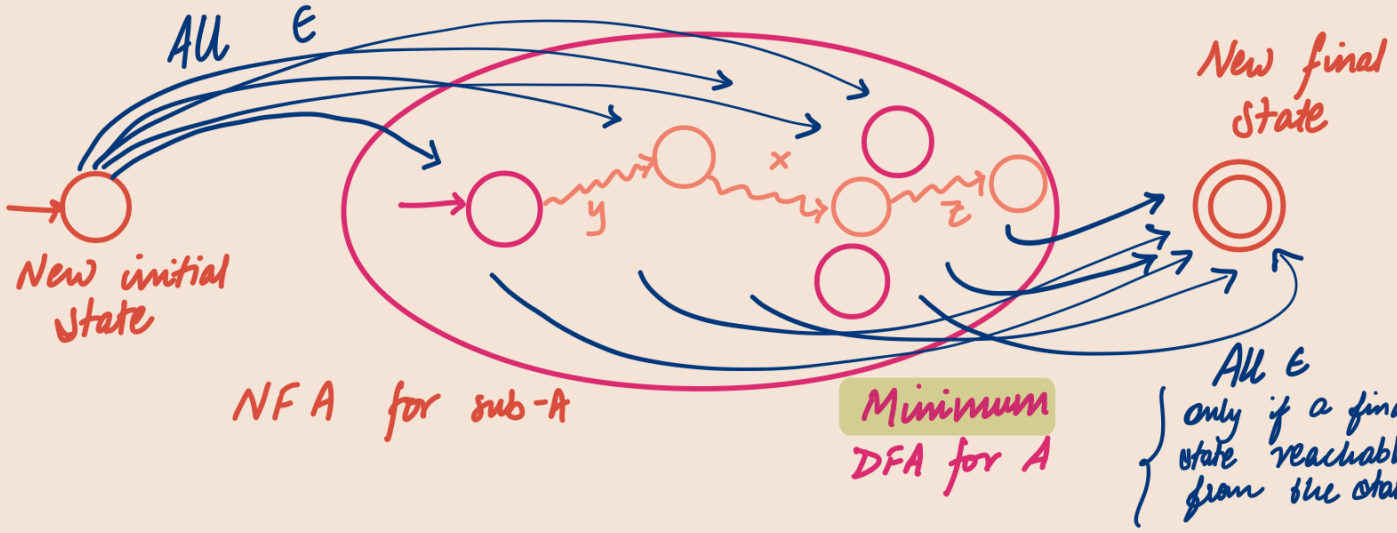
∴ F is a DFA with minimum number of states accepting A.

Def: Substring Let A be a regular set

Define $\text{sub-}A := \{ x \mid \exists\, y, z \in \Sigma^* \text{ s.t. } yxz \in A \}$

Set of All substrings of A

Lemma: A is regular $\Rightarrow$ sub-A is regular



→ yxz accepted by $DFA_A \Rightarrow x$ accepted by $NFA_{sub\text{-}A}$

start $\xrightarrow{y}$ s_1 $\xrightarrow{x}$ s_2 $\xrightarrow{z}$ Accept v/s start $\xrightarrow{\epsilon}$ s_1 $\xrightarrow{x}$ s_2 $\xrightarrow{\epsilon}$ Accept

→ x accepted by $NFA_{sub\text{-}A} \Rightarrow \exists yz$, yxz accepted by DFA_A

$s_0 \xrightarrow{\epsilon} s_1 \xrightarrow{x} s_2 \xrightarrow{\epsilon}$ Accept v/s

What if we $\nexists\, y$ / $\nexists\, z$?

$\exists\, y$ for $s_0 \to s_1$
$\exists\, z$ for $s_2 \to$ accept
$\Rightarrow yxz$ accepted

If $\nexists\, y$ then s_1 is superfluous & it can be deleted w/o effect (s_1 can exist in general DFA). But, we use minimum DFA $\Rightarrow \exists\, y$ to reach s_1

If $\nexists\, z$, then no extension of x is accepted but NFA makes $\xrightarrow{\epsilon}$ to acceptance NFA $\Rightarrow \exists\, z$ to take s_2 to acceptance in DFA

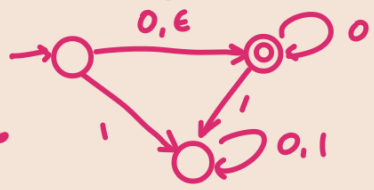
Hence, proved

- Transition from NFA to DFA

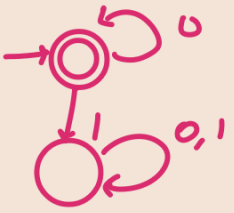
$n^0 = |Q|, Q \longrightarrow 2^n \cdot 2^Q$ ← Exponential blowup

Does not always happen

Ex:



NFA ⇒ DFA



Accepts 0^*

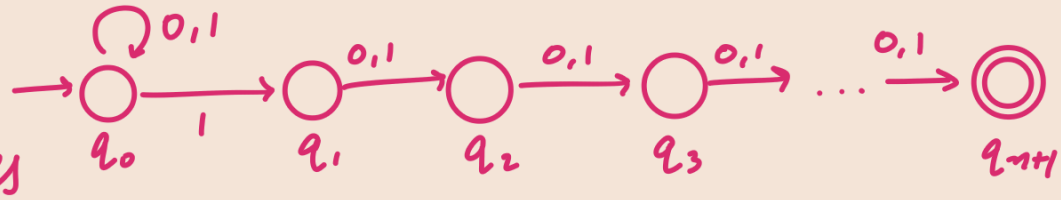
Do, is the exponential blowup essential? Yes

Lemma: There is a regular set with an accepting NFA having n states & any DFA that accepts the set has at least 2^{n-1} states

Pf: Let $A = \{x1y \mid |y| = n\}$, $\Sigma = \{0, 1\}$

↳ Last $(n+1)^{th}$ bit is 1

NFA:



↳ $n+2$ states

Consider A & eq. classes induced by A

$x \, R \, y$ iff $\forall \, z \quad xz \in A$ iff $yz \in A$

We intend to find a lower bound on # of eq. classes ⇒ lower bound on minimum states

Let $w \in \{0,1\}^*$, $|w| = n+1$

$E_w = \{xw \mid x \in \{0,1\}^*\}$ ← No. of such E_w is 2^{n+1}

▷ Each E_w is contained in an eq. class induced by A { membership defined only by $n+1$ last bits }

Let $xw, yw \in E_w$

For any z, $x\underbrace{wz}_{\geq n+1 \text{ bits}} \in A$ iff $wz \in A$ iff $ywz \in A$

▷ Diff E_w's are contained in diff. eq. classes

Consider E_w & $E_{w'}$ for $w \neq w'$

Let $w \in E_w$ & $w' \in E_{w'}$

Suppose w, w' differ on k^{th} ($\leq n+1$) bit from right

Consider $w0^{n+1-k}$ & $w'0^{n+1-k}$

WLOG, assume $\underbrace{w_k = 1}_{k^{th} \text{ bit from right in } w}$ & $w'_k = 0$

$\Rightarrow w0^{n+1-k} \in A$ & $w'0^{n+1-k} \in A$

$\Rightarrow w$ & w' belong in diff. eq. classes

$\therefore$ Each E_w $\in$ unique eq. class &

$\Rightarrow 2^{n+1} = \#\ E_w \leq \#$ eq. class

$\Rightarrow$ Size of DFA $\geq 2^{n+1} = 2^{(n+2)-1}$

$\Rightarrow$ Exponential blow-up.

☆ What are non-regular sets?
How to know if a set is not regular?

Ex: $A = \{0^n 1^n \mid n > 0\}$ is not a regular set

Pf: Eq. classes induced by A on $\{0,1\}^*$

For x, y in an eq. class,
$xz \in A$ iff $yz \in A$

Consider $\{0^m \mid m > 0\}$

$0^{m_1} 1^{m_1} \in A$ but $0^{m_2} 1^{m_1} \notin A$

$\Rightarrow$ Each element of $\{0^m \mid m > 0\}$ belongs to a diff eq. class

$\Rightarrow \infty$ elements $\Rightarrow \infty$ eq. class

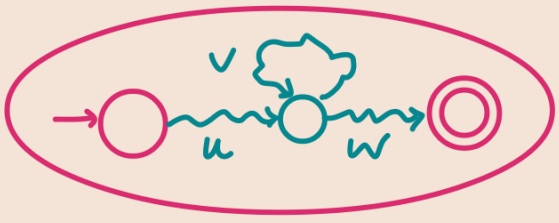
Pumping Lemma: Suppose A is regular. Then $\exists\ n > 0$ s.t. $x \in \Sigma^*$, $|x| > n$, $\exists\ u, v, w$ s.t. $x = uvw$, $|uv| \leq n$ & $|v| \geq 1$, if $uvw \in A$

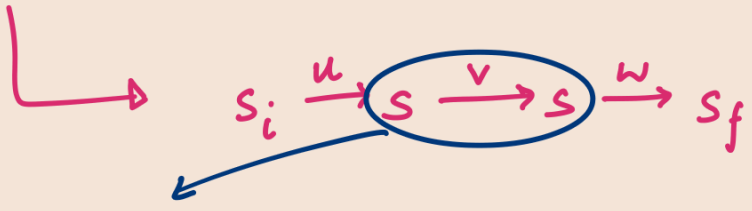
Pf:

DFA for A

If A is finite take n to be the max. length & let $|t| = n$, $t = uv$ then $\nexists\ w$ s.t. $uvw \in A$ So, lemma is vacuously true

If A is infinite, consider $x \in A$ & $|x| > n$, n is the no. of states in the DFA. Now, $|x| > n \Rightarrow$ some state (s) Pick the first of these ⟵ reached twice





Can pick this loop any number of times & end up at $s_f \in F$
$\Rightarrow uv^i w \in A \quad \forall \quad i \geq 0$

A regular $\Rightarrow$ { $(\exists\ n > 0)\ (\forall\ x, |x| > n)$
$(\exists\ u, v, w \quad x = uvw, |uv| \leq n, |v| \geq 1)$
$uvw \in A \Rightarrow \forall\ i \geq 0\ uv^i w \in A$ }

(shown above) ↙

Contrapositive ↓

{ $(\forall\ n > 0)\ (\exists\ x, |x| > n)$
$(\forall\ u, v, w, x = uvw, |uv| \leq n, |v| \geq 1)$
$(uvw \in A \ \&\ \exists\ i \geq 0\ uv^i w \notin A)$ } $\Rightarrow$ A is not regular

Ex: $A = \{0^m 1^m \mid m \geq 1\}$. Is A regular?
($\forall$) Adversary picks n, ($\exists$) we pick x, $x = 0^n 1^n \in A$. ($\forall$) Adversary splits $x = uvw$ $|uv| \leq n$, $|v| \geq 1$ ⟶ $x = 0^{j+k+l} 1^n$

$u = 0^j$, $v = 0^k$, $w = 0^\ell 1^n$

($\exists$) We get to pick i ($i = 0$)

then $uw = 0^{j+\ell} 1^n \notin A$ $\because$ $j + \ell < n$

By above lemma, A is not regular.

Ex: $A = \{3^m \mid m \geq 0\}$ over $\{0, 1\}$

($\forall$) Adversary chooses $n > 0$

($\exists$) We choose $x = 3^n \in A$ ($|x| = n \log_2 3 > n$)

($\forall$) Adversary chooses

$x = uvw$, $|uv| \leq n$, $|v| \geq 1$

N_u ← No. represented by u in binary; N_v; N_w

$|v| = \ell$, $|w| = k$

$x = N_u 2^{\ell+k} + N_v 2^k + N_w$

Pumping v i times gives number

$uv^i w = N_u 2^{k+i\ell} + N_v 2^{k+(i-1)\ell} + N_v 2^{k+(i-2)\ell} + \ldots + N_v \cdot 2^k + N_w$

Assume $uv^i w \in A \Rightarrow uv^i w = 3^{n_i}$

Consider $uv^{i+1} w - uv^i w$

$= N_u 2^{k+(i+1)\ell} + N_v \cdot 2^{k+i\ell} - N_u \cdot 2^{k+i\ell}$

$= 2^{k+i\ell} (N_u \cdot 2^\ell + N_v - N_u)$

$= 3^{n_{i+1}} - 3^{n_i}$

$\Rightarrow 3^{n_i} (3^{n_{i+1} - n_i} - 1) = 2^{k+i\ell} (N_u \cdot 2^\ell - N_u + N_v)$

$i \uparrow$ $n_i \uparrow$ $3^{n_i} \uparrow$ ↪ more powers of 3

$i \uparrow$ $N_u \cdot 2^\ell - N_u + N_v \leftrightarrow$ ↪ $O(1)$ powers of 3

$\Rightarrow$ For large enough i, 3^{n_i} does not divide $N_u \cdot 2^\ell - N_u + N_v$

$\Rightarrow$ A is not regular

- Previously, we only used CPU and I/O

We now introduce memory in a restricted way in the form of a **Stack** (also called **Pushdown Automata**)

- A pushdown automata is a 7-tuple $(Q, \Sigma, \Gamma, s_0, \perp, \delta, F)$ where

Q :	Finite set of states
Σ :	Input alphabet
Γ :	Stack alphabet
s_0 :	Starting state
$\perp$:	Bottom of stack (denotes)
δ :	$Q \times \Sigma \cup \{\epsilon\} \times \Gamma \longrightarrow Q \times \Gamma^k$
F :	$F \subseteq Q$, set of final states

Acceptance can be of two types:

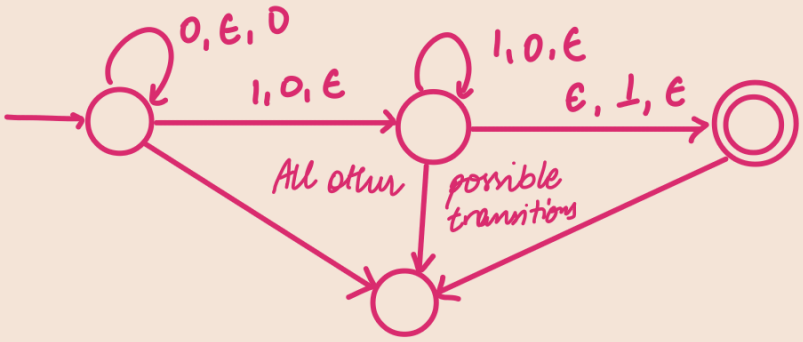
- **Final state** after the input is read
- **Empty stack** after the input is read

δ : (State, (Input, Read, Write)) $\longrightarrow$ State

$\perp$ is a special character in the stack denoting its bottom

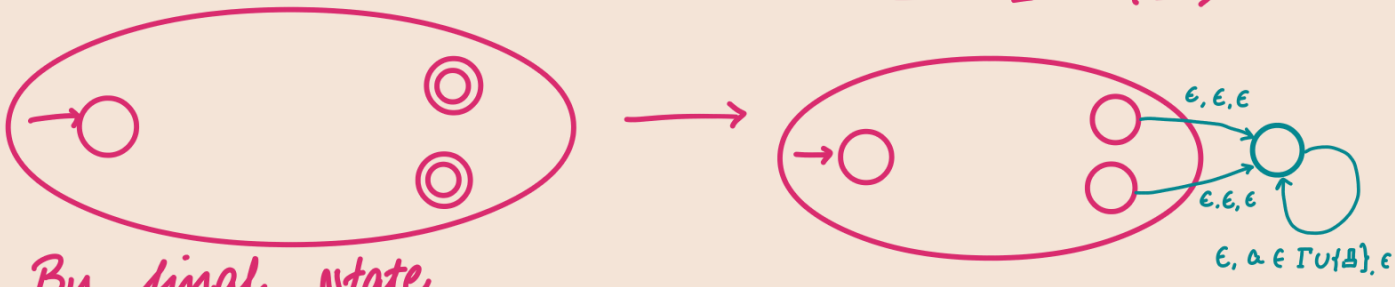
Ex: $A = \{ 0^n 1^n \mid n \geq 1 \}$

OR



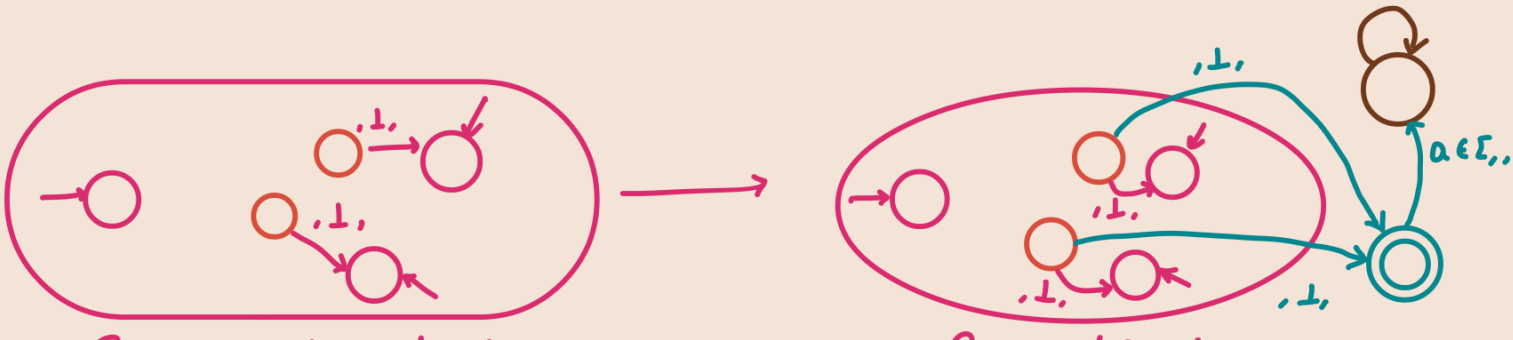
Equivalence of acceptance by final state & empty stack

$\Gamma' = \Gamma \cup \{\text{卌}\}$



By final state → By empty stack

Due to 卌 in Γ', the stack never gets empty on the right { stack might've gotten empty on the left at a non-final state}

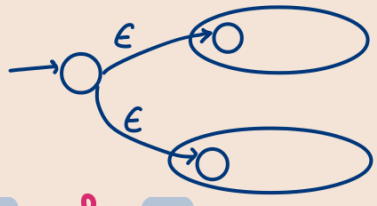


By empty stack (and input exhausted) → By final state

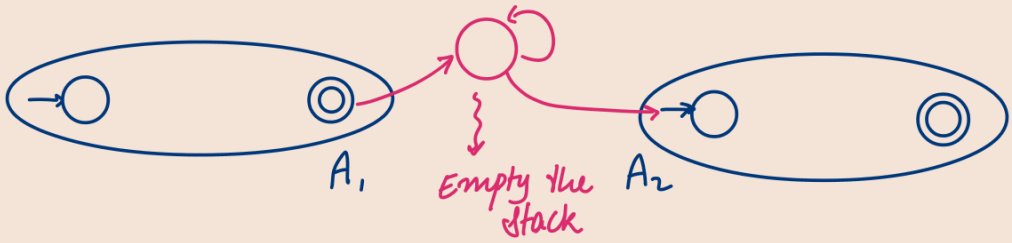
Properties of Sets accepted by PDAs

→ Push Down Automata

* Closure under **union**



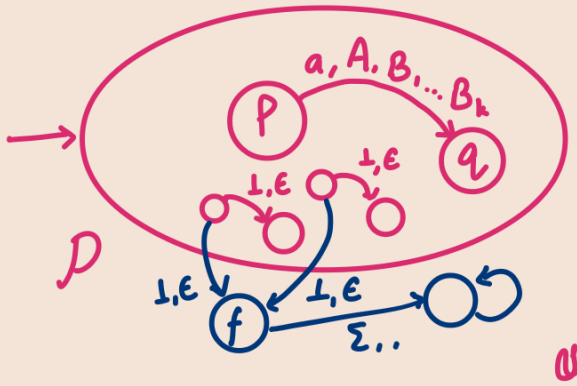
* Closure under **concatenation** & *



· **Not** closed under **complement** & **intersection**

Theorem: Any PDA can be simulated by a PDA with one state

Proof: Given $(Q, \Sigma, \Gamma, s_0, \perp, \delta, F)$
Assume that PDA accepts by an empty stack so $F = \phi$



· Define $\Gamma' = Q \times \Gamma \times Q$
(q_1, A, q_2) means P moves from q_1 on input A to state q_2 (after a series of transitions) and stack no longer has A at the top (rest of the stack remains the same)

{ It might be the case that A is at the top of the stack multiple times  }

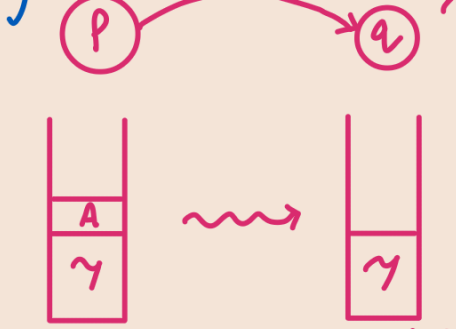
· Define a new PDA as:

$(\{r\}, \Sigma, \Gamma', r, \perp', \delta', \phi)$ → Accepts only on empty stack

$\perp' := (s_0, \perp, f)$

δ':

→ If input is accepted then ∃ a moment where



at state p

at state $q' = q_k$

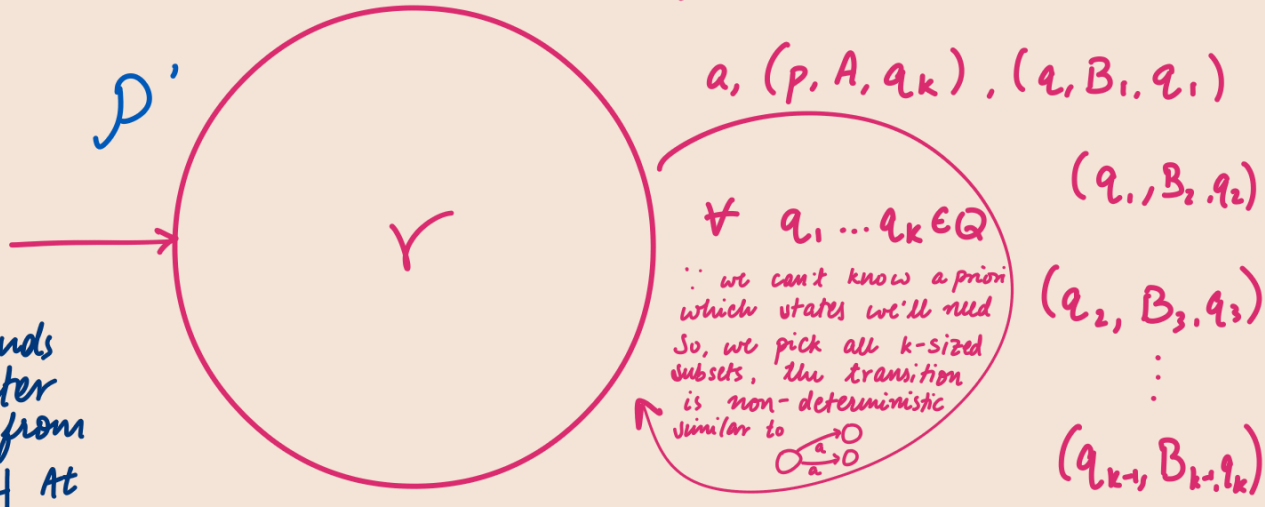
stack has "decreased" from (@ p) to (@ q')
Assume B_i pushed last onto the stack

B_1 is popped, B_2 is popped, B_k is popped

$p \rightsquigarrow q_1 \rightsquigarrow q_2 \rightsquigarrow \dots \rightsquigarrow q_k \rightsquigarrow q'$

This transition is captured as such:

(l, C, m)
l: Current state
C: Character on top of stack
m: State P lands on m after stack goes from



$$[p, x, c_1 \dots c_m] \xrightarrow[P]{n} [q, \epsilon, \epsilon]$$

PDA P, in state p, on reading input x with stack $c_1 \dots c_m$, in n steps moves to state q with no input left and an empty stack.

Thm: P accepts a string s iff P' accepts string s.

Pf: By induction on the number of steps n to show the following:

for any $p \in Q$, $x \in \Sigma^*$, $c_1 \dots c_m \in \Gamma^*$

$[p, x, c_1 \dots c_m] \xrightarrow{n}$

$[p, x, c_1 \dots c_m] \xrightarrow[p]{} [q, \epsilon, \epsilon]$

$\Rightarrow \exists \; q_1 \dots q_{m-1} \in Q : [r, x, \langle p, c_1, q_1 \rangle \langle q_1, c_2, q_2 \rangle \dots \langle q_{m-1}, c_m, q \rangle] \xrightarrow[p']{n} [r, \epsilon, \epsilon]$

If above is true, then

$[q_0, x, \perp] \xrightarrow[p]{n} [f, \epsilon, \epsilon]$ iff

$[r, x, \langle q_0, \perp, f \rangle] \xrightarrow[p']{n} [r, \epsilon, \epsilon]$ for some $n > 0$

Base Case: $n = 1 \longrightarrow$ Condition reduces to

$[p, a, \perp] \xrightarrow[p]{1} [f, \epsilon, \epsilon]$ iff $(a \in \Sigma \cup \{\epsilon\})$

$[r, a, \langle p, \perp, f \rangle] \xrightarrow[p']{1} [r, \epsilon, \epsilon]$

Induction Step: Assume for n. Need to show:

$[p, x, c_1 \dots c_m] \xrightarrow[p]{n+1} [f, \epsilon, \epsilon]$ iff

$[r, x, \langle p, c_1, q_1 \rangle \dots \langle q_{m-1}, c_m, q_m \rangle] \xrightarrow[p']{n+1} [r, \epsilon, \epsilon]$

Suppose $[p, x, c_1 \dots c_m] \xrightarrow[p]{n+1} [f, \epsilon, \epsilon]$

This implies $\exists \; a \in \Sigma \cup \{\epsilon\}$, state $q \in Q$, and $B_1, \dots, B_k \in \Gamma$ such that:

(i) $x = ay$

(ii) $[p, ay, c_1 \dots c_m] \xrightarrow[p]{1} [q, y, B_1 \dots B_k c_1 \dots c_m]$

(iii) $[q, y, B_1 \dots B_k c_2 \dots c_m] \xrightarrow[p]{n} [f, \epsilon, \epsilon]$

Context Free Grammars

- Specified by a finite set of terminals, a finite set of non-terminals & a finite set of productions connecting terminals

and non-terminals.

Terminals = $\Sigma \cup \{\epsilon\}$ ⟶ alphabet
↳ represented by small letters

Non-terminals ⟶ intermediate symbols representing a set of strings over terminals
↳ represented as capital letters

Productions ⟶ tells us how each non terminal is translated to a set of terminals

$A \to \gamma, \quad \gamma \in (\text{Terminals} + \text{Non-terminals})^*$

Ex: Consider $\{0^n 1^n \mid n \geq 1\}$

$S \to 0S1, \quad S \to 01$

$S \to 0S1 \to 00S11 \to 000S111 \to 00001111 = 0^4 1^4$

Ex: Arithmetic expressions using +-*/ over integers

$S \to S+S$; $S \to S*S$; $S \to S-S$; $S \to S/S$;
$S \to (S)$; $S \to I$; $S \to -I$; $S \to -S$;
$I \to 0I$; $I \to 1I$; $I \to 2I$; ...; $I \to 9I$;
$I \to 0$; $I \to 1$; $I \to 2$; ...; $I \to 9$;

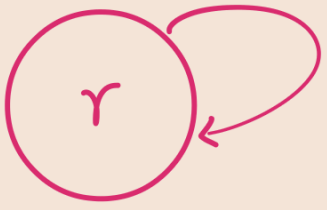
Def: The set of strings of terminals generated by a context-free grammar (CFG) is called a context-free language (CFL).

∴ Context-free : Production rules for a non-terminal are free of its context (left and right neighbours).
Counterpart is ∴ context-sensitive ∴

Theorem: A set of Σ^* is a CFL iff it is accepted by a PDA

Pf: Let $G = (\Sigma, \Gamma, P)$ be a CFG
(Σ: terminals, Γ: non-terminals, P: productions)

- For each $a \in \Sigma$ introduce a new non-terminal N_a in Γ & new production $N_a \to a$ in P
- Convert every production $A \to \gamma$ of P by replacing any terminal in γ except the one that is the 1st letter of γ by N_a corresponding to that a
- PDA



$a, A, B, B_2 \dots B_k$

$P: A \to a B_1 \dots B_k$
$(a \in \Sigma \cup \{\epsilon\})$

Pumping Lemma for CFLs

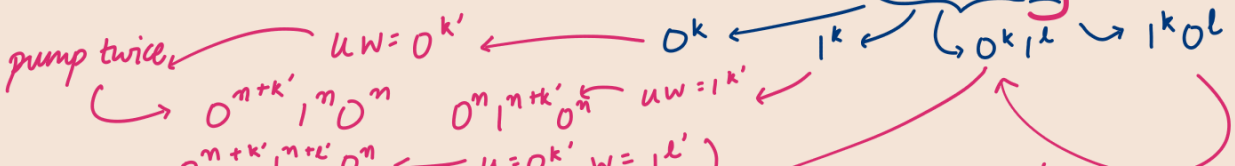
Let A be a CFL. $\exists\ n > 0$ s.t. for every $xuvwy \in \Sigma^*$ of length $> n$, the following holds:

(i) $|uvw| \leq n$ (ii) $|uw| \geq 1$

(iii) If $xuvwy \in A$ then $xu^i v w^i y \in A\ \forall\ i$

Ex: $\{0^m 1^m 0^m \mid m > 0\}$ is not a CFL

Consider $0^n 1^n 0^n = xuvwy$



$0^n 1^{l'} 0^k 1^{l+l^2} 0^n \leftarrow u = 0^k 1^{l'},\ w = 1^{l^2}$

$0^{n+k'} 1^l 0^{k^2} 1^n 0^n \leftarrow u = 0^{k'},\ w = 0^{k^2} 1^l$

Similar to this case

can use $i = 0 \xrightarrow{i=2} 0^{n-k'} 1^{n-l'} 0^n,\ 0^{n-k} 1^{n-l'-l^2} 0^n,\ 0^{n-k'-k^2} 1^{n-l} 0^n$

Ex: $\{ ww \mid w \in \{0,1\}^* \}$

Consider $0^n 1^{2n} 0^n 1^{2n}$

$= uvw$

- first half → pump 0 times → first half → ...0, second half → ...1
  - $0^{n-k_1} 1^{2n-k_2} 0^n 1^{2n}$
  - length $= (n-k_1+2n-k_2+n+2n)/2 = 3n - \frac{k_1+k_2}{2} = 3n - k_1 - k_2 + \frac{k_1+k_2}{2}$
  - ⇒ first half ends w/ 0
- second half → similar to
- $1^k 0^l$ → pump 0 times → $0^n 1^{2n-k} 0^{n-l} 1^{2n}$

- Given a CFL A and a string x, how to determine if $x \in L$?
  Consider $CFG = (\Sigma, \Gamma, P)$ generating A

Chomsky Normal Form (CNF)

A CFG (Σ, Γ, P) is in Chomsky Normal Form if every production in P has one of the following two forms:

$B \to a$ $\qquad$ $B \to CD$

- Converting a CFG to its (equivalent) CNF form:
  - For every $a \in \Sigma$, introduce a new production $T_a \to a$
  - For every production $B \to \gamma$ in P, replace terminals in γ be newly introduced terminals unless $\gamma \in \Sigma$
  - Productions now have the following forms:
    $B \to a$, $B \to C_1 C_2 \dots C_k$, $k > 1$
    $B \to \epsilon$ $\quad$ $B \to C$ $\quad \to B \to C T_c$, $T_c \to \epsilon$

Replace by $B \to C_1 R_1$, $R_1 \to C_2 R_2$, $R_2 \to C_2 R_3$, ... $R_{k-1} \to C_{k-1} C_k$

→ Replace each occurrence of B on RHS, by enumerating all the possibilities:

Ex: $C \to \gamma_1 B \gamma_2 B \gamma_3$: $C \to \gamma_1 B \gamma_2 B \gamma_3 \mid \gamma_1 B \gamma_2 \gamma_3 \mid \gamma_1 \gamma_2 B \gamma_3 \mid \gamma_1 \gamma_2 \gamma_3$

- The conversion is complete

Idea: Given string $x \in \Sigma^*$, $x = a_1 a_2 a_3 \cdots a_n$, compute the set of non-terminals T_{ij} that generate string $x_{ij} = a_i a_{i+1} \cdots a_j \quad \forall \; i, j$

- Computing $T_{i,i}$: Pick only those where $T \to a_i$

  $\Rightarrow T_{i,i} = \{ C \mid C \to a_i \text{ is in } P(\text{productions}) \}$
- Computing $T_{i,i+1}$:

  $\Rightarrow T_{i,i+1} = \{ C \mid C \to T_1 T_2, \; T_1 \in T_{i,i}, \; T_2 \in T_{i+1,i+1} \}$
- Computing $T_{i,j}$: $\quad i - j \geq 1$

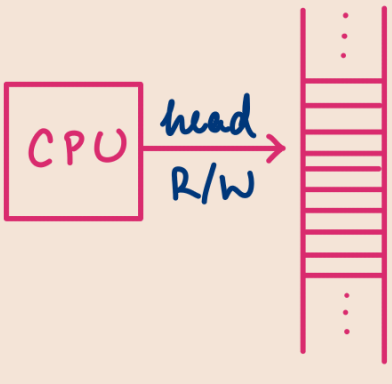
  $\Rightarrow T_{i,j} = \{ C \mid C \to T_1 T_2, \; T_1 \in T_{i,k}, \; T_2 \in T_{k+1,j}, \; k \in [i, j) \}$
- Above computations are possible due to CNF

- To check if $s \in A$, just check if $S \in T_{1,n}$.

  Time complexity:

  $O(|s|^3 \cdot |P| \cdot f(T)) \Rightarrow O(|s|^3)$

  ($|P|$: # of Productions; $f(T)$: Time to check membership in set; Independent of s)
- The above algorithm is the **Cocke - Younger - Kasami (CYK) Algorithm**
- Time complexity to check membership in a regular set: $O(|s|)$ → Run it on a DFA



Transition from one state to another

if input ended

Need both accept + reject states as final states

- At the beginning of the computation, input is written as a contiguous finite state of memory, and head pointing to the leftmost input cell.
  Plus, the state is start-state.
- Usually, PC (program counter) can move any number of places, but allowing head to move just one place at a time is sufficiently powerful.

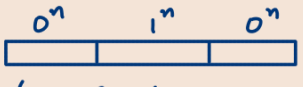
* This computation model is called a **Turing Machine**

Turing Machine : $(Q, q_0, \Sigma, B, \delta, F_{accept}, F_{reject})$ — 7-tuple

- Q: Set of states
- q_0: Start state
- Σ: Input Alphabet
- B: Special signal written in memory cells except input cells (Stands for Blank)
- δ: $Q \times \Sigma \cup \{B\} \to Q \times \Sigma \cup \{B\} \times \{L, R\}$
- F_{accept}: Set of Accepting states
- F_{reject}: Set of Rejecting states

- Memory is called tape in Turing machines.
  Although we are studying an infinite tape, the memory in practice is finite.

Ex: $\{0^n 1^n 0^n \mid n \geq 1\}$



↳ Read a 0, replace it with a blank, go to the right end, write a C_1^*, go to the the left end. Repeat.
→ Read a 1, replace it w/ B, go to right end, write C_1^*, go to left end. repeat
→ Read a 0, replace it w/ B, go to right end, pop a C_1^*, go to left end. repeat.
Need to take care, which end of the right "write" part we write to, so that other inputs are not wrongfully accepted.

$O(n^2)$ but no auxiliary space needed

Alternatively, ✓
→ Use two tapes/stacks.
→ One tape: read head, other tape is write head.
→ 0^n: fill w/ 0
  1^n: fill w/ 1
  0^n: fill w/ B

$O(n)$ but auxiliary space needed

- **Multitape TMs**: A TM with k tapes, each with its own head. Transitions done based on contents of all heads.
- **Multitrack TMs**: TMs with one tape that has multiple tracks, each with its own head.

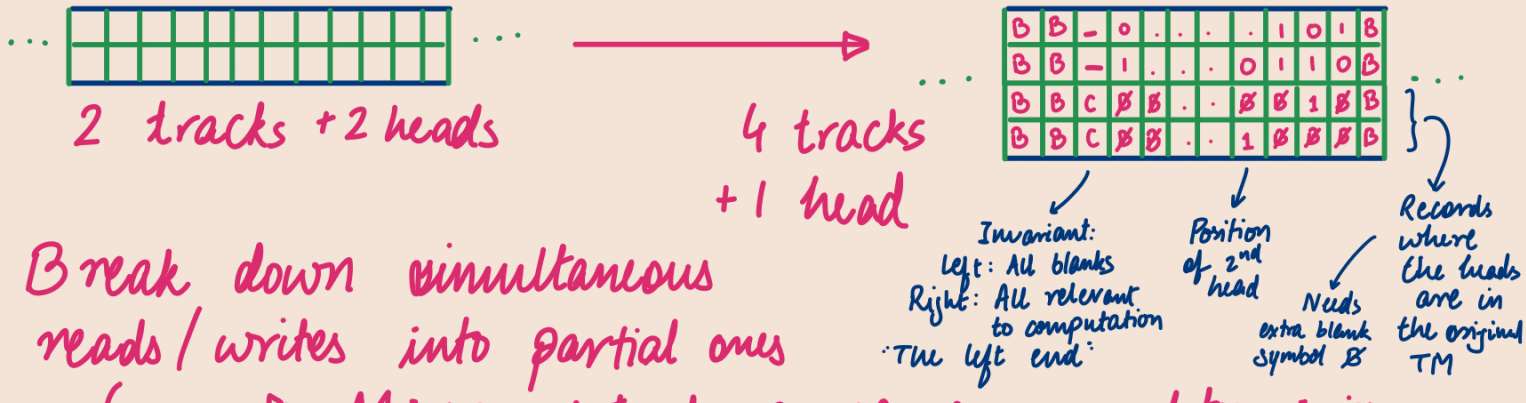
Theorem: Multitape TMs are equivalent to multitrack TMs

- **Multitrack TMs with one head**: One head reading contents of all tracks in a cell simultaneously

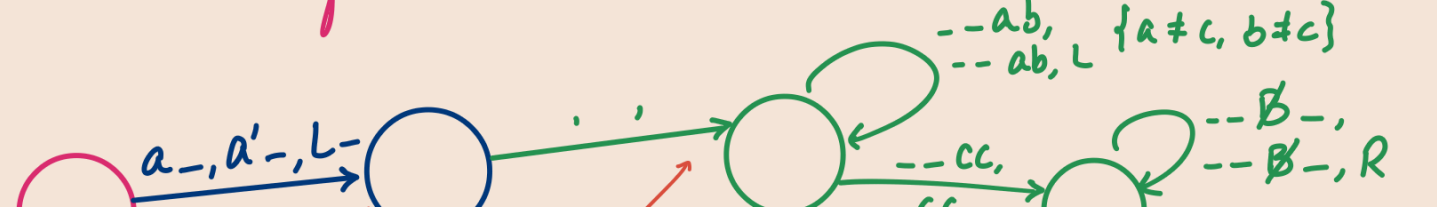
Theorem: Multitrack TMs are equivalent to Multitrack TMs with one head.

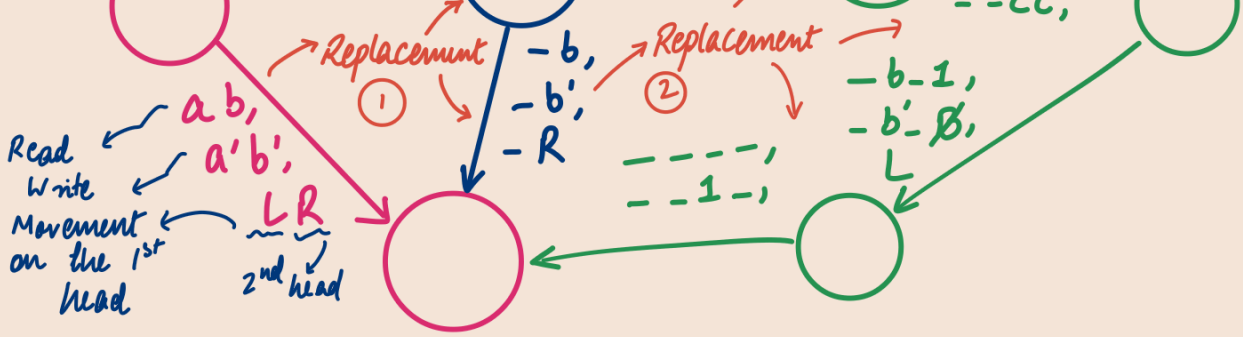
Proof:

Consider a 2-track TM with 2 heads

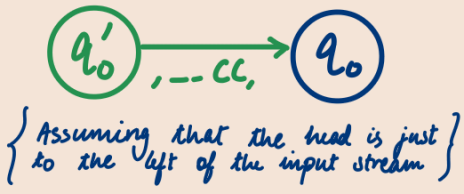


- Break down simultaneous reads/writes into partial ones
  - ↳ Might introduce some non-determinism
- For an operation doing an r/w in a single track, go to the left (identified by --CC); then keep going to the right until a "1" is encountered; do the r/w; update the track recording the head.
  - ↳ "No change"
- In terms of the state table





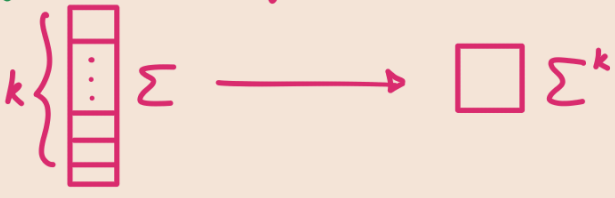
- Need to be careful when the original TM writes on the left end: Shift "--CC" to the left
- Also need to write "--CC" initially on the left end before entering the initial state



- This can be used to reduce any number of tracks:
  k heads → Choose any two → Combine them using above construction → k-1 heads

Theorem: Multitrack TMs with one head are equivalent to single track TMs.

Proof: Change the alphabet by combining current alphabets



Theorem: One track, one head TMs are equivalent to one track, one head TMs with $\Sigma = \{0, 1\}$

Proof: Suppose $|\Sigma| = k$

Let $l = \lceil \log_2 k \rceil$

Code each symbol as l-bit binary sequence

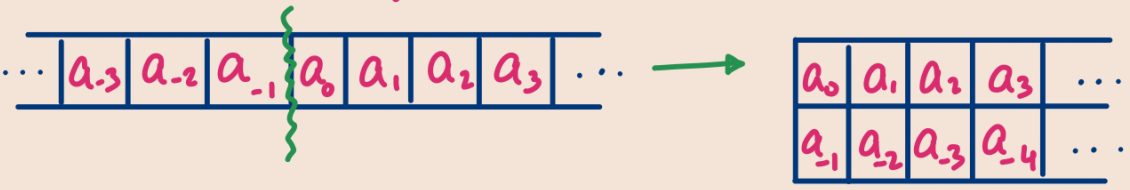
Theorem: TMs with $\Sigma = \{0, 1\}$ are equivalent to

TMs with $\Sigma = \{0\}$

Proof: Use β to view 0 as 0β, 1 as $\beta 0$, β as $\beta\beta$

Theorem: 2-way infinite TMs are equivalent to 1-way infinite TMs

Proof:

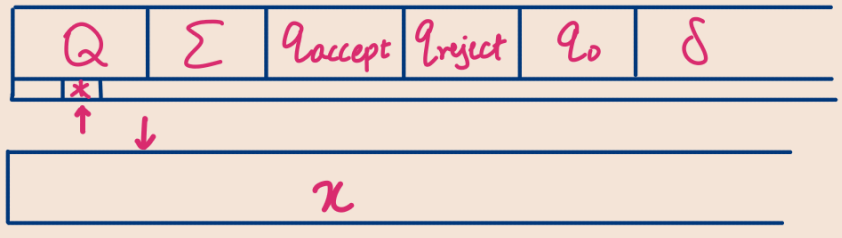


Church - Turing Thesis

Computability is **exactly** **captured** by **TMs**

Universal TMs

- Input is of the form (p, x) (Description of a TM M; Input to M)
- Simulates M on x and accepts iff M accepts x



* There exists a **Universal Turing Machine** (UTM)

It takes as input $\langle p, x \rangle$ (a number) and **accepts** iff TM described by **p** **accepts** **x**.

TM description $p \Rightarrow p$ is a string $\Rightarrow$ write p as a view p as a number $\Leftarrow$ binary string

$\therefore$ a TM can be viewed as a number

$\Rightarrow$ 1, 2, 3, 4, ... = $\mathbb{N}$ enumerates all TMs

- Adopt convention that a number that does not correspond to a TM represents a TM that

rejects all inputs

Theorem: Number of subsets of $\{0,1\}^*$ are uncountable

Proof: Consider a real number in the range $[0,1)$ and write it in binary notation and $[0,1)$ iso/homeomorphic to $\mathbb{R}$ {tan func.}

Ex: $0.01001100\ldots$ — represents 1, represents 2, 6 present, 5 present

The **Halting Problem**

$H = \{(p,x) \mid$ TM p halts on input x after finitely many steps $\}$

Theorem: There is **no** **TM** that **accepts** H

Proof: Suppose TM M accepts H — represented by q, represented by r

Define another TM $\bar{M}$ as:

On input (p,x), if M accepts $(p,(p,x))$, go in an infinite loop.

If M rejects $(p,(p,x))$, accept

- What does M do on input $\langle r, \langle r,x\rangle\rangle$?

M accepts $(r,(r,x))$ $\Rightarrow \bar{M}$ does not halt on (r,x) $\Rightarrow M$ rejects $(r,(r,x))$

M rejects $(r,(r,x))$ $\Rightarrow \bar{M}$ halts on (r,x) $\Rightarrow M$ accepts $(r,(r,x))$

$\therefore$ M does not exist

Def. Set A is **decidable** if $\exists$ a TM that accepts all strings in A and rejects all strings in $\Sigma^* \setminus A$

Otherwise, set A is **undecidable**

★ **Halting Set** H is **undecidable**

· Given p, is there an input on which TM p does not halt?

Define $\hat{H} := \{p \mid \text{TM } p \text{ halts on all inputs}\}$

Is $\hat{H}$ decidable? ND

→ Suppose $\hat{H}$ is decidable. Let q be the description of TM M_q that accepts $\hat{H}$.

M_s { M_r { On input (p, x), define a TM that works as follows:

On input y, ignores y, writes x on tape, and simulates M_p on input x.

Runs M_q on input r and accepts (p, x) iff M_q accepts r.

M_s accepts $\Leftrightarrow$ M_q accepts r $\Leftrightarrow$ M_r halts on all inputs (p, x) $\Leftrightarrow$ M_p halts on x ⇏

⇒ ⇐ H is decidable

∴ $\hat{H}$ not decidable.

Q. Given a TM p, does it accept any string?

Undecidable. Towards contradiction, assume

M_q: TM that solves the problem

M_s: On input $\langle p, x \rangle$, generates r, accepts $\langle p, x \rangle$ iff M_q accepts r.

M_r: On input y, erase y, run M_p on x and do as it does.

Now, if M_p accepts input x, then M_r accepts Σ^*, otherwise ϕ; Feed r to M_q, it can tell whether M_r accepts any string or not, but then we know if p halts on x or not ⇒ Halting problem solved

$\Rightarrow\Leftarrow$ ∴ The question is undecidable

Q. Given TM p, does it accept strings only in 0^*?
Undecidable. Suppose M_q does the task.
M_s: Given $\langle p, x\rangle$, generate r, accept iff M_q accepts r.
M_r: Given y, erase y, if M_p halts on x,
As before, M_r either accepts Σ^* or ϕ.
Now, feed r into M_q. If M_q rejects r, then M_s will accept $\langle p, x\rangle$, else reject
$\Rightarrow$ M_s can solve the halting problem.

(M_q rejects r ↙ ↘ M_q accepts r → M_p halts on x)

Q. Given a TM p, does it accept 0?
Undecidable. Same construction as before.

Q. Given a TM p, and a state q_i of M_p, is there an input x such that M_p never comes to q_i on x? Undecidable.
Suppose M_q solves the problem.
On input $\langle p, x\rangle$, construct r in a similar way. Feed r to M_q along with $(q_{accept} \xrightarrow{r} q_i)$.
Then, it is the same as asking if r accepts anything.
But, this was undecidable $\Rightarrow$ M_q does not exist.

Q. Given a TM p, state q_i of M_p, does there $\exists\, x$ s.t. M_p comes to state q_i on x?
Undecidable. Negation of prev. statement.

Q. Given a TM p and number n, is there an input x of M_p such that M_p runs on x

for more than n steps?

Decidable.

1) Run M_p on all inputs of length $< n$. If it takes more than n steps on any, accept.
2) Run M_p on all inputs of length $= n$. If it takes more than n steps or crosses the rightmost bit of input, accept.
3.) Reject otherwise. Because, the TM does not even consider the complete input. So, all inputs of length $> n$ can be replaced by their prefix (of length $= n$).

Q. Given a TM p and number n, decide if M_p has more than n states.

Decidable. Just look at the encoding of M_p.

* In all of the above [undecidable] problems, the "constructions" were actually reductions, where an instance of Halting problem ($\langle p, x \rangle$) was reduced to an instance of the given problem.

· Consider any property of computable sets.
Define it as:

P: Collection of all computable sets $\longrightarrow \{T, F\}$

* Property P is non-trivial if it does not assign same value to all computable sets.

Rice's Theorem: For any non-trivial property P, checking if it holds for a given TM M_p as input, is undecidable.

Pf: Suppose $P(\phi) = \text{false}$. Let $x \in L_0$ s.t. $P(x) = \text{true}$.

Pf: Suppose $P(\phi)$ = false. Let $X \subseteq \{0,1\}^*$ s.t. $P(X)$ is true. Suppose M_q decides property P. Otherwise, talk about $\bar{P}$. Define TM M_s as:

On input (p, x), construct description of a TM r that works as follows: → $L(M_r) = X$ or $L(M_r) = \phi$

On input y, run M_p on x. If it halts, run TM for X on y and accept iff $y \in X$

Run M_q on r and accept (p, x) iff M_q accepts r.

→ M_s feeds r into M_q → M_q's decision on r helps M_s solve the halting problem for $\langle p, x \rangle$.

A contradiction ⇒ M_q does not exist

Reduction: Given sets $A, B \subseteq \{0,1\}^*$, we say that **A reduces to B** if ∃ a function

$f: \{0,1\}^* \longrightarrow \{0,1\}^*$ such that:

1) f is computable

2) $x \in A$ iff $f(x) \in B$

Lemma: Suppose A reduces to B and A is undecidable. Then B is also undecidable.

- In the previous example, Halting problem was reduced to a property P { M_s reduced $\langle p, x \rangle$ to r/ produced r from $\langle p, x \rangle$ }

Q. Is there a TM that outputs itself?

Yes. Consider the following TMs:

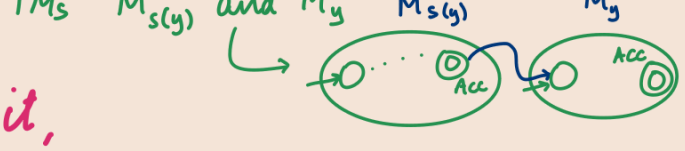
M_a := On input y, output $s(y)$ where,

$M_{s(y)}$:= On any input, outputs y.

M_b := On input y, outputs $s(y) \,||\, y$.

concatenation of ↙

Now, consider $M_{s(b)\|b}$:



- $M_{s(b)}$ receives input, erases it, writes b. Now, M_b receives b as input & outputs $s(b)\|b$. Thus, $M_{s(b)\|b}$ reproduces itself.
- These TMs are not used for sets, rather, they compute functions.

☆ Application of reduction:

Is FOL (first order logic) decidable?

{ The domain of some quantifiers might be infinite, in that case, give it as a TM as well }

No, it is not decidable because if it were, it would solve the halting problem ∵ the halting problem reduces as such:

∵ M_p halts on x iff $\exists\ t$ $H(p, x, t)$:

where $H(p, x, t) :=$ TM M_p halts on input x within t steps

Some important Results

- **Turing Recognizable**: Set L is called so if $\exists$ TM M s.t. M accepts (and thus halts) $\forall\ x \in L$. For $x \notin L$, it may not halt, or it may reject.
- **Decidable**: Set L is called decidable if $\exists$ a halting TM M that accepts it.
- **Halting TM**: TM that halts on all inputs (accepts or rejects each input).
- **Undecidable**: As expected.
- **Thm**: $\exists$ a TM unrecognizable language.
  $L = \{ s_i \mid M_i \text{ does not accept } s_i \}$
- **Thm**: $\exists$ an undecidable language.

$A_{TM} = \{\langle M, w\rangle \mid M \text{ accepts } w\}$

Pf: Suppose H decides A_{TM}. Construct N as follows: Input is a TM $\langle M\rangle$. Simulate H on $\langle M, \langle M\rangle\rangle$. If H rejects, accept. If H accepts, reject.

H is a halting TM $\Rightarrow$ N is a halting TM.

Feed $\langle N\rangle$ to N:

N accepts $\langle N\rangle \Leftrightarrow$ H rejects $\langle N, \langle N\rangle\rangle \Leftrightarrow$ N doesn't accept $\langle N\rangle$

- Thm: Halting Problem (set) is undecidable.

$H_{TM} = \{\langle M, w\rangle \mid M \text{ halts on } w\}$

Pf: H accepts H_{TM}. Construct H':

Input $\langle M, \langle w\rangle\rangle$: Simulate H on $\langle M, \langle w\rangle\rangle$. If H rejects, reject. Else, run M on $\langle w\rangle$ and do as M does. Observe how H' is a halting TM.

$\therefore$ H' decides A_{TM}. $\Rightarrow\Leftarrow$

- L is TR & $\Leftrightarrow$ L (& $\bar{L}$) is decidable

$\bar{L}$ is TR

$\Rightarrow$ Run M_L & $M_{\bar{L}}$ concurrently.

- Corollary: L is undecidable & TR $\Rightarrow$ $\bar{L}$ is non-TR
- co-TR $:= \{L \mid \bar{L} \text{ is TR}\}$
- $L_1 \leq_m L_2$: L_1 reduces to L_2 under computable function f such that: $x \in L_1 \Leftrightarrow f(x) \in L_2$
- ★ $L_1 \leq_m L_2 \Rightarrow$ 1) L_2 decidable $\Rightarrow$ L_1 decidable
  2) L_1 undecidable $\Rightarrow$ L_2 undecidable
  3) L_1 non-TR $\Rightarrow$ L_2 non-TR 4) L_2 TR $\Rightarrow$ L_1 TR
- $\overline{A_{TM}}$, $\overline{H_{TM}}$ are non-TR
- $\overline{A_{TM}} \leq_m E_{TM}$: $E_{TM} = \{\langle M\rangle \mid M \text{ is a TM & } L(M) = \phi\}$

construct f taking $\langle M, w\rangle$ & producing M' s.t. M does not accept $w \Leftrightarrow L(M') = \phi$

M': Input w. $w \neq x \Rightarrow$ reject. Else simulate M on w & do as it does.

$\Rightarrow L(M') = \{w\}$ or ϕ

- $E_{TM} \leq_m EQ_{TM} = \{\langle M_1, M_2\rangle \mid L(M_1) = L(M_2)\}$

f takes M as input, $M_1 \leftarrow M$ & M_2 rejects all inputs, ie, $L(M_2) = \phi$.

$L(M) = \phi \Leftrightarrow L(M_1) = \phi \Leftrightarrow L(M_1) = L(M_2)$

- $\overline{A_{TM}} \leq_m REG_{TM} = \{\langle M \rangle \mid L(M) \text{ is regular}\}$

f constructs M' from $\langle M, w \rangle$: M' simulates M on w & rejects if rejected. Otherwise, check if input $x = 0^n1^n$ & accept, else reject.

M accepts $w \Rightarrow L(M') = \{0^n1^n\} \Rightarrow$ Not regular

Else $L(M') = \phi$ which is regular

- $EQ_{TM} \leq_m CON_{TM} = \{\langle M_1, M_2, M_3 \rangle \mid L(M_1) = L(M_2) \cdot L(M_3)\}$

f takes $\langle M_1, M_2 \rangle$, M_3 only accepts ϵ.

- E_{TM}, EQ_{TM}, REG_{TM}, CON_{TM} are all non-TR.
- $L_1 \leq_m L_2 \Rightarrow \overline{L_1} \leq_m \overline{L_2}$

Time-bounded TMs

What can be computed in reasonable time?

Polynomial time ← What is this?

Polynomial-time computations

A TM that on input of size n, runs for at most $c \cdot n^k$ steps, for constants c & k.

$c \cdot n^k \rightarrow O(n^k)$

* Any non-trivial computation will take time $\geq n$ (simply to read input)

- Let **P** be set of all sets that can be solved in polynomial time. Let **FP** be the set of all functions that can be computed in polynomial time.
- Ex: {Sorting, Matrix Multiplication, Shortest Path, Max. Flow} $\subseteq$ FP

Primality Testing $\in$ P. Integer Factoring?

Problems not in P

$D = \{(i, x) \mid \text{TM } M_i \text{ rejects } (i,x) \text{ w/ in } |x|^i \text{ steps}\}$

- D is a computable set but $D \notin P$.
  Suppose $D \in P \Rightarrow \exists$ TM M_j accepting D & runs for at most $c \cdot n^k$ steps.
- $(j,x) \in D \Leftrightarrow M_j$ rejects (j,x) w/ in $|x|^j$ steps
  $\Leftrightarrow M_j$ rejects (j,x) w/ in $c|x|^k$ steps
  $\Leftrightarrow (j,x) \notin D$ for $|x| \geq c$
- $j > k$ needed. If $j \leq k$ then add superfluous states to M_j to pump j up.

Examples of problems not known to be in P

1) Satisfiability
   $F = C_1 \wedge C_2 \ldots \wedge C_m$, $SAT = \{F \mid \exists x_1 \exists x_2 \ldots \exists x_n \; F\}$
   ↳ CNF
2) Hamiltonian Cycle
   $HAM = \{G \mid G \text{ has a Hamiltonian Cycle}\}$
3) Traveling Salesman Problem
   (Shortest path version)
4) Sudoku ($n^2 \times n^2$)
   $SUD = \{\text{puzzles that have a solution}\}$
5) Vertex cover
   $VC = \{(G,k) \mid G \text{ has a vertex cover of size} \leq k\}$

The class NP

Set $A \in NP$ if $\exists$ a ND poly. time TM accepting it.

Thm: SAT, HAM, SUD, VC are in NP.

☆ We know that $P \subseteq NP$
It is conjectured that $P \neq NP$ — $P \neq NP$ Hypothesis

Def. Let A is said to be NP-Hard if $\forall B \in NP$, $\exists$ a poly-time computable func. f s.t.

$$\forall x: \quad x \in B \text{ iff } f(x) \in A.$$

i.e., $B \leq_m A$

Thm. Assuming $P \neq NP$, if A is NP-Hard, then $A \notin P$

Pf: If $A \in P$, then $\forall B \in NP$ $\exists$ poly time reduction f s.t. $x \in B \longrightarrow f(x) \in A$ can be solved in poly time $|f(x)| \leq p(|x|)$. Let Algo. for A run in $q(n)$ time, then we have a $q(p(n))$ time algo for arbitrary $B \in NP \Rightarrow B \in P \Rightarrow P = NP$

Thm. SAT is NP-Hard

Pf: Let M be a ND polytime TM

$M = (Q, \Sigma, q_0, \delta, F_{accept})$. Consider input x w/ $|x| = n$.

↳ Runs for $\leq p(n)$ steps. Define the following literals:

State (q, t): $q \in Q$ & $0 \leq t \leq p(n)$

TM is in state q at time t

Head (j, t): $1 \leq j \leq p(n)$, $0 \leq t \leq p(n)$

Head of M at time t points to j^{th} cell

Tape (s, j, t): $1 \leq j \leq p(n)$, $0 \leq t \leq p(n)$, $s \in \Sigma$

Value of j^{th} cell at time t is symbol s

For $t = 0$:

size $p(n)+2$: State $(q_0, 0) \wedge$ Head $(1, 0) \wedge$ Tape $(1, x[1], 0) \wedge$ ... Tape $(n, x[n], 0) \wedge$ Tape $(n+1, \text{B}, 0) \ldots \wedge$ Tape $(p(n), \text{B}, 0)$

For t:

$\forall q$: State $(q, t) \Rightarrow \neg$ State (q', t) $\forall q' \neq q$

$\forall j$: Head $(j, t) \Rightarrow \neg$ Head (j', t) $\forall j' \neq j$

$\forall j \; \forall s$: Tape $(j, s, t) \Rightarrow \neg$ Tape (j, s', t) $\forall s' \neq s$

size $= \underbrace{p(n)}_{t} * (\underbrace{|Q|^2}_{q,q'} + \underbrace{p^2(n)}_{j,j'} + \underbrace{p(n)}_{j} \cdot \underbrace{|\Sigma|^2}_{s,s'}) = O(p^3(n))$ { ∵ $|Q|, |\Sigma|$ independent

For $t > 0$:

$\forall j \; \forall q \; \forall s$

$[State(q,t) \wedge Head(j,t) \wedge Tape(j,s,t)]$

$\Rightarrow [State(q_1, t+1) \wedge Head(j_1, t+1) \wedge Tape(j_1, s_1, t+1)]$

$\vee [State(q_2, t+1) \wedge Head(j_2, t+1) \wedge Tape(j_2, s_2, t+1)]$

$\vee \ldots$

All transitions $\rightarrow$ size $\leq \underbrace{p^2(n)}_{t,j} \cdot \underbrace{|Q|}_{q} \cdot \underbrace{|\Sigma|}_{s} \cdot \underbrace{|Q|\,|\Sigma|}_{\delta:\text{Transition table}} = O(p^2(n))$

$\forall j \; \forall j' \neq j \; \forall s$: $Head(j,t) \Rightarrow [Tape(j', s, t+1) \Leftrightarrow Tape(j', s, t)]$

size $= O(p^3(n)) = \underbrace{t \cdot j \cdot j' \cdot s}$

$\bigvee_{q \in F} State(q, p(n)) \rightarrow$ size $= O(1) = |F|$

So, polynomial reduction from any $B \in NP$ to SAT

☆ This is the "Cook – Levin Theorem"

Thm. TM M accepts x, $|x| = n$, in $p(n)$ steps iff Formula F is satisfiable

☆ If SAT $\in P$ then $NP = P$

$\Rightarrow$ SAT is NP-Hard

Def: Set A is NP-complete if A is NP-Hard and $A \in NP$

Thm: 3SAT, Clique, Independent Set, Vertex Cover, Hamiltonian Path, Undirected Hamiltonian Path are all NP-complete

Pf: Checking membership in NP is fairly straightforward. Proving NP-Hard follows from the following reductions:

$SAT \leq_m 3SAT \leq_m IS \leq_m VC$, $3SAT \leq_m Clique$,

$SAT \leq_m VC \leq_m IS$, $3SAT \leq_m HP \leq_m UHP$