Theory of Computation

alphabet $\Sigma = \{a, b, c\}$ (a, b, c: letter) — finite size

$|\Sigma| = 3$

word or string on alphabet Σ $\quad w = abacab$

$w' = aacccbbaa$

Length of word $\quad |w| = 6 \qquad |w'| = 9$

ϵ as string of length 0

$w\epsilon = w = \epsilon w$

$w_1 \quad w_2$

Concatenation of w_1 and w_2

$w_1 w_2$

$\Sigma = \{a, b\}$

a — palindrom

ϵ — palindrom

aba — palindrom

abba — palindrom

ababa — palindrom

Prefix of a string w

$w = abcaab$

Prefix of w: ϵ, a, ab, abc ... abcaab

Proper prefix: remove ϵ, abcaab

Suffix of w: ϵ, b, ab, aab, --- abcaab

Proper Suffix of w: remove ϵ, abcaab

$\Sigma = \{a, b\}$

$\Sigma^1 = \{a, b\}$ $\Sigma^2 = \{aa, ab, ba, bb\}$

$\Sigma^i = \{$ all possible string of length i which can be formed from the letters a, b $\}$

$\Sigma^* = \bigcup_{i \geq 1} \Sigma^i$ can $\Sigma^0 = \epsilon$.

Countable Set

we get a one to one mapping from $|N|$.

Language is a set of strings of letters from some alphabet

Example: $\Sigma = \{a, b\}$ $L \subseteq \Sigma^*$

$\Sigma = \{a, b, c\}$ $L \subseteq \Sigma^*$

$S = \{$ infinitely many infinite length strings on some alphabet $\Sigma\}$

S countable?

S uncountable

$S \subseteq \Sigma^*$ countable

$f : S \longrightarrow \Sigma^*$ one to one

diagonalization S uncountable

0 1 1 0 0 — — — —

1 0 0 1 1 — — — —

⋮

0	1
1	2
00	3
01	4
10	⋮
11	
000	
001	
⋮	

$S \longrightarrow$ $|s|=i \longrightarrow j$

Proof Strategies:

- Induction,
- Direct
- Contrapositive
- Existense
  - Constructive
  - Non constructive
- Well ordering or Extremal principal
- Contradiction

Relation:

R on a set S.

$R \subseteq S \times S$ — Cartesian Product

reflexive	if $aRa \quad \forall a \in S$
transitive	$aRb, bRC \Rightarrow aRC$
Symmetric	$aRb \Rightarrow bRa$
Asymmetric	$aRb \Rightarrow bRa$ is false

Equivalence Relation :-

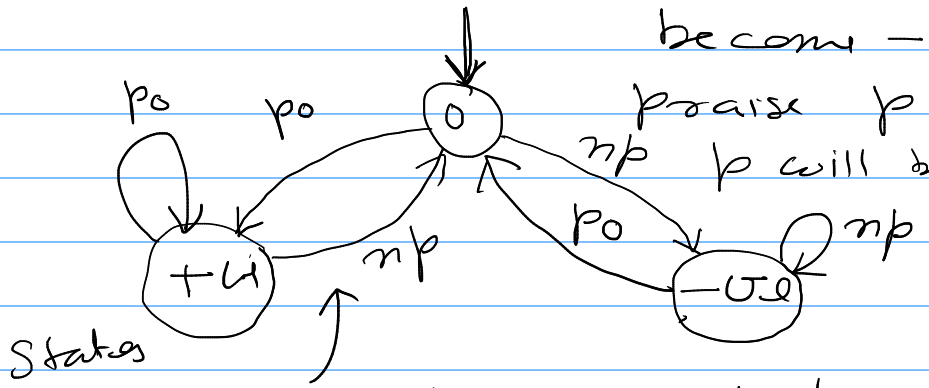
R is an equivalence relation if it is reflexive, transitive, and symmetric.

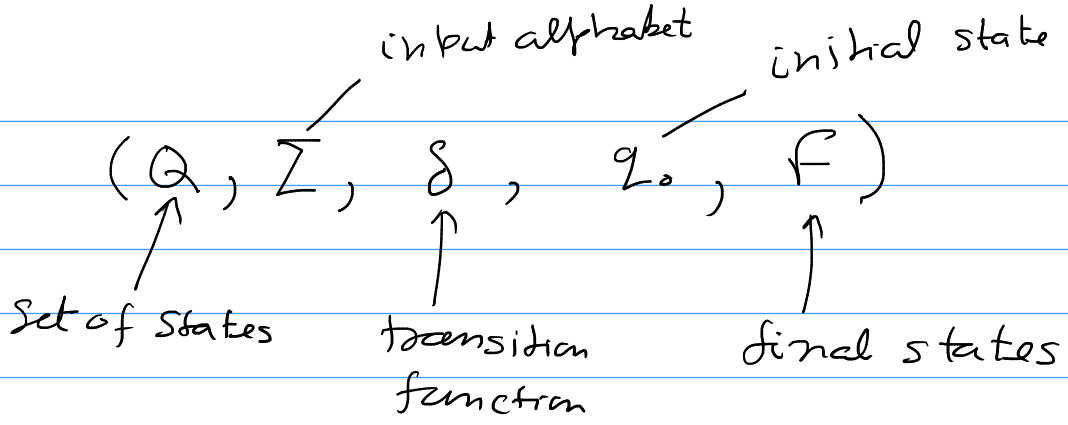
Thm R partition S into disjoint non empty equivalence classes.

Finite Automata

States
input symbols
transition from one state to another state

p —— $\overrightarrow{p_1}$ talks against p
then mood of p will become −q
praise p then p will become +q



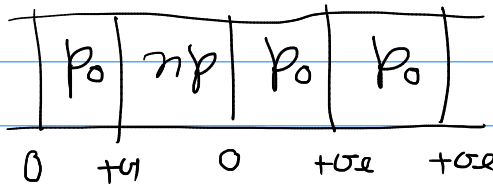


$$F \subseteq Q$$

$$\delta : Q \times \Sigma \longrightarrow Q$$

$q', q \in Q$, $a \in \Sigma$ $\quad \delta(q, a) = q'$

$$\delta(+ve, np) = 0$$



input string

We say that a string w accepted by the given automata when we reach final state after scanning the full string.

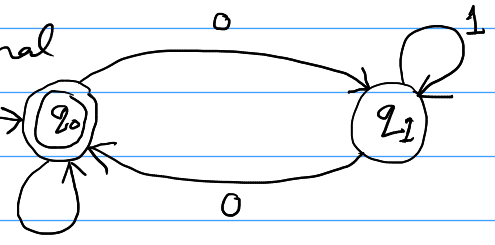
Design a finite automata which accept string on $\{0,1\}$ and each of the string contains even number of 0's.

Language

$$L = \{ x \in \{0,1\}^* \mid x \text{ has even number of zero's} \}$$

Number of 0's present in x mod 2

ϵ 0
ϵ 1
0001010 and final
1 start

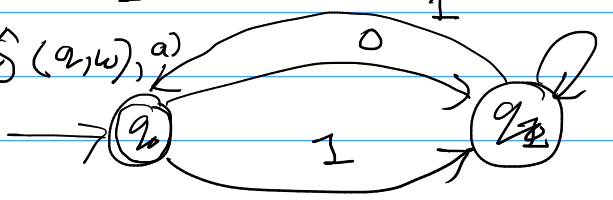


M

$\omega = 0001010$
$a = 0$

0001 rejected
$q_0\ q_1\ q_0\ q_1\ (q_1)$ Not final state

$\hat{\delta}(q_0, \omega a) = \delta(\hat{\delta}(q_0, \omega), a)$



$\hat{\delta}(q, \epsilon) = q$ (p = q)

$\hat{\delta}(q, \omega a) = \delta(\hat{\delta}(q, \omega), a)$

$\forall$ string ω and input letter a

$\hat{\delta}(q_0, 0001\,00) = \delta(\hat{\delta}(q_0, 0001\,0), 0)$

$= \delta(q_0, 0)$

$= q_1$

ω is accepted by FA, M if $\hat{\delta}(q_0, \omega) \in F$ (final state)
if not so then reject ω.

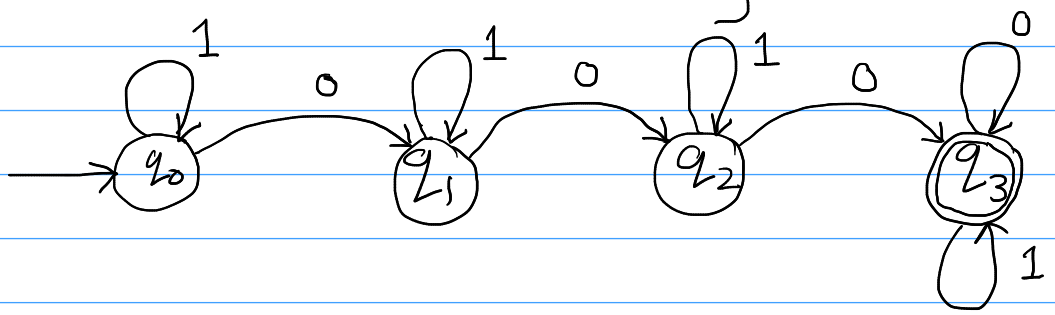
$L(M)$: Set of strings (all) accepted by FA M.

$L'(M) = \{00, 010\} = L(M)$ Not

$L(M) = L$ ($L(M) \subseteq L$, $L \subseteq L(M)$)
L is accepted by FA M

$\hat{\delta} : Q \times \Sigma^* \longrightarrow Q$.

$L = \{ x \in \{0,1\}^* \mid x$ contains at least 3 3ero's $\}$

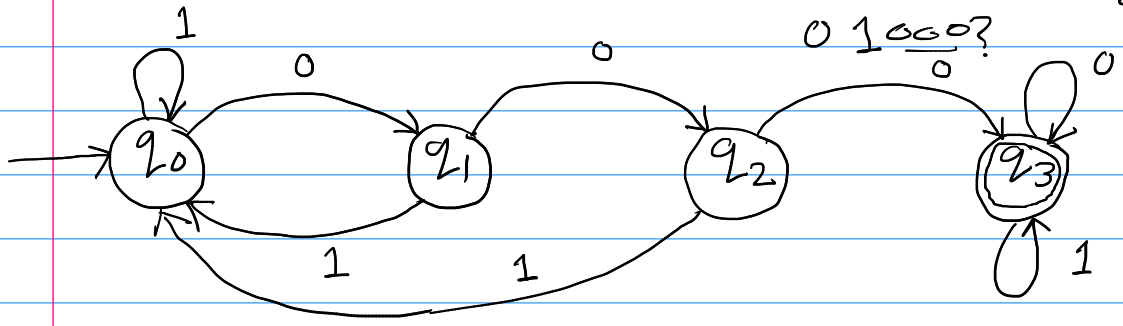


$L = \{ x \in \{0,1\}^* \mid x$ contains 000 as a factor $\}$

w contains factor y

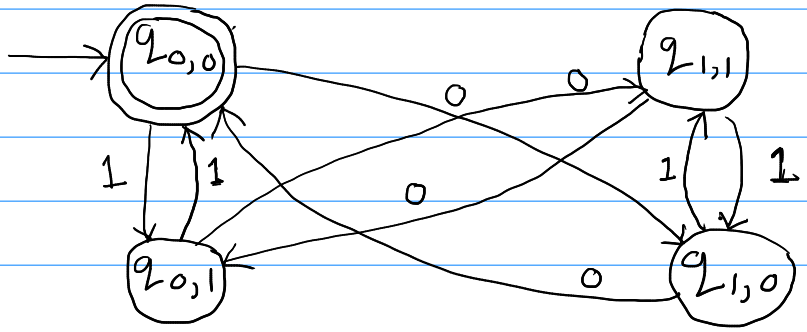
$w = uyv$ u, v might be empty

0 1 000?



$L = \{ x \in \{0,1\}^* \mid x$ contains even number of 3ro's and even number of 1's $\}$

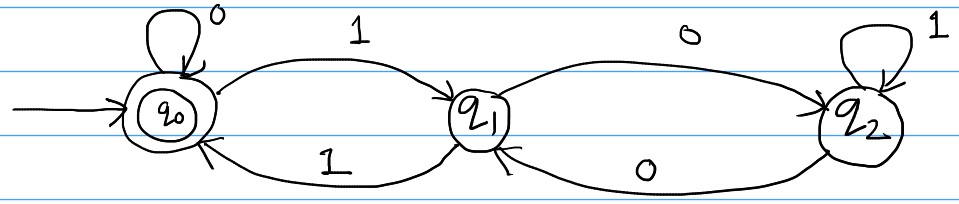
4 possibilities of parity of number of 0's and 1's



$d(10) = 2 \quad d(11) = 3 \quad d(0) = 0$

$$L = \{ x \in \{0,1\}^* \mid d(x) \bmod 3 = 0 \}$$

M



Prove that $L(M) = L$

$d(x) \bmod 3 = 0$

$x0 \quad d(x0) \bmod 3 = 0$

$d(x0) = 2d(x)$

$x1 \quad d(x1) = 2d(x) + 1$

$d(x1) \bmod 3 = 1$

$d(x) \bmod 3 = 1$

$d(x0) = 2d(x)$

$= 3 \times 2K + 2$

$d(x0) \bmod 3 = 2$

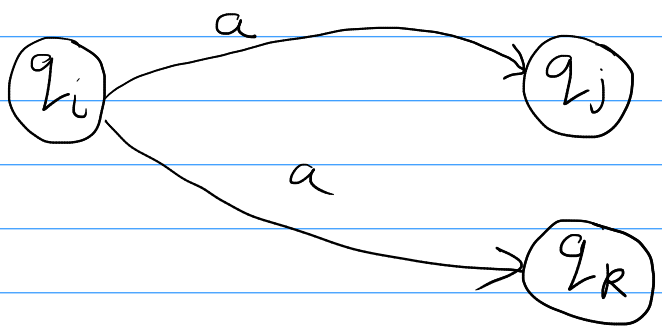
$d(x1) \bmod 3 = 0 \quad (q_0)$

$$\hat{\delta}(q_0, x) = q_{d(x) \bmod 3}$$

Deterministic finite automata

All the languages which are accepted by DFA, we call regular languages.

Non deterministic finite automata (NFA)



allowed

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

Power set of Q

$$\delta : Q \times \Sigma \longrightarrow 2^Q$$

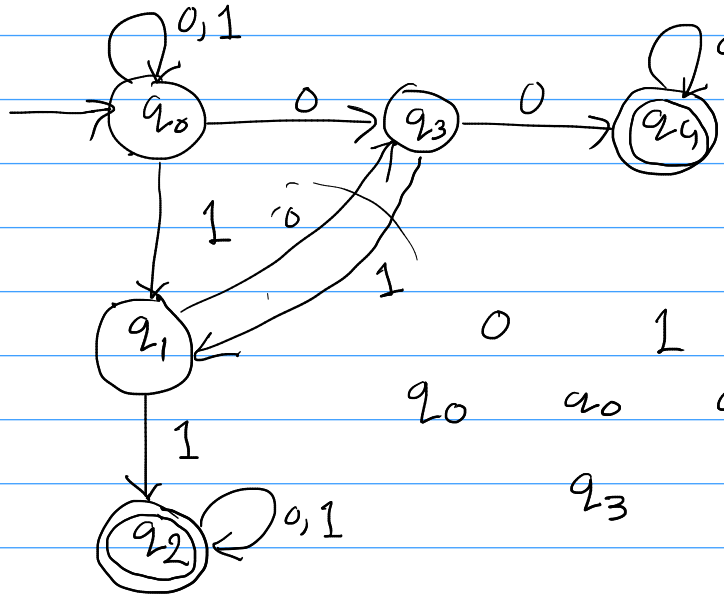
0 1 1 0 0 0 1

$\uparrow$ - - - - - $\{q_{i_1}, q_{i_2}, \ldots, q_{i_k}\}$

Accept: if ∃ a state which is final the last set of states.

Reject: ∀ state which are present in the final last set of state are non final.

$L = \{ x \in \{0,1\}^* \mid x$ has either two consecutive 0 or " " 1 $\}$

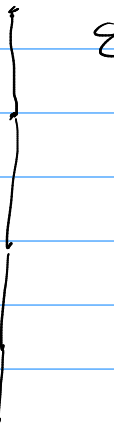


accepted

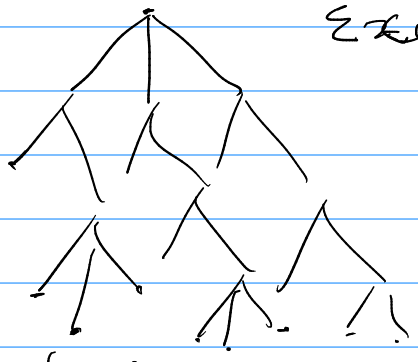
0	1	0	0	1	Last State
q_0	q_0	q_0	q_0	q_0	q_0
q_3	q_1	q_3	q_3	q_1	q_4
				q_4	

0	1	0	1	rejected
q_0	q_0	q_0	q_0	q_0
	q_3	q_1	q_3	q_1

No final State



Execution of DFA



Execution of NFA

final state

Defn: $\exists$ path in above tree which ends at final state

Accept:

Reject $\forall$ leaf nodes are not final

Defn2: Guess and Verify

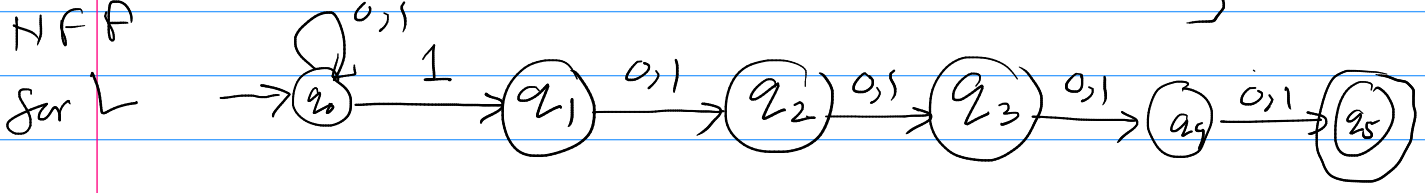
I/p guess

I/p : String, path

O/p : Y/N. Verify the guessed path.

$L = \{x \in \{0,1\}^* :$ the fifth symbol from right is $1\}$

NFA for L



Proof

$\hat{\delta}(q_0, x) = q_0 \Leftrightarrow d(x) \bmod 3 = 0$

$\hat{\delta}(q_0, x) = q_1 \Leftrightarrow d(x) \bmod 3 = 1$

$\hat{\delta}(q_0, x) = q_2 \Leftrightarrow d(x) \bmod 3 = 2$

Base case $x = \epsilon$ $\quad d(x) \bmod 3 = 0$

$\hat{\delta}(q_0, x) = q_0$.

The above expressions are true upto length x

xa

$$\hat{\delta}(q_0, xa) = \delta(\hat{\delta}(q_0, x), a)$$

$$= \delta(q_{d(x) \bmod 3}, a)$$

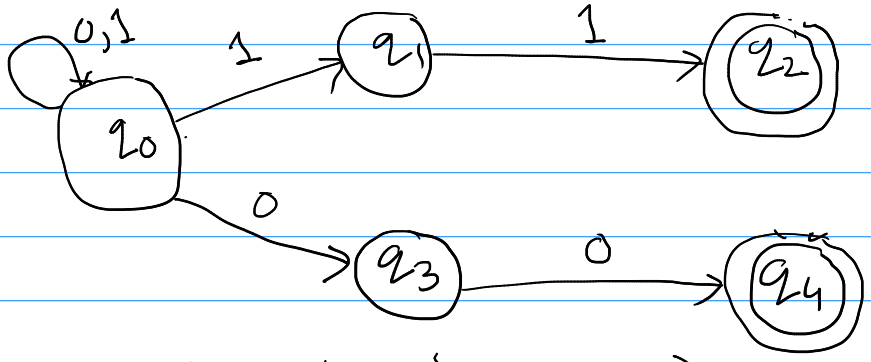
$$= q_{(2\, d(x) \bmod 3 + a) \bmod 3}$$

$$= q_{d(xa) \bmod 3}$$

NFA (hope less)

Thm:- Every NFA can be converted into an DFA.

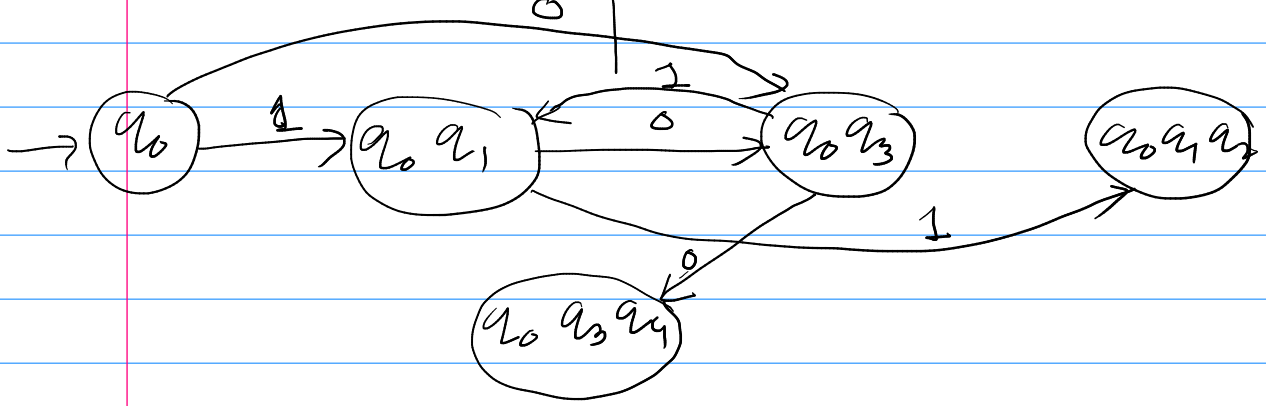
M

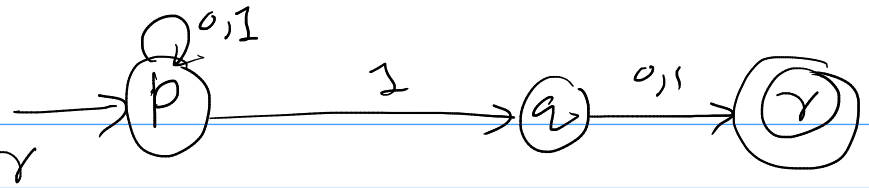


DFA $M' = (Q', \Sigma', \delta', q_0, F')$

$\Sigma' = \Sigma$

	0	1
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_3\}$	$\{q_0, q_3, q_4\}$	$\{q_0, q_1\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_3, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_3, q_4\}$	$\{q_0, q_3, q_4\}$	$\{q_0, q_1, q_4\}$



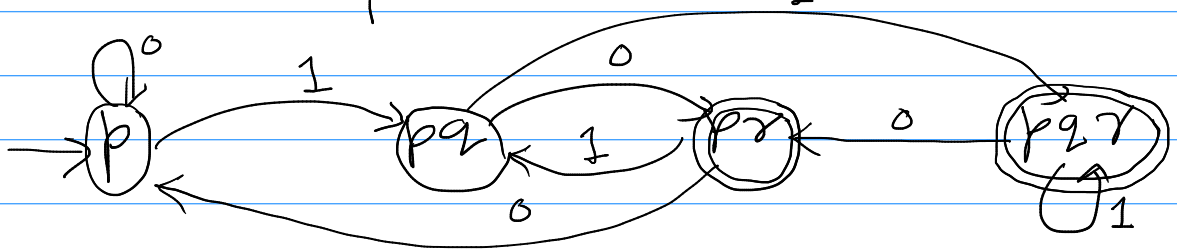


p q r

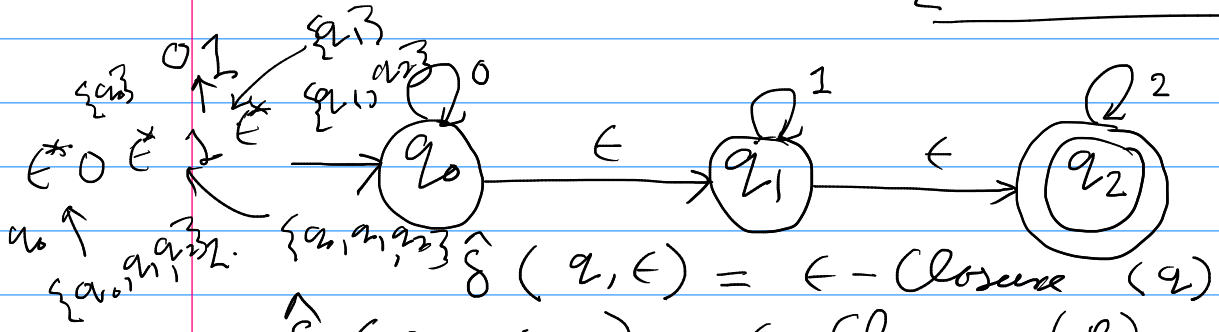
pq pr qr

pqr

	0	1
p	p	{p, q}
{p, q}	{p, r}	{p, q, r}
{p, r}	{p}	{p, q}
{p, q, r}	{p, r}	{p, q, r}



Subset Construction $L = \{ 0^i 1^j 2^k \mid i, j, k \geq 0 \}$



$\hat{\delta}(q, \epsilon) = \epsilon\text{-Closure}(q)$

2. $\hat{\delta}(q, wa) = \epsilon\text{-Closure}(P)$

$w \in \Sigma^*$ $\quad P = \{ p : \exists r \in \hat{\delta}(q_0, w), p \in \delta(r, a) \}$

$a \in \Sigma$

$\epsilon\text{-Closure}(q) = \{ p :$ $\exists$ a path from q to p labelled $\epsilon \}$

$\epsilon\text{-Closure}(q_0) = \{ q_0, q_1, q_2 \}$

$$\hat{\delta}(q_0,0) = \epsilon\text{-Closure}(\delta(\hat{\delta}(q_0,\epsilon),0))$$

$$= \epsilon\text{-Closure}(\delta(\{q_0,q_1,q_2\},0))$$

$$= \epsilon\text{-Closure}(\{q_0\} \cup \phi \cup \phi)$$

$\delta(q,0)$: do not have to calculate ϵ-Closure

$$= \{q_0, q_1, q_2\}$$

$\hat{\delta}$

$$\hat{\delta}(q_0, 01) = \epsilon\text{-Closure}(\delta(\hat{\delta}(q_0,0), 1))$$

$$= \epsilon\text{-Closure}(\delta(\{q_0,q_1,q_2\}, 1))$$

$$= \epsilon\text{-Closure}(\phi \cup \{q_1\} \cup \phi)$$

$$= \{q_1, q_2\}$$

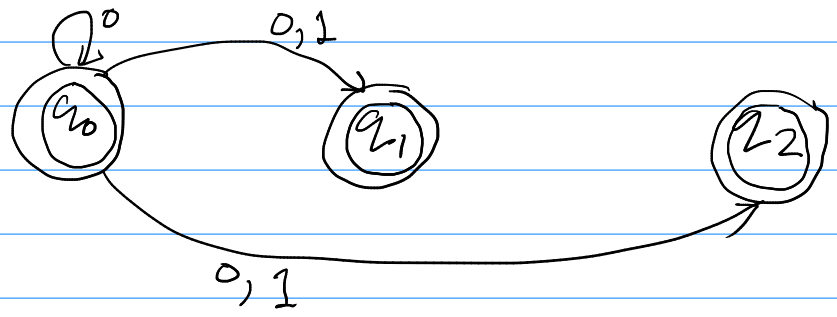
ϵ-NFA $\longleftrightarrow$ NFA

$\forall q_{\in Q} \forall a \in \Sigma$ Calculate $\hat{\delta}(q,a)$

in New NFA add transitions accordingly

$$\hat{\delta}(q_0,0) = \{q_0, q_1, q_2\}$$

$$\hat{\delta}(q_0,1) = \{q_1, q_2\}$$



NFA

Final state:- $\forall q \quad \epsilon\text{-Closure}(q) \cap F \neq \phi$

ϵ – NFA $\leftrightarrow$ NFA $\leftrightarrow$ DFA

Regular Expression :-

$L(a^*) =$ regular expression a^* $= \{\epsilon, a, aa, aaa, \dots\}$

$L(a^+) =$ regular expression a^+ $= a a^* = \{a, aa, aaa, \dots\}$

$L(\{00,11\}^*) = \{00, 11\}^* = \{\epsilon, 00, 11, 0000, 0011, 1100, 1111, 111100, 001100, 00111100, \dots\}$

~~00~~ 111

$L(\{0,1\}^*) = \{0,1\}^* = \{\epsilon, 0, 1, 001, \dots\}$

L_1, L_2

$L_1 L_2 = \{xy \mid x \in L_1, y \in L_2\}$

$L^i = L L^{i-1}$ (concatenation)

$L_1^* = \bigcup_{i=0}^{\infty} L^i$

$(0+1)$ = Either 0 or 1

$(0+1)^* = \{0,1\}^*$

ϕ r.e denotes empty set

ϵ r.e $\{\epsilon\}$

a r.e $\{a\}$

$a \in \Sigma$

$*$, finite concatenation, $+$

r, s r.e. represent $L(r)$, $L(s)$

$r+s$ " $L(r) \cup L(s)$

r_s r.e. $L(r)\,L(s)$

r^* r.e $(L(r))^*$

r.e. $\longleftrightarrow$ DFA

r.e on $\{0,1\}$ Such no two zeros are consecutive and string starts with 1

$1(1^*(01)^*1^*)^*$ or

$\{1, 10\}^+$ 101

r.e. $\longrightarrow$ ϵ-NFA

*, > concatenation, > + precedence relation b/w operators

r.e.: R

$0^*1 + O(00)^* + 1(01)^*$

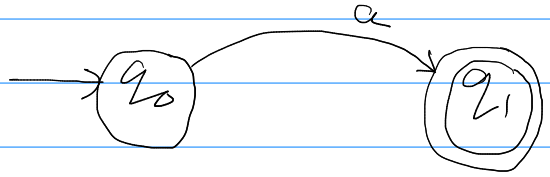
Base Case: (no operator)

ϕ ϵ a

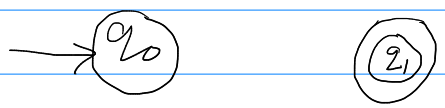
a

$\phi = \{\}$

ϵ a ϵ b ϵ $\{\epsilon\}$



ϵ

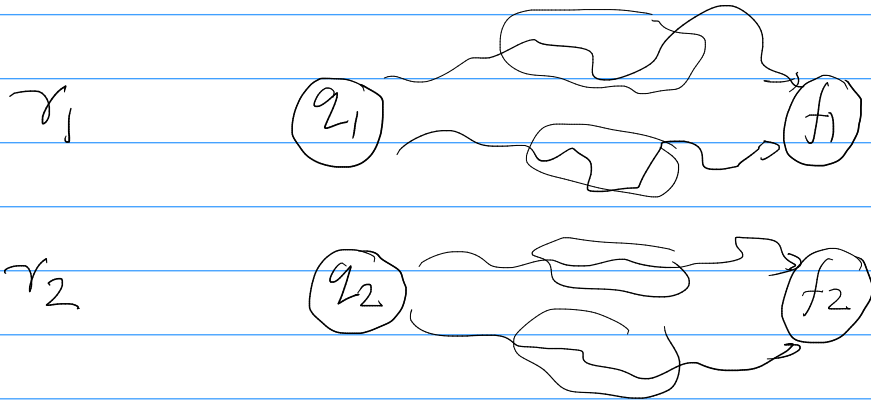


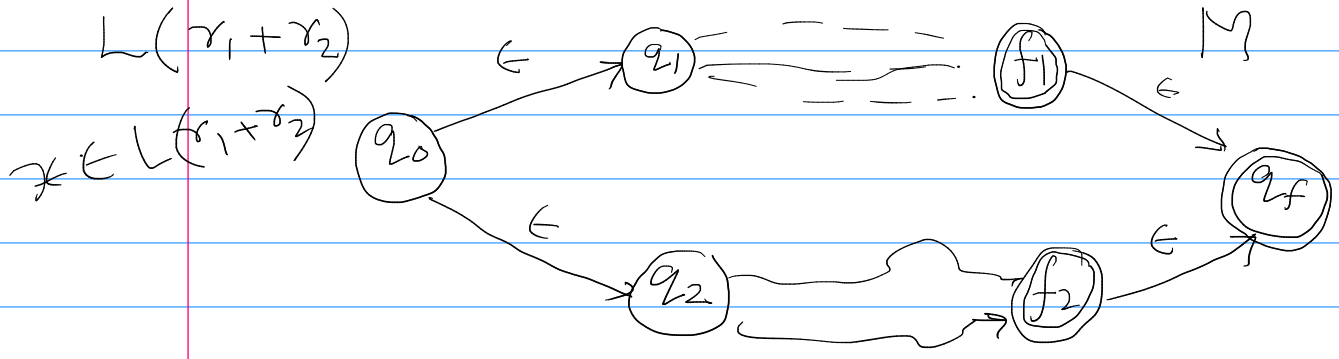
Assume that the theorem is true for r.e. with fewer than i number of operators.

r.e. $r = r_1 + r_2$

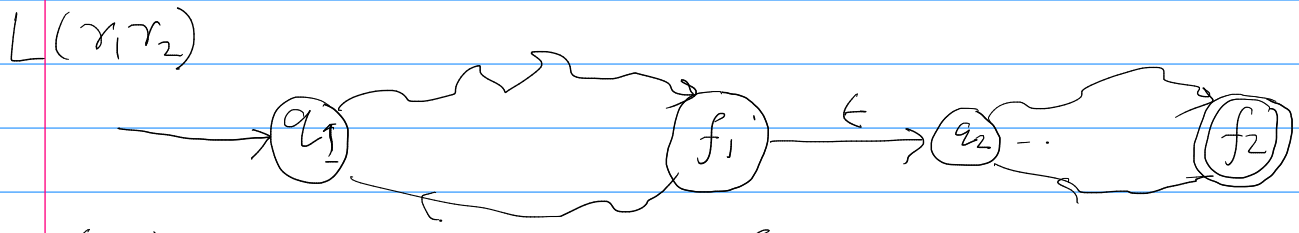
$r_1 r_2$

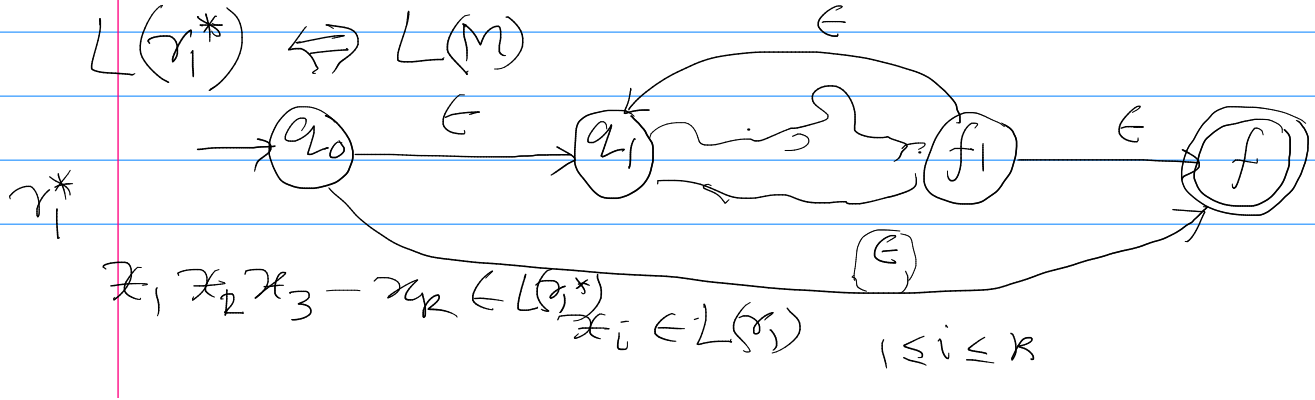
$(r_1)^*$



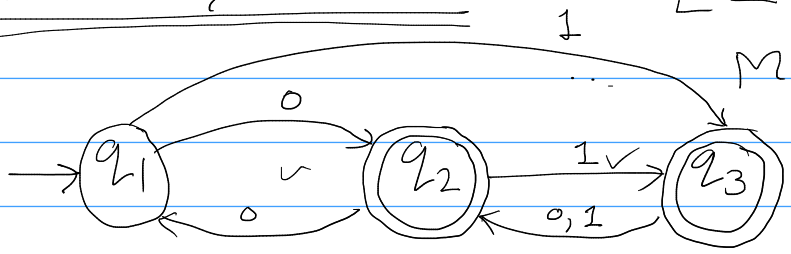


$x \in L(M)$





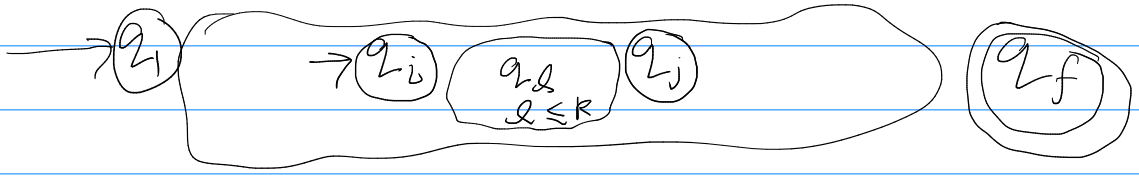
DFA $\longrightarrow$ r.e: $L = L(M)$

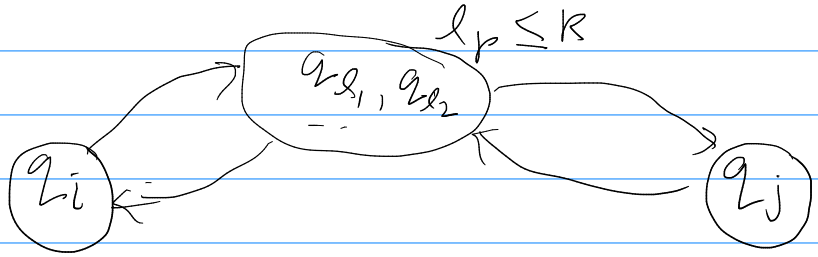


$0(00)^*$

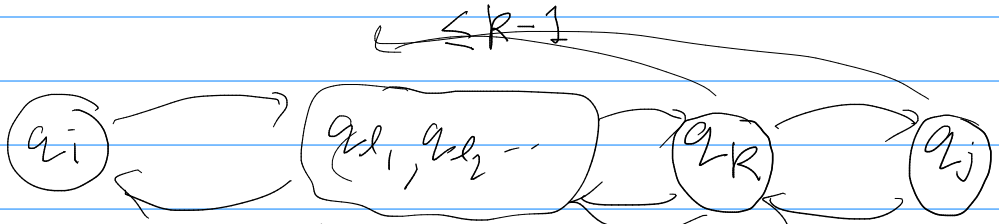
Recursively:

R_{ij}^k : denotes all string that takes q_i to q_j without visiting states q_ℓ such that $\ell > k$.





$R_{ij}^k =$

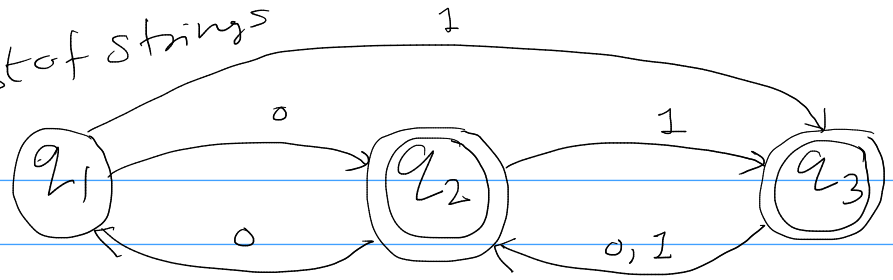


$$R_{ij}^k = \underbrace{R_{ij}^{k-1}}_{} \cup \underbrace{R_{ik}^{k-1} \left(R_{kk}^{k-1}\right)^* R_{kj}^{k-1}}_{\text{using } q_k}$$

Base Case

$$R_{ij}^0 = \begin{cases} \{a : \delta(q_i, a) = q_j\} & i \neq j \\ \{a \cup \{\epsilon\} \ \ \delta(q_i, a) = q_j\} & i = j \end{cases}$$

R - Set of strings



$r_{12}^3 + r_{13}^3 = ?$

$$r_{12}^3 = r_{13}^2 (r_{33}^2)^* r_{32}^2 + r_{12}^2$$

r_{13}^2 , r_{33}^2 , r_{32}^2 , r_{12}^2

$$r_{13}^2 = r_{12}^1 (r_{22}^1)^* r_{23}^1 + r_{13}^1$$

$r_{11}^2 = (00)^*$

	R=0	R=1	R=2
r_{11}^R	ϵ	ϵ	$(00)^*$
r_{12}^R	0		
r_{13}^R	1		
r_{21}^R	0		
r_{22}^R	ϵ		
r_{23}^R	1		

$r_{11}^1 = \epsilon$

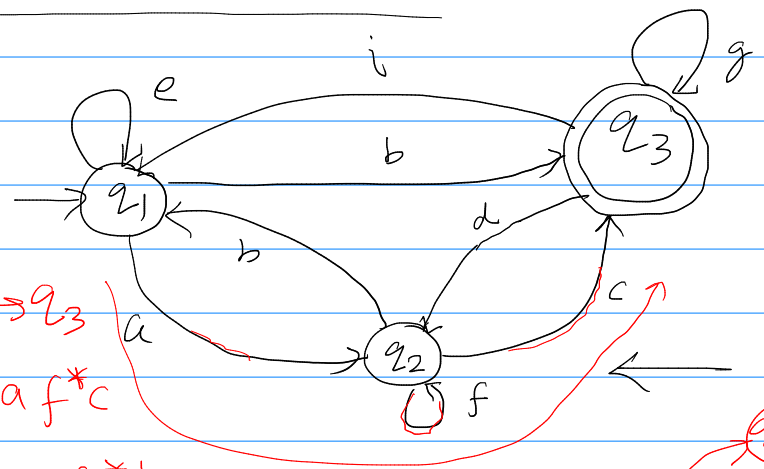
$r_{22}^1 = r_{21}^0 (r_{11}^0)^* r_{12} + r_{22}^0$

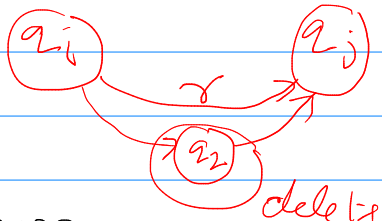
=

$r_{11}^2 = r_{12}^1 (r_{22}^1)^* r_{21} + r_{11}^1$

=

Alternate Method





removing q_2

q_i q_2 q_j

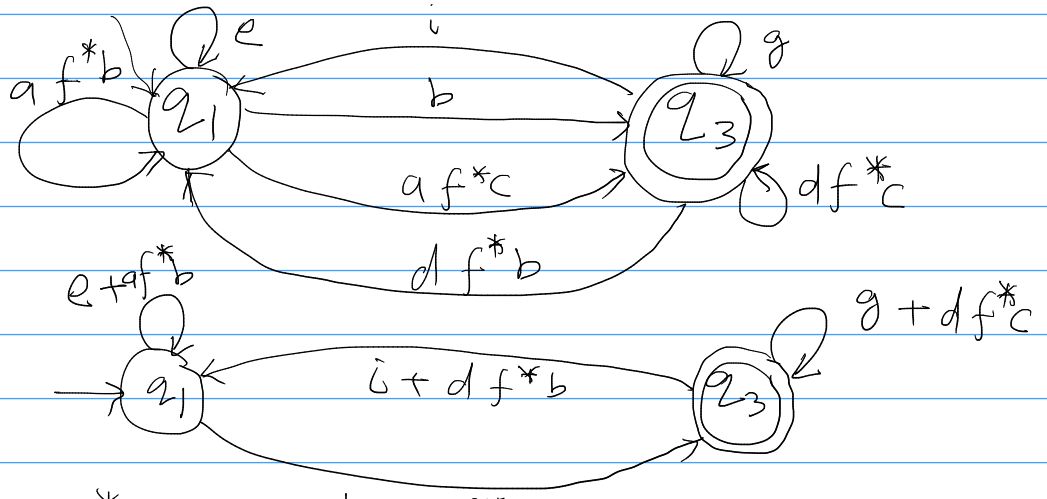
$q_1 \rightarrow q_2 \rightarrow q_3$

(q_1, q_3) af^*c

(q_1, q_1) af^*b

(q_3, q_1) df^*b

(q_3, q_3) df^*c



$$(e+af^*b)^*(b+af^*c)^{b+af^*c}(g+df^*c)^* = X$$

$$(X(i+df^*b)X)^* + X$$

FA with o/p.

Mealy M/C — o/p will be associate with transition

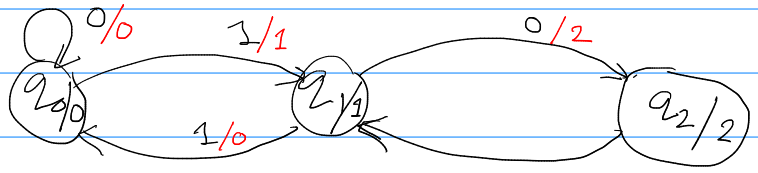
Moore M/C — o/p will be associated with States

Moore M/C

$d(x) \in \{0,1\}^*$ mod 3 = 0

i/p: 0 1 1 0

o/p: 0 0 1 0 0



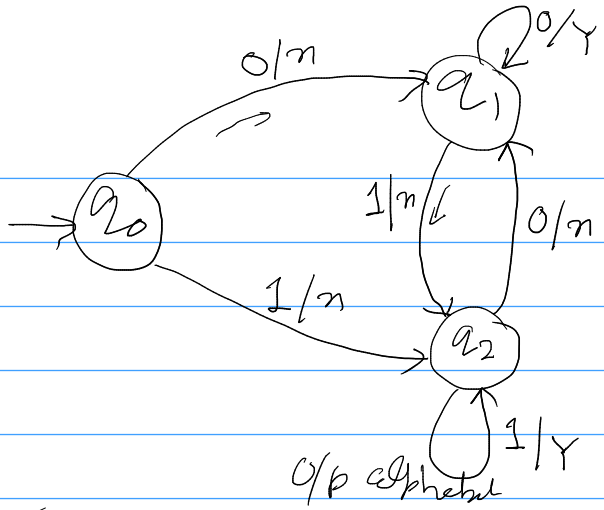
0 1 0 2 — — 0

Mealy M/C

$x \in \{0,1\}^*$ ends with 00 or 11

i/p: 0 1 1 0

o/p: n n y n



o/p alphabet, o/p function

$$M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$$

$$\begin{cases} \lambda : Q \rightarrow \Delta & \text{Moore} \\ \lambda : Q \times \Sigma \rightarrow \Delta & \text{Mealy} \end{cases}$$

New M/C

$$M = (Q, \Delta, \Sigma, \delta, \lambda, -) \quad \text{Mealy}$$

$Q \times \Delta$

$\Delta = \{y, n\}$

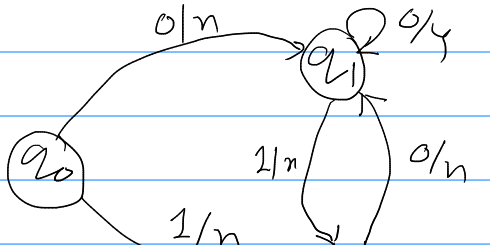
$[q, y]$ $[q, n]$ $\forall q \in Q$

$$\delta([q, b], a) = [\delta'(q, a), \lambda(q, a)] \quad a \in \Sigma$$

transition function for new M/C

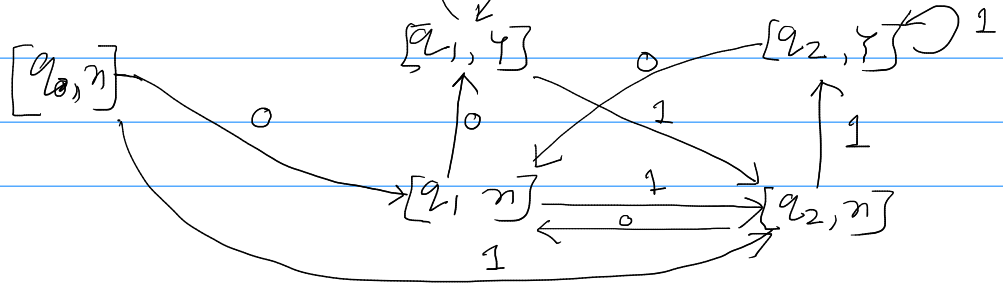
old M/C old

Moore



$\delta'(q_1, 1) = q_2$

$\lambda(q_1, 1) = n$

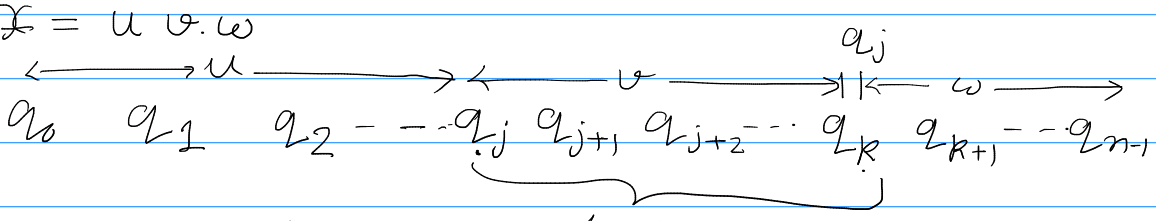


Pumping Lemma

DFA M finite number of states. n

$x \in \Sigma^*$ $|x| > n$

c/b $x = uvw$



$|uv| \leq n$

q_0 q_1 q_2 q_j q_{k+1} - - q_{n-1}

uw accepted if x has got accepted.

$v^i = vv$ - - i times

uv^iw will also be accepted.

Let L be a regular Language. Then the following property holds for L.

$\exists\ K \geq 0$ such that for any $z \in L$ and $|z| \geq K$ $\exists\ (u, v, w)$ such that $z = uvw$ $|v| \geq 1$ $|uv| \leq n$ and for all $i \geq 0$ the string $uv^iw \in L$.

Necessary condition not sufficient

$p \rightarrow q$ Contraposition $\overline{q} \rightarrow \overline{p}$

quantifiers $\forall, \exists$ are alternating.

$\forall R \geq 0 \quad \exists z \in L \quad |z| \geq R$ and

$\forall (u, v, w)$ with $z = uvw$ and $|v| \geq 1$, $|uv| \leq n$

$\exists i$ such that $uv^i w \notin L$

$\Rightarrow$ {L is not regular}

Combinatorial Game

$\forall$ player 1

$\exists$ player 2

player 1 ($\forall$)

Pick $R \geq 0$

$z = uvw \quad |v| \geq 1$

$|uv| \leq n$

player 2 ($\exists$)

$|z| \geq R, \; z \in L$

$uv^i w \quad i \geq 0$

if $uv^i w \notin L$

player 2 will win

otherwise player 1 will win.

$L = \{ a^{R^2} \mid R \geq 1 \}$ Not regular.

$\subseteq$ a^* is regular

Choose $n > 0$

$$z \in L \qquad |z| = n^2$$

$$z = a^{n^2}$$

$$z = uvw = a^{n^2}$$

$$u = a^i \qquad v = a^j$$

$$w = a^{n^2 - i - j}$$

$1 \le |v|$

$|uv| \le n$

$j \ge 1$

$i + j \le n$

$$uv^2w = a^i a^j a^j a^{n^2-i-j}$$

$$= a^{n^2+j} \in L \quad \text{Not}$$

$n^2 + j = m^2$ possible for all j

Hence L is not regular.

$$L = \{ 1x \mid x \in \{0,1\}^* \quad d(1x) \text{ is prime} \}$$

($d(1x)$: integer number corresponding to $1x$)

Not regular.

{ infinitely many primes.

Fermats Little theorem

$a^p = a \bmod p$ when p is prime

finitely many primes

$p_1, p_2, \ldots p_n$

$M = p_1 p_2 \cdots p_n + 1$

M is not divisible by any of the above prime numbers.

means we have more prime numbers

Fermat's little theorem

Lemma: $\binom{p}{r}$ is divisible by p (prime)

$$\binom{p}{r} = \frac{p(p-1)\cdots(p-r+1)}{r(r-1)\cdots 1}$$ when p is a prime, $r < p$

$r(r-1)\cdots 1$ can not divide p

Since $\binom{p}{r}$ is an integer $p \mid \binom{p}{r}$

Thm $a^p - a$ is divisible by p (prime)

Induction: $a = 1$ true.

$a^p - a$ is divisible by p.

$(a+1)^p - (a+1)$ is divisible by p

$a - b$ is divisible by $p \Rightarrow$

$a^k - b^k$ " " " p

$k = 1$ $a - b$ is divisible by p

$k = m$ $a^m - b^m$ " " p

Base for

$k = m+1$ $a^{m+1} - b^{m+1}$ divisible by p.

$$a^m - b^m = pt$$

$$a^{m+1} - ab^m = apt$$

$$a - b = pl$$

$$a = pl + b$$

$$a^{m+1} - ab^m - b^{m+1} = apt$$

$$a^{m+1} - b^{m+1} = (at + lb^m)\,p$$

Hence proved

$a - b$ divisible by p

$a \equiv b \bmod p$ ↑ Congruence ⇓

$a^k \equiv b^k \bmod p$

change of notation

binary numbers

x $\quad d(x)$

yx $\quad d(yx) = 2^{|x|}\, d(y) + d(x)$

yyx $\quad d(yyx) = 2^{|yx|}\, d(y) + d(yx)$

$= 2^{|y|+|x|}\, d(y) + 2^{|x|}\, d(y) + d(x)$

$= d(y)\, 2^{|x|}\,(2^{|y|} + 1) + d(x)$

$d(yyyx) = d(y)\, 2^{|x|}\,(2^{2|y|} + 2^{|y|} + 1) + d(x)$

$d(zyyyx) = d(z)\, 2^{|x|+3|y|} + d(y)\, 2^{|x|}\,(2^{2|y|} + 2^{|y|} + 1) + d(x)$

$L = \{1x \mid x \in \{0,1\}^* \;\; d(1x) \text{ is prime}\}$

Suppose L is regular.

Pick $n > 0$, $z \in L$, $|z| \geq n$

$\Rightarrow$

$z = uvw$ $\quad d(z)$ as prime.

$uv^i w \quad \forall i \geq 0 \quad d(uv^i w)$ is prime

$\Rightarrow uv^i w \in L$

Not the case for all i

$z = uvw = 1^x \in L \quad x \in \{0,1\}^*$

$d(z) = d(uvw)$

$p = 2^{|v|+|w|} d(u) + 2^{|w|} d(v) + d(w)$ is prime

$i = p$

Claim $d(uv^p w)$ is not prime

$$d(uv^p w) = d(u)2^{|w|+p|v|} + 2d(v)\left(1 + 2^{|v|} + 2^{2|v|} + \cdots 2^{(p-1)|v|}\right) + d(w)$$

$p \neq 2$

$2^{(p-1)|v|} \equiv 1 \bmod p \Leftarrow 2^{(p-1)} \equiv 1 \bmod p$

$2^p \equiv 2 \bmod p$

$\frac{2(2^{p-1}-1)}{p}$

$p \neq 2$

$S = 1 + 2^{|v|} + 2^{2|v|} + \cdots 2^{(p-1)|v|} = \frac{2^{p|v|}-1}{2^{|v|}-1}$

$(2^{|v|}-1)\, S = 2^{p|v|} - 1 \Leftarrow$

$a \equiv b \bmod p \Rightarrow ac \equiv bc \bmod p$

$2^{(p-1)|v|} \equiv 1 \bmod p$

$\Downarrow$

$2^{(p-1)|v|} \times 2^{|v|} \equiv 1 \times 2^{|v|} \bmod p$

$2^{p|v|} \equiv 2^{|v|} \bmod p$

$2^{p|v|} - 1 \equiv 2^{|v|} - 1 \bmod p$

multiply with s

$s(2^{p|v|}-1) \equiv s(2^{|v|}-1) \bmod p$

$s(2^{p|v|}-1) \equiv (2^{|v|p}-1) \bmod p$

$(s-1)(2^{p|v|}-1) \equiv 0 \bmod p$

Claim: $\boxed{2^{p|v|}-1 \bmod p \neq 0}$

$$2^{p|v|} \equiv 2^{|v|} \bmod p$$

$$2^{p|v|}-1 \equiv 2^{|v|}-1 \bmod p$$

$d(uvw) = p$

$|v| < |z| = |uvw|$

$\Rightarrow$ $\boxed{2^{p|v|}-1 \text{ not divisible by } p}$ $\forall v$ $\because d(v) < d(uvw) = p$

$2^{|v|}-1 < d(uvw) = p$

Hence $S-1$ is divisible by p.

$\Rightarrow S \equiv 1 \bmod p$

$$d(uv^p w) \overset{\Rightarrow}{=} \boxed{d(u)\,2^{|w|+p|v|} + d(v)\,2^{|w|}S + d(w)}$$

$$\equiv d(u)\,2^{|w|+p|v|} + d(v)\,2^{|w|} + d(w) \bmod p$$

$2^{p|v|} \equiv 2^{|v|} \bmod p$

$$\equiv d(u)\,2^{|w|+|v|} + d(v)\,2^{|w|} + d(w) \bmod p$$

p is odd

$$\equiv 0 \bmod p$$

$$S \equiv 1 \bmod p$$

$$d(u)2^{|w|+p|v|} + d(w) + d(v)2^{|w|}S \equiv d(v)2^{|w|} + d(w) + d(u)2^{|w|+p|v|} \bmod p$$

$$d(uv^p w) \equiv d(v)2^{|w|} + d(w) + d(u)2^{|w|+p|v|} \bmod p$$

$2^{|v|p} \equiv 2^{|v|} \bmod p$

$$\equiv d(v)2^{|w|} + d(w) + d(u)2^{|w|+|v|} \pmod p$$

$$\equiv d(uvw) \bmod p$$

$$\equiv p \bmod p$$

$d(uv^p w) \equiv 0 \bmod p$ Hence L is not regular.

$2^p \equiv 2 \mod p$ FLT

$2^{p|w|} \equiv 2^{|w|} \mod p$ (hold)

$2^{p|w|} - 1 \equiv 2^{|w|} - 1 \mod p$ (hold)

$d(uvw) = p$ minimum

$2^{|v|} < p$

Closure properties of regular languages

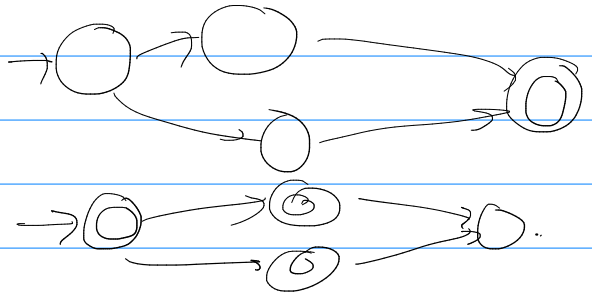
L_1, L_2 are regular. $\Rightarrow$

$L_1 \cup L_2$, $L_1 L_2$, L_1^*, $L_1 \cap L_2$, $\overline{L_1}$

$L_1^R = \{w^R \mid w \in L_1\}$ are regular.

$\overline{L_1}$: Let M be a DFA which accepts L_1.

{ all non final state make it final
{ all final " " " non final



L

$\overline{L}$

$L_1 \cap L_2$: $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ regular

Non Constructive Proof

$L_1 = L(M_1)$ $\quad M_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$

$L_2 = L(M_1)$ $\quad M_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$

$$M = (Q_1 \times Q_2, \Sigma, \delta, q_{01} \times q_{02}, F)$$

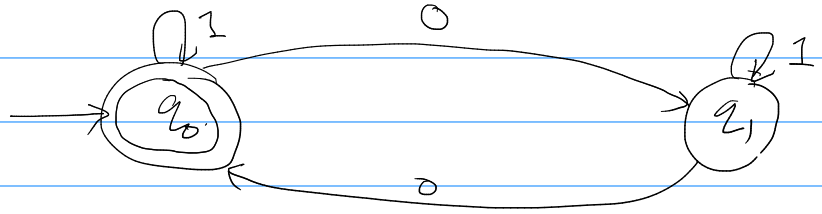
$$F = F_1 \times F_2 \qquad (\overset{\in Q}{q_i}, \overset{\in F_1}{q_f}) \notin F$$

$$\delta((q_i, q_j), a) = (\delta_1(q_i, a), \delta_2(q_j, a))$$

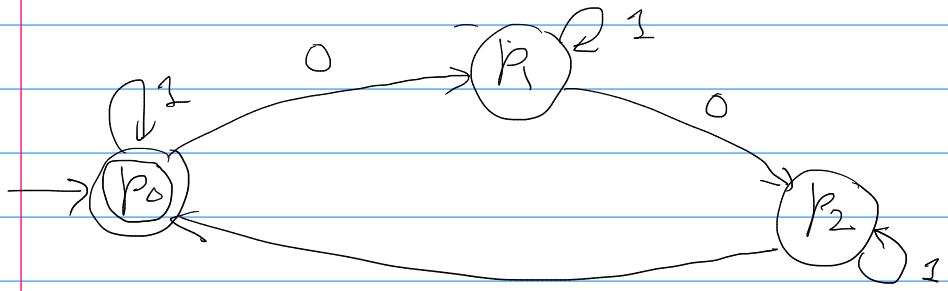
$L_1 = \{ x \in \{0,1\}^* \mid x$ contains even number of zeros$\}$

$L_2 = \{ x \in \{0,1\}^* \mid$ number of zeros in x mod 3 is 0$\}$

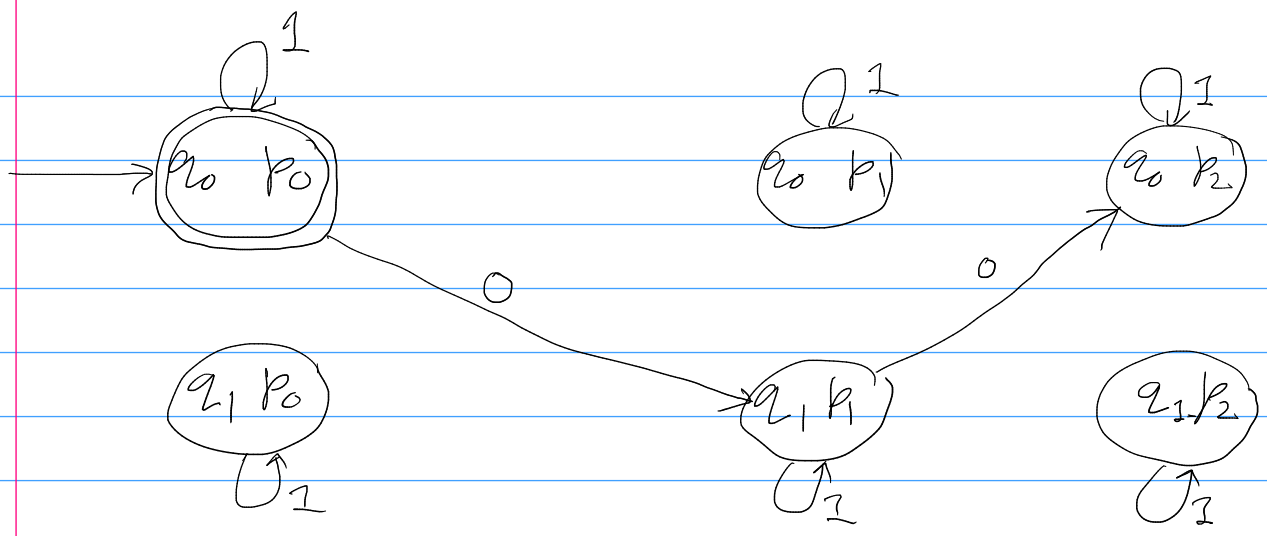
$L_1 \cap L_2 = \{ x \in \{0,1\}^* \mid$ number of zeros in x is divisible by 2 and 3$\}$



L_1



L_2



Let L be a regular language.

$L^R = \{ w^R \mid w \in L \}$

$w = a_1 a_2 \cdots a_n \in L$

$w^R = a_n a_{n-1} \cdots a_2 a_1 \in L^R$

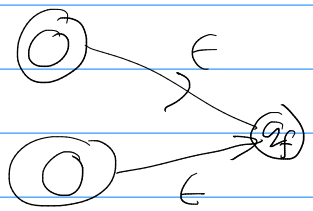
Claim: L^R is regular.

DFA M for L

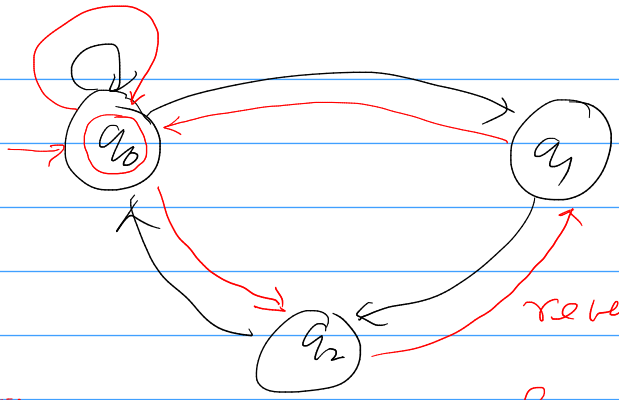
M' : 1. if M contains more than one final states then

(a) add a new final state q_f

(b) add $\in$ transition from the all the previous final states to q_f



2: Reverse all the arc present in the above obtained DFA.



$w \in L(M) \iff w^R \in L(M')$

Homomorphism

$0 \to abc$
$1 \to ac$

$0\ 1\ 1$

$abc\ ac\ ac$

Σ, Γ alphabets

$h : \Sigma \longrightarrow \Gamma^*$

$w = a_1 a_2 \cdots a_n \in \Sigma^*$

$h(w) = h(a_1)\, h(a_2) \cdots h(a_n) \in \Gamma^*$

$h(0) = abc$ $\quad \Sigma = \{0,1\}$

$h(1) = ac$ $\quad \Gamma = \{a, b, c\}$

$w = 011$ $\quad h(w) = abc\,ac\,ac$

$h(L) = \{ h(w) \mid w \in L \}$ $\quad \Gamma = \{a, b, c\}$

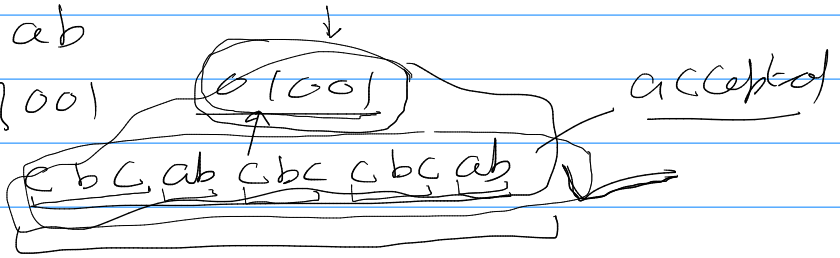
$\Gamma^* = \{ab, cac, \cdots\}$

$h(0) = cbc$

$h(1) = ab$

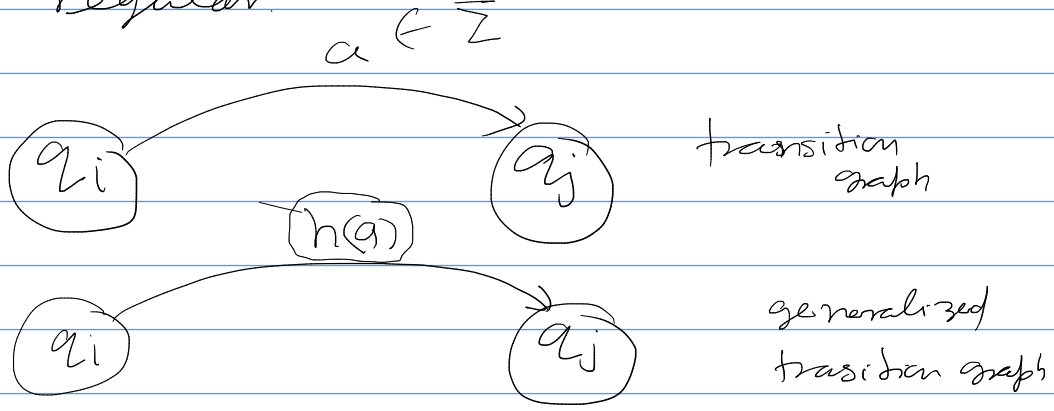
$w = 0\,1\,001$

$h(w) = cbc\,ab\,cbc\,cbc\,ab$



$h(w) \to w$

if L is regular then $h(L)$ is also regular.



$\forall\ i, j$ and a in the original M/C.

$$x \in L(M) = L \iff h(x) \in L(M')$$

Let L_1 and L_2 be languages on some alphabet

$$h(x) = w$$

$$w \in L(M') \Rightarrow \exists\, x \in L(M)$$

$$\forall\, x \in L(M) \Rightarrow h(x) \in L(M')$$

L_1, L_2 regular

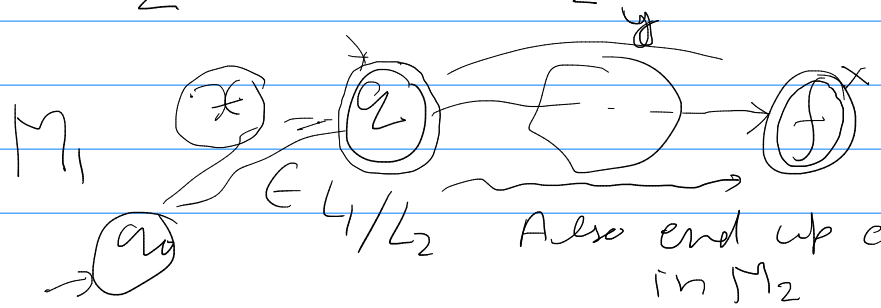
right quotient

Claim: $L_1 / L_2 = \{ x : xy \in L_1 \text{ for some } y \in L_2 \}$

regular.

L_1 DFA M_1

L_2 DFA M_2



M_1 q_0

$M_i' = (Q, \delta, \Sigma, \{q_i\}, F)$

$L(M_i') \cap L(M_2)$ is not empty

then make q_i as final state

$L_1 = \{a^n b^m : n \geq 1, m \geq 0\} \cup \{ba\}$ regular

$L_2 = \{b^m : m \geq 1\}$ $m \geq 0$

$L_2 = \{b, b^2, b^3, \ldots\} \in L_2$ regular.

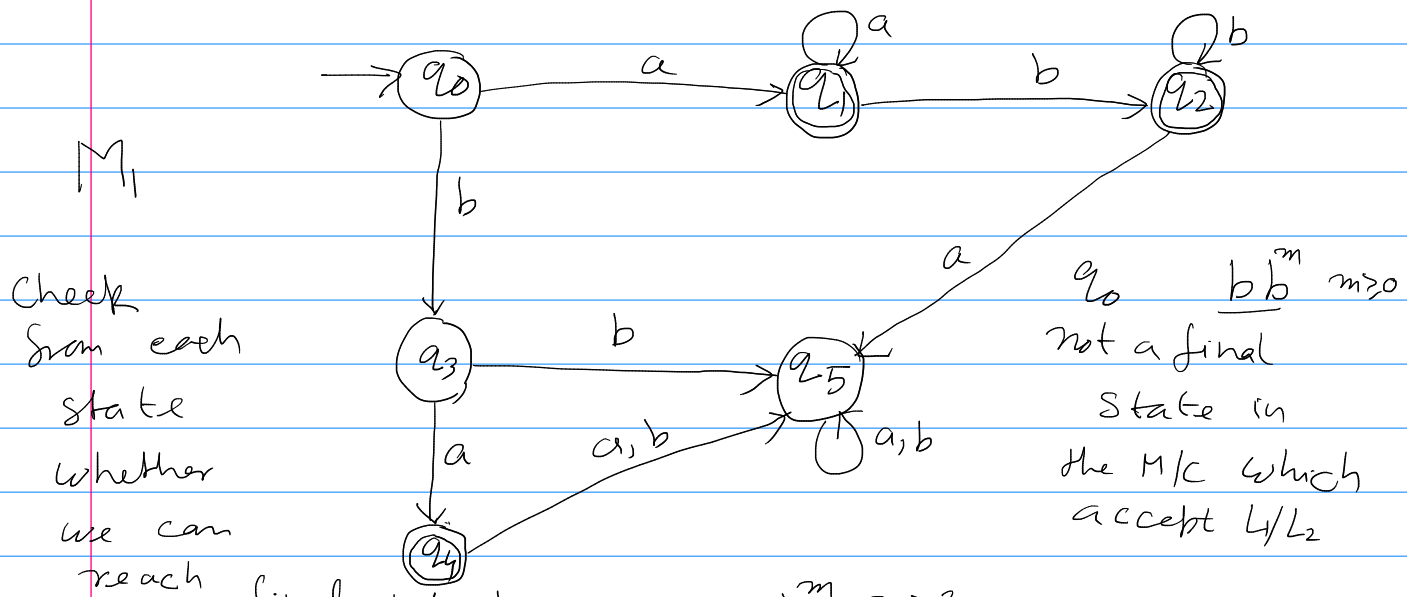
$L_1/L_2 = \{a^n b^m : n \geq 1, m \geq 0\}$

$L_1/L_2 = \{x \mid \exists\, y \in L_2 \text{ such that } xy \in L_1\}$

$ba \in L_1$ but there exist no string in L_2 which ends with a.

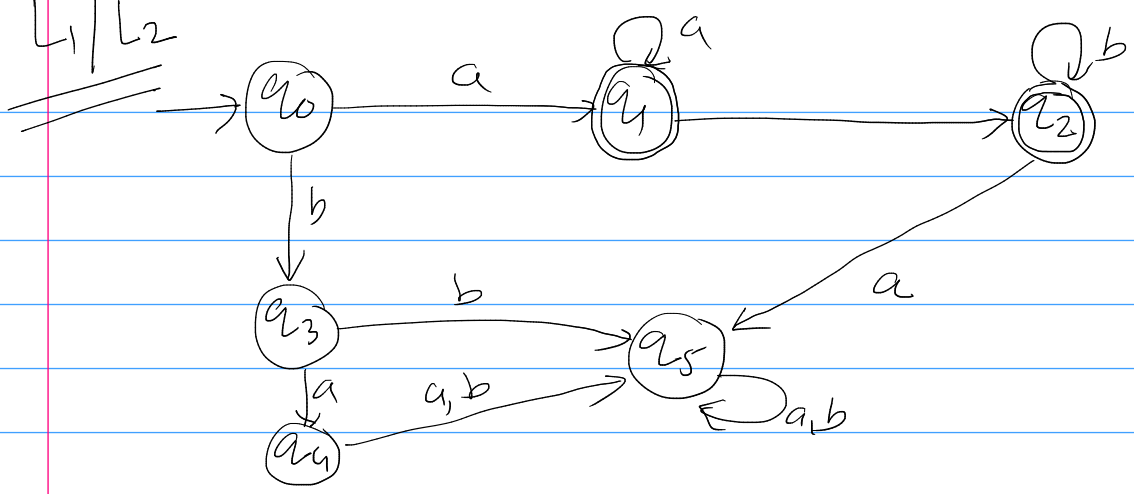
$x = a^n \in L_1/L_2$ $\quad i \geq 1$

$a^n b \in L_1/L_2$ $\quad y = b^i \in L_2$ $\quad xy = a^n b^i \in L_1$



Check from each state whether we can reach final state by scanning b^m $m \geq 1$.

q_0 bb^m $m \geq 0$ not a final state in the M/C which accept L_1/L_2



Equivalence Relation

S

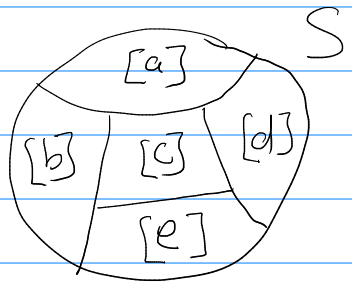
Relation $R \subseteq S \times S$ is an equivalence relation if the following conditions hold.

(a) $a R a \quad \forall a \in S$

(b) $a R b \Rightarrow b R a \quad \forall a, b \in R$

(c) $a R b \wedge b R c \Rightarrow a R c \quad \forall a, b, c \in R.$

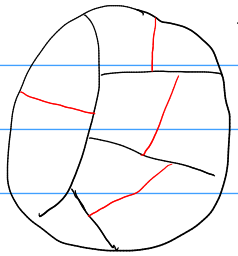
If R is an equivalence relation then it partition the set S in to equivalence classes.



$[a] = \{ x \mid a R x \}$

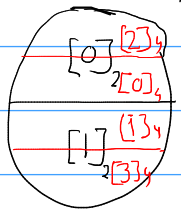
Refinement of equivalence relation R.

An equivalent relation R' is a refinement of R if every equivalent class in R' is completely contained in a equivalent class in R



R' partition

R partition.



$\mathbb{N}$ R mod 2

R' mod 4

refinement of R.

$x R' y \Rightarrow x R y$

$\Rightarrow$ R' is refinement of R.

An equivalent relation R has finite index if number of equivalence classes in R is finite.

Language $L \subseteq \Sigma^*$

Equivalence relation

$x R_L y$ if and only if $\forall z \in \Sigma^*$

either xz and yz both belong to L

or neither xz nor yz belong to L

x, y are not related in R_L

$x \not R_L y \Leftrightarrow \exists z$ such that $xz \in L$ and $yz \notin L$

or $xz \notin L$ and $yz \in L$.

it also implies that x, and y are in two different equivalent class

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

$x, y \in \Sigma^*$ $\quad x R_M y \iff \delta(q_0, x) = \delta(q_0, y) = q \in Q$

Claim: Equivalence relation.

$[x]$ — state

L = Union some equivalent classes.

$y \in [x]$ $\quad x R_M y \iff \delta(q_0, x) = \delta(q_0, y)$

right invariant $\quad R \subseteq \Sigma^* \times \Sigma^*$

if $x R y$ then $xz R yz \quad \forall z \in \Sigma^*$

R_M is right invariant:-

$x R_M y \iff \delta(q_0, x) = \delta(q_0, y) = q$

$\forall z \in \Sigma^*$ $\quad xz R_M yz \iff \delta(q_0, xz) = \delta(q_0, yz)$

$\delta(\delta(q_0, x), z) = \delta(\delta(q_0, y), z)$

$\delta(q, z) = \delta(q, z)$

Myhill Nerode Theorem:- The following 3 statements are equivalent.

1. $L \subseteq \Sigma^*$ is accepted by some FA
2. L is the union of some of the equivalence classes of a right invariant equivalence relation of finite index.

3. Equivalence relation R_L be defined by: $x R_L y \Leftrightarrow \forall z \in \Sigma^* \; xz \in L$ exactly when $yz \in L$. Then R_L is of finite index.

Proof: (1) $\Rightarrow$ (2)

L $\quad M = (Q, \Sigma, \delta, q_0, F) \leftarrow$

$x R_M y \Leftrightarrow \delta(q_0, x) = \delta(q_0, y)$

R_M is right invariant.

R_M has finite index.

L = union of equivalence classes which were associated with final states in M.

(2) $\Rightarrow$ (3)

R' for Statement 2

We will prove that. R' is a refinement of R_L.

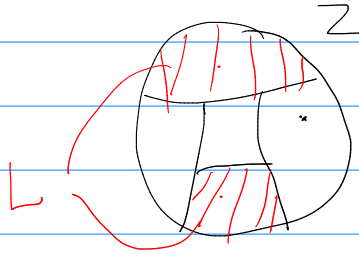
$x R' y \Rightarrow \forall z \in \Sigma^* \; xz R' yz$

because R' is right invariant.

$xz \in L \Leftrightarrow yz \in L$

$\Uparrow$ definition of R_L

Σ^*



$x R_L y$

Hence R' Refines R_L

Since R' has finite index.

R_L finite index.

(3) $\Rightarrow$ (1)

$x R_L y \iff \forall z \in \Sigma^* \quad xz \in L$ exactly when $yz \in L$.

R_L has finite index.

Claim: R_L is right invariant.

$x R_L y \Rightarrow \forall z' \in \Sigma^* \quad xz' R_L yz'$

$\forall z' \in \Sigma^* \; xz' R_L yz'$

$x R_L y \Rightarrow \forall z \in \Sigma^* (xz \in L \iff yz \in L)$

$\Rightarrow \forall z'z'' \in \Sigma^* \; (xz'z'' \in L \iff yz'z'' \in L)$

$\Rightarrow \forall z'' \in \Sigma^* ((xz')z'' \in L \iff (yz')z'' \in L)$

$\Rightarrow xz' R_L yz'$

R_L is right invariant.

Construct DFA M which accepts L.

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

R_L has finitely many equivalence classes.

Each equivalence class will work as state for M.

$[x] = \{y : x R_L y\}$

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

Well defined or Consistent

not well defined function

$\begin{cases} f(m/n) = m+n \\ f(3/2) = 5 \\ f(6/4) = 10 \end{cases}$

$y \in [x] \qquad \hat{\delta}(q_0, y) = \hat{\delta}(q_0, x)$

may arise in the previous definition

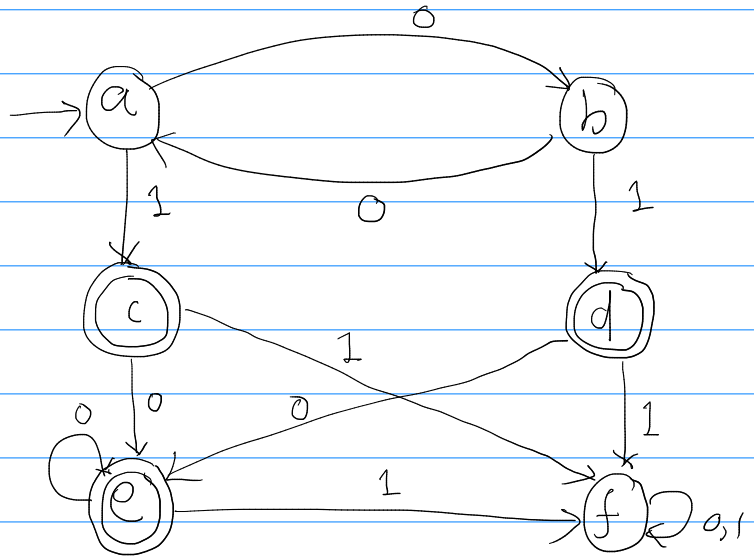
$y \in [x] \qquad \hat{\delta}(q_0, xa) \neq \hat{\delta}(q_0, ya)$

Since R_L is right invariant

$[xa] = [ya]$

$q_0 = [\epsilon] \qquad F = \{ [x] \mid x \in L \}$

$L = L(0^* 1 0^*)$



R_M

$[a] = (00)^*$

$[b] = (00)^* 0$

$[c] = (00)^* 1$

$[d] = (00)^* 01$

$[e] = (00)^* 100^* + (00)^* 0100^* = X = 0^* 100^*$

$[f] = (00)^* 011(0+1)^* + (00)^* 11(0+1)^* + X1(0+1)^* = 0^* 10^* 1(0+1)^*$

$L = [c] \cup [d] \cup [e]$

R_L

1: No one's in string

2: Exactly one 1 in string

3: ≥ 2 1's in string.

Union of above 3 sets is $\{0,1\}^*$

(1) (2) R_L

$00 \in$ first set –

$01 \in$ Second Set –

$z = 1$

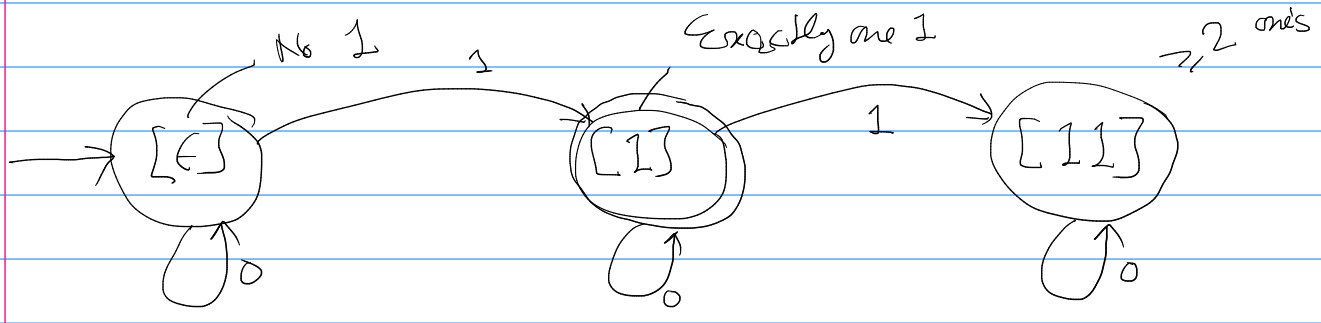
$00z \in L$

$011 \notin L$

$00 \in 1^{st}$ Set

$000 \in 1^{st}$ Set

$\forall z \in \{0,1\}^*$ $\quad 00z \; R_L \; 000z$



Minimization of states in DFA.

States p, q

p is indistinguishable to q if $\exists\ x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice versa.

xRy
$\forall z \in \Sigma^*$, $xz, yz \in L$
or $xz, yz \notin L$

Context free Grammars

$G = (V, T, P, S)$

- P: Set Production
- S: Start Symbol.
- T: Set of terminals
- V: Set of variables

$V = \{E\}$ $\quad T = \{+, *, (,), id\}$

P: $\{$
$S \rightarrow E$
$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow (E)$
$E \rightarrow id$ $\}$

G is context free if LHS of P contains exactly one variable and RHS contains some string from alphabet $(V \cup T)^*$

$E \rightarrow E+E \mid E*E \mid (E) \mid id$

Derivations $V = \{\}$, S.

$T = \{a, b\}$

P: $S \to aSb$ $\quad$ $S \to ab$

$S \to aSb \to aaSbb \to aaabbb$

$aaabbb$ is a string generated from G.

$L = \{ a^n b^n \mid n \geq 1 \}$

$A \to \beta \quad \beta \in (V \cup T)^*$ Production

$\alpha A \gamma \to \alpha \beta \gamma$ one step derivation

$\alpha, \gamma \in (V \cup T)^*$

$\alpha A \gamma \xrightarrow{*} \alpha \beta \gamma$ in zero or more steps

$aSb \xrightarrow{*} a^4 S b^4$

G, $L(G) = \{ x \mid x \in T^*, S \xrightarrow{*} x \}$

Language generated by the grammar G

L, G

L is generated by G if $L(G) = L$.

G_1 and G_2 are equivalent

if $L(G_1) = L(G_2)$

G_1: $T=\{a,b\}$, S, $V=\{\}$

$P=\{S \rightarrow aSb \mid ab\}$

$L(G_1) = \{a^n b^n \mid n \geq 1\}$

Proof: $a^{n-1} S b^{n-1}$

$S=ab$ $\quad a^n b^n$

1: $\rightarrow S \rightarrow aSb$ $\quad \alpha \in (V \cup T)^*$

2: $S \rightarrow ab$ $\quad |x|+2$

Apply 1 production some number of times and then use 2nd production if you want to stop $a^n b^n$.

G_1:

$V = \{S, A, B\}$ $\quad T = \{a, b\}$

$P = \{ S \rightarrow aB \mid bA,$

$A \rightarrow aS \mid bAA \mid a,$

$B \rightarrow bS \mid aBB \mid b \}$

Claim: $L(G_1) = \{x \in \{a,b\}^* :$ number of a in x is same as number of b in $x\}$

abbbAA ← right most

abbbAa

↓ abbbAA → abbbaA

abbbaa ← left most

$S \rightarrow aB \rightarrow abS \rightarrow abbA$

$\rightarrow abbbAA \rightarrow abbbaa$

abbbaa ← sentential form $\in (V \cup T)^*$

A derivation a production is applied to the left most variable of every sentential form then the derivation is said to be left most.

$S \rightarrow aB \mid bA$

$A \rightarrow aS \mid bAA \mid a$

$B \rightarrow bS \mid aBB \mid b$

To Prove $S \xrightarrow{*} w \in \{a,b\}^*$ $\Leftrightarrow$ w has equal number of a's and b's.

use induction.

induction parameter Length of w : $|w|$

$A \xrightarrow{*} w \Leftrightarrow$ w has exactly one more a than b

$B \xrightarrow{*} w \Leftrightarrow$ w has exactly one more b than a.

$A \rightarrow bAA \rightarrow baa$

$A \rightarrow aS \rightarrow abA \xrightarrow{*} abbaa$

Base Case: $|w| = 1$

$A \rightarrow a$ $B \rightarrow b$

a b holds

Induction hypothesis: Assume the above statements are true $|w| \leq k-1$

Need to prove for $|w| = k$

$S \xrightarrow{*} w$

$S \rightarrow aB$ $S \rightarrow bA$

$S \rightarrow aB$ $w = a w_1$

By induction hypothesis $|w_1| = k-1$

$B \xrightarrow{*} w_1$ number of b's in w_1 is exactly one more than number of a's in w_1

if $S \xrightarrow{*} w$, then w contains equal number of a's and b's

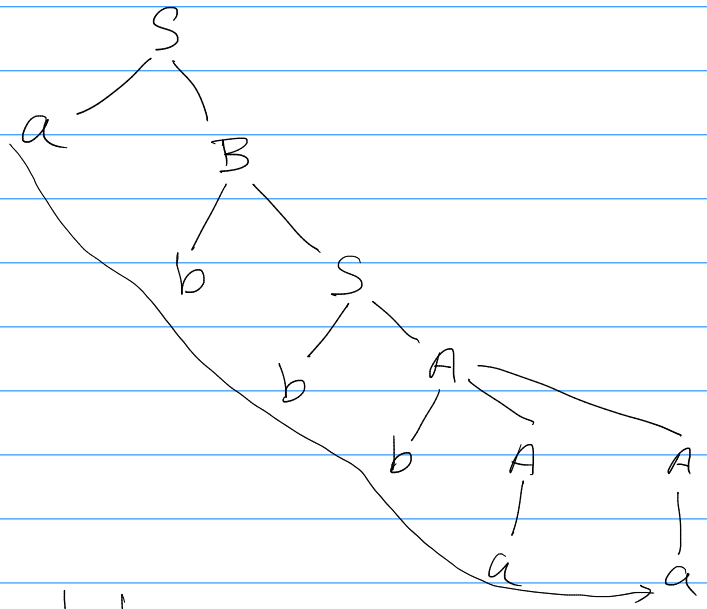
if w contains equal number of a's and b's then $S \xrightarrow{*} w$.

W.l.g. $w = a w_1$

$B \xrightarrow{*} w_1$

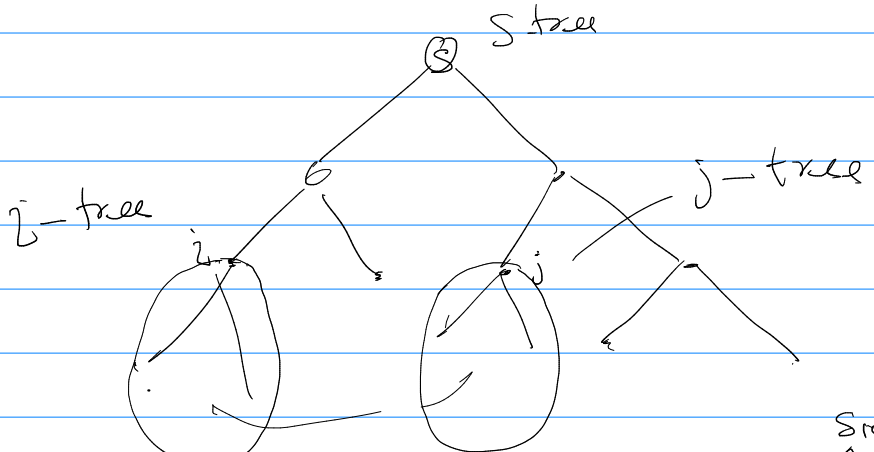
$S \xrightarrow{*} aB$.

Derivation tree



yield of above derivation tree.

a b b b a a



→ yield of i tree will be on left side of yield of j tree in the yield of S tree

$G = (V, T, P, S)$

$w \in L(G)$ derivation tree for w.

$S \xrightarrow{*} \alpha \iff \exists$ a derivation tree in G with yield α.

Sentential form

$G = (V, T, P, S)$ CFG

Vertex has a label from $V \cup T \cup \{\in\}$ (Leaf node)

root has label S.

If A is vertex which is interior then $A \in V$

If a vertex has label A and $X_1, X_2, \ldots X_k$ are sons of A in derivation tree then,

$A \longrightarrow X_1 X_2 \ldots X_k \in P$

P: $E \longrightarrow I \mid E+E \mid E*E$ $\quad V = \{E, I\}$

$I \longrightarrow \in \mid 0 \mid 1 \mid 2 \ldots \mid 9$ $\quad T = \{0, 1, 2, \ldots 9\}$

$3*2+5$

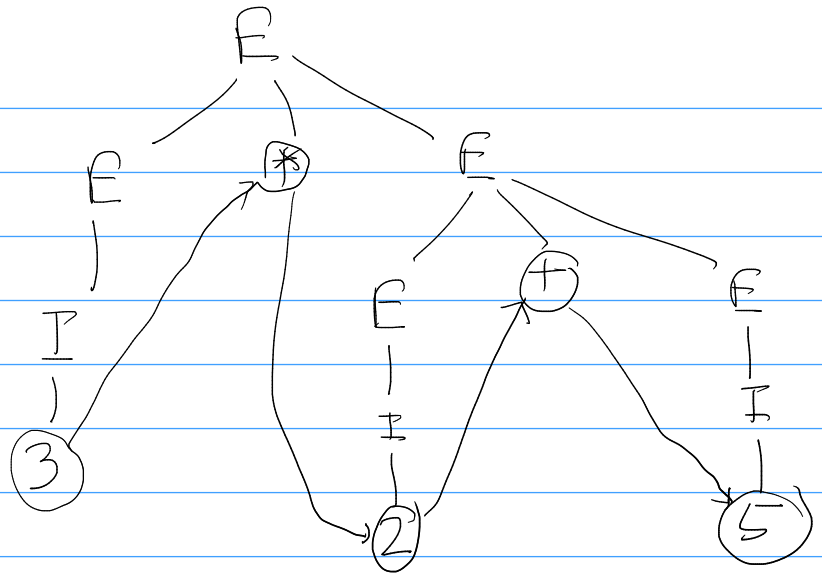
Derivation:

$E \Rightarrow E*E \Rightarrow E*E+E$

$\Rightarrow I*E+E \Rightarrow I*I+E$

$\Rightarrow I*I+I \overset{*}{\Rightarrow} 3\times 2+5$

3*2+5



3*2+5

Left most derivation

$E \Rightarrow E*E \Rightarrow I*E \Rightarrow 3*E$

$\Rightarrow 3*E+E \overset{*}{\Rightarrow} 3*2+E \Rightarrow 3*2+5$

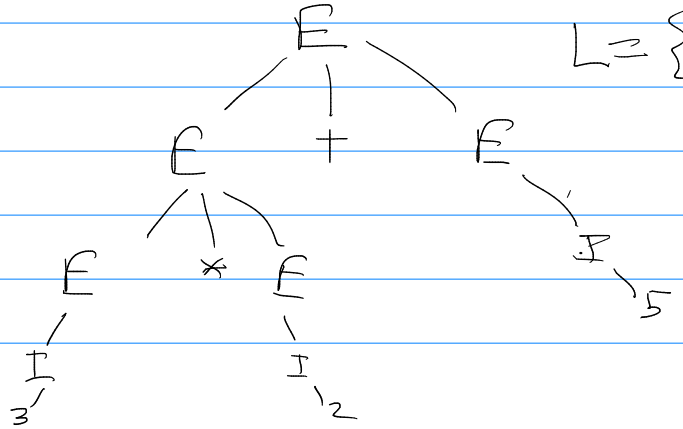
Ambiguous Grammars

A grammar G, is ambiguous if we have more than one parse tree for a string in L(G).

more than one left most or " " " right most derivation.

3*2+5 = 11

= 21



$L = \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

$\cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$

Simplification of CFG

1. Useless symbols.
2. Remove $\in$ Productions
3. " Unit productions

Useless Symbols:

Usefull Symbols: V'

$V \setminus V'$ set of useless symbols.

$T \setminus T'$

$G_1 = (V, T, P, S)$

$A \in V$ $\boxed{A} \xrightarrow{*} w \in T^*$ usefull.

$P: \begin{cases} S \to AB \mid a \\ A \to a \end{cases}$

$\boxed{B \xrightarrow{*} w \in L \subseteq \{a\}^*}$ use

$A \xrightarrow{*} a \in L \subseteq \{a\}^*$ useful

$A \in V$ does not generate string the we call it non generating symbols.

$X \in V \cup T$

$S \xrightarrow{*} \alpha X \beta \quad \alpha, \beta \in (V \cup T)^*$

↳ use full.

Start symbol.

Otherwise it is use less.

$A \to a\beta$

Use less symbol we name it non reacha — ble symbol.

$G = (V, T, P, S)$

$V' = \{ A \mid A \to \omega \in T^* \}$

$\forall \exists (A') \longrightarrow X_1 X_2 \cdots X_R$

$\forall i \quad X_i \xrightarrow{*} \omega \in T^*$

$\Downarrow$

if $\forall i \; X_i \in V'$ then $A' \in V'$

$V \setminus V'$ Set of useless symbols

non reachable symbols

reachable symbols $V' = \{S\}$

if $A \in V'$

$T' =$ and $A \to \alpha_1 \mid \alpha_2 \cdots \mid \alpha_n$

$\alpha_i \in (V \cup T)^*)$

then add all variable from α_i in V'

" " terminals from α_i in T'.

$V \setminus V'$, $T \setminus T'$ Set of non reachable symbol

$S \to AB \mid a$

$A \to a$

$L(G) = \{ \omega : S \xrightarrow{*} \omega \}$

1. non generating symbols. X will be removed
2. non reachable "

$S \not\to \alpha X \beta$

$X \xrightarrow{*} \omega$

Usefull $\{S, A\}$
Useless $\{B\}$

$S \longrightarrow a$
$A \longrightarrow a$
$S \xrightarrow{*} A$ No
$\{A\}$ useless : non reachable from S.

$S \longrightarrow a$ Simplified grammar

$S \longrightarrow A^{\checkmark} B^{\checkmark} \mid a_{\checkmark}$
$A \longrightarrow a^{\checkmark}$ $S \longrightarrow AB$
↓
$S \longrightarrow a$
$A \longrightarrow a$ } Not OK

Removal of ϵ Production

$S \longrightarrow ABaC$
$A \longrightarrow BC$
$B \longrightarrow b \mid \epsilon$
$C \longrightarrow D \mid \epsilon$
$D \longrightarrow d$

$V = \{S, A, B, C, D\}$
$T = \{a, b, d\}$

Collect all variables X such that
$X \xrightarrow{*} \epsilon$

$\{A, B, C\}$ nullable variables
$B \to \epsilon$, $C \to \epsilon$
$A \to BC \to \epsilon$

$A \rightarrow X_1 X_2 \cdots X_n$

$X_{i_1}, X_{i_2} \cdots X_{i_m}$, $m \leq n$

$\{i_1, i_2 \cdots i_m\} \subseteq \{1, 2, \cdots n\}$ are nullable variables.

add productions

$A \rightarrow X_{j_1} X_{j_2} \cdots X_{j_k}$

$\{j_1, j_2 \cdots j_k\} \subseteq \{1, 2, \cdots n\} \setminus N$

$N \subseteq \{i_1, i_2, \cdots i_m\}$

$S \rightarrow ABaC$

$A \xrightarrow{*} \epsilon$, $B \xrightarrow{*} \epsilon$ $C \xrightarrow{*} \epsilon$

$\{A, B, C\} =$

$\{A\}, \{B\}, \{C\}$

$\{A, B\}, \{B, C\}, \{A, C\}$

$\{A, B, C\}$

$S \rightarrow BaC$	$\{A\}$
$S \rightarrow AaC$	$\{B\}$
$S \rightarrow ABa$	$\{C\}$
$S \rightarrow aC$	$\{A, B\}$
$S \rightarrow Ba$	$\{A, C\}$
$S \rightarrow Aa$	$\{B, C\}$
$S \rightarrow a$	$\{A, B, C\}$

$S \rightarrow ABaC \mid BaC \mid ABa \mid AaC \mid aC \mid Ba \mid Aa \mid a$

$S \longrightarrow ABC$ if A, B, C are all nullable.

then do not include $S \longrightarrow \in$

* $A \longrightarrow BC$

$A \longrightarrow BC \mid B \mid C$

$B \longrightarrow b$

$C \longrightarrow D$

$D \longrightarrow d$

Removal Unit Production

$G_1 = (V, T, P, S)$

$A, B \in V$

$A \longrightarrow B$ Unit production

Collect all variables $X \in V$ such that

$X \xrightarrow{*} Y \in V$

$S \longrightarrow Aa \mid B$

$B \longrightarrow A \mid bb$

$A \longrightarrow a \mid bc \mid B$

$\{S, B, A\}$

$S \xrightarrow{*} B$

$B \xrightarrow{*} A$

$A \xrightarrow{*} B$

$S \xrightarrow{*} A$ because $S \to B \Rightarrow A$

non unit production

$S \longrightarrow Aa$

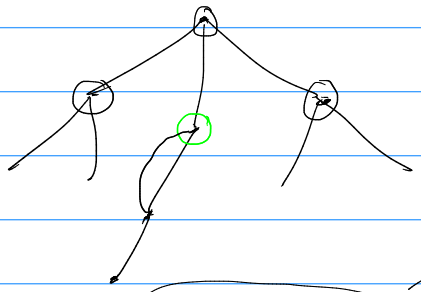
$B \longrightarrow bb$

$A \longrightarrow a / bc$

add

$A \longrightarrow a / bc / bb$ because $A \xrightarrow{*} B$

$B \longrightarrow bb / a / bc$ bcz $B \xrightarrow{*} A$



because $S \xrightarrow{*} A$, $S \xrightarrow{*} B$

$S \longrightarrow a / bc / bb \mid Aa$

Safe Order

1. Remove $\in$ Production
2. " unit "
3. " useless "

Chomsky Normal Form (CNF)

$G_1 = (V, T, P, S)$

~~$A \rightarrow BCD$~~

~~$A \rightarrow aAa$~~

$A \longrightarrow BC$ or $A \longrightarrow a$

$B, C, D \in V$ $A, B, C \in V$ $a \in T$

Assumption ∈ – Production

$S \longrightarrow bA \mid aB$

$A \longrightarrow bAA \mid aS \mid a$

$B \longrightarrow aBB \mid bS \mid b$

$A \longrightarrow a$, $B \longrightarrow b$

~~$S \longrightarrow$~~ bA $\quad C_b \longrightarrow b$ ✓

Add $S \longrightarrow C_b A$ ✓

~~$S \longrightarrow$~~ aB $\quad C_a \longrightarrow a$ ✓

$S \longrightarrow aB$

$S \longrightarrow C_a B$ ✓

$\{ S \longrightarrow C_b A \mid C_a B$

$C_b \longrightarrow b$, $C_a \longrightarrow a \}$

$A \longrightarrow bAA$

$A \longrightarrow C_b AA$

$\{ D_{aa} \longrightarrow AA$

$A \longrightarrow C_b D_{aa}$

$A \longrightarrow aS$

$A \longrightarrow C_a S$ ✓

$B \longrightarrow aBB$

$B \longrightarrow C_a BB$

$\{ D_{bb} \longrightarrow BB$, $B \longrightarrow C_a D_{bb} \}$

$A \longrightarrow X_1 X_2 - - X_k$ $\quad X_i \in V$

$X_1 \; B_1$

$B_1 \longrightarrow X_2 X_3 - - X_k$

$B_1 \longrightarrow X_2 B_2$

$B_2 \longrightarrow X_3 - - X_k$

Greibach Normal Form

Lemma: $G_1 = (V, T, P, S)$

$(A \longrightarrow \alpha_1 B \alpha_2) \in P$

$B \longrightarrow \beta_1 \mid \beta_2 \mid \cdots \mid \beta_r \qquad \alpha_i, \beta_j \in (V \cup T)^*$

add: $A \longrightarrow \alpha_1 \beta_1 \alpha_2 \mid \alpha_1 \beta_2 \alpha_2 \mid - - \mid \alpha_1 \beta_r \alpha_2$

remove $A \rightarrow \alpha_1 B \alpha_2$

G_1'

$L(G_1) = L(G_1')$

Lemma: $G_1 = (V, T, P, S)$

Left recursion $A \longrightarrow A\alpha_1 \mid A\alpha_2 \mid - - \mid A\alpha_r \qquad \alpha_i \in (V \cup T)^*$

$A \longrightarrow \beta_1 \mid \beta_2 \mid - - - \mid \beta_s$

in β_i Prefix of length 1 is not A.

$A \longrightarrow \beta_1 B \mid \beta_2 B \mid - - \mid \beta_s B$

$A \longrightarrow \beta_1 \mid \beta_2 \mid \cdots \mid \beta_s$

$$B \longrightarrow \alpha_1 / \alpha_2 - - \; / \alpha_r$$

$$B \longrightarrow \alpha_1 B / \alpha_2 B / - - / \alpha_r B$$

G_1' Consists of above productions will accept the same Language which is accepted by G_1.

$$A \longrightarrow A\alpha_{i_1} \longrightarrow A\alpha_{i_2}\alpha_{i_1}$$

$$\longrightarrow \beta_i \alpha_{i_k}\alpha_{i_{k-1}} - \alpha_{i_2}\alpha_{i_1}$$

$$A \longrightarrow \beta_i B$$

$$\longrightarrow \beta_i \alpha_{i_k}\alpha_{i_{k-1}} - \alpha_{i_2}\alpha_{i_1}$$

GNF

$$A \longrightarrow a\alpha \qquad \alpha \in V^*$$

$$G_2 = (\{S, A, B\}, \{a, b\}, P, S)$$

$$S \longrightarrow AB$$

$$A \longrightarrow BS / b$$

$$B \longrightarrow SA / a$$

Step 1: $S \equiv A_1$, $A \equiv A_2$, $B \equiv A_3$

$$A_1 \longrightarrow A_2A_3 \quad \checkmark$$

$$A_2 \longrightarrow A_3A_1 / b \quad \checkmark$$

$$A_3 \longrightarrow A_1A_2 / a$$

Step 2: $A_i \longrightarrow A_j \gamma$ $\quad \gamma \in V^*$ $\quad j > i$

$A_3 \longrightarrow A_1 A_2$ remain

Add $A_3 \longrightarrow A_2 A_3 A_2 \mid a$ remain

Add $A_3 \longrightarrow A_3 A_1 A_3 A_2 \mid b A_3 A_2 \mid a$

remain Left recursion

$A_3 \longrightarrow A_3 A_1 A_3 A_2 \mid \cdots$ (A_3 = α_1)

$A_3 \longrightarrow b A_3 A_2 \mid a$ ($b A_3 A_2$ = β_1, a = β_2)

introduce B_3

$$\begin{cases} A_3 \longrightarrow b A_3 A_2 B_3 \mid a B_3 \\ B_3 \longrightarrow A_1 A_3 A_2 \\ B_3 \longrightarrow A_1 A_3 A_2 B_3 \end{cases}$$

$A_1 \longrightarrow A_2 A_3$

$A_2 \longrightarrow A_3 A_1$

Since A_3 in GNF we can get

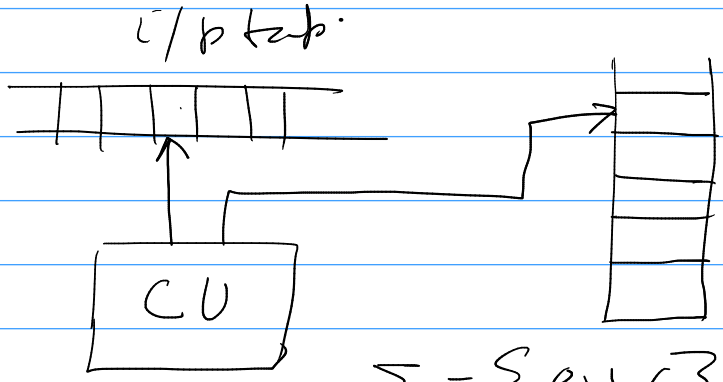
A_2 in GNF $\Rightarrow A_1$ in GNF

$\Rightarrow B_3$ in GNF

Push down automata

$$L = \{ w c w^R \mid w \in \{0,1\}^* \}$$

Not regular.



$\Sigma = \{0, 1, c\}$ $\quad \Gamma = \{B, G\}$

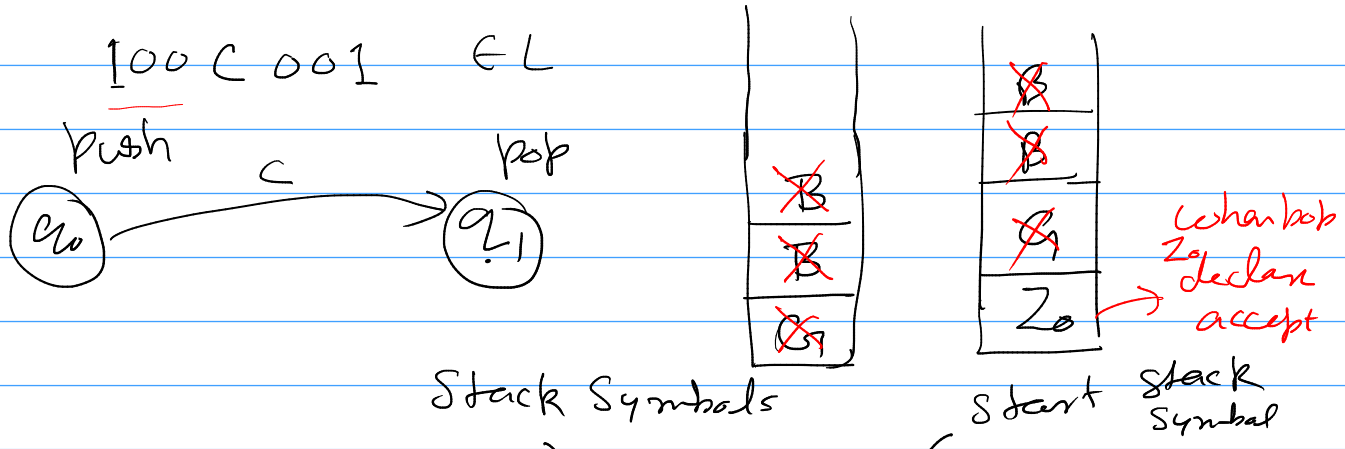
$w \in \{0,1\}^*$

$w c w^R$

q_0 { 0 — Push B in stack
1 — Push G in "
c — do not do anything change the state, reach q_1

q_1 { 0 — B on top of stack then delete B.
1 — G on top of stack then delete G.

finally Empty stack if wcw^R



PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

Two definitions of acceptance

→ Based on final states $L(M)$

→ Empty Stack. $N(M)$

$L(M) = N(M)$ w.r.t. PDA

$F \subseteq Q$

Non deterministm

$\delta : Q \times \Sigma \cup \{\epsilon\} \times \Gamma \longrightarrow P(Q \times \Gamma^*)$

Top of stack

$(q_0, 0, B) \longrightarrow (q_0, BB)$

Top Symbol

$\delta(q_0, 0, B) = \{(q_0, BB), (q_0, RB)\}$

Non deterministic PDA

PDA

Deterministic PDA

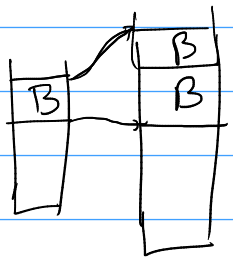
DPDA

$$L = \{ wcw^R \mid w \in \{0,1\}^* \}$$

$$M = (\{q_0, q_1, q_2\}, \{0, 1, c\}, \{R, B, G\},$$

q_0 q_1 q_2 δ, q_0, R, ϕ)

q_1 push q_2 pop



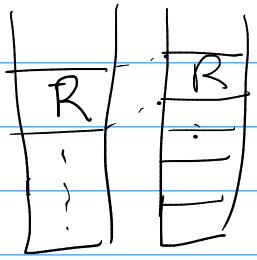
$$\delta(q_0, \epsilon, \epsilon) = (q_1, R)$$

Push

$$\begin{cases} \delta(q_1, 0, R) = \{(q_1, BR)\} & \text{at } q_2 \\ \delta(q_1, 0, B) = \{(q_1, BB)\} & \text{write transition} \\ \delta(q_1, 0, G) = \{(q_1, BG)\} & \text{for 1, use G} \end{cases}$$

$\{R, B, G\}$

$$\delta(q_1, c, Z) = \{(q_2, Z)\}$$



$$\delta(q_2, 0, B) = \{(q_2, \epsilon)\}$$

$$\delta(q_2, 1, G) = \{(q_2, \epsilon)\}$$

$$\delta(q_2, \epsilon, R) = \{(q_2, \epsilon)\}$$ accept

acceptance based on empty stack.

I-D: Instantaneous description

$$(q, aw, Z\alpha) \vdash_M (p, w, \beta\alpha)$$

q: present state; aw: input string; $Z\alpha$: stack content, Z on top; p: next state; $\beta\alpha$: Z replaced with β

$(p, \beta) \in \delta(q, a, Z)$

Exp:

$(q_1, 011, GGR)$

$\vdash (q_1, 11, BGGR)$ --

- - - - - - -

$\vdash^* (q_2, \epsilon, \epsilon)$ accept

Let M be a PDA.

$L(M)$ Language accepted by final state

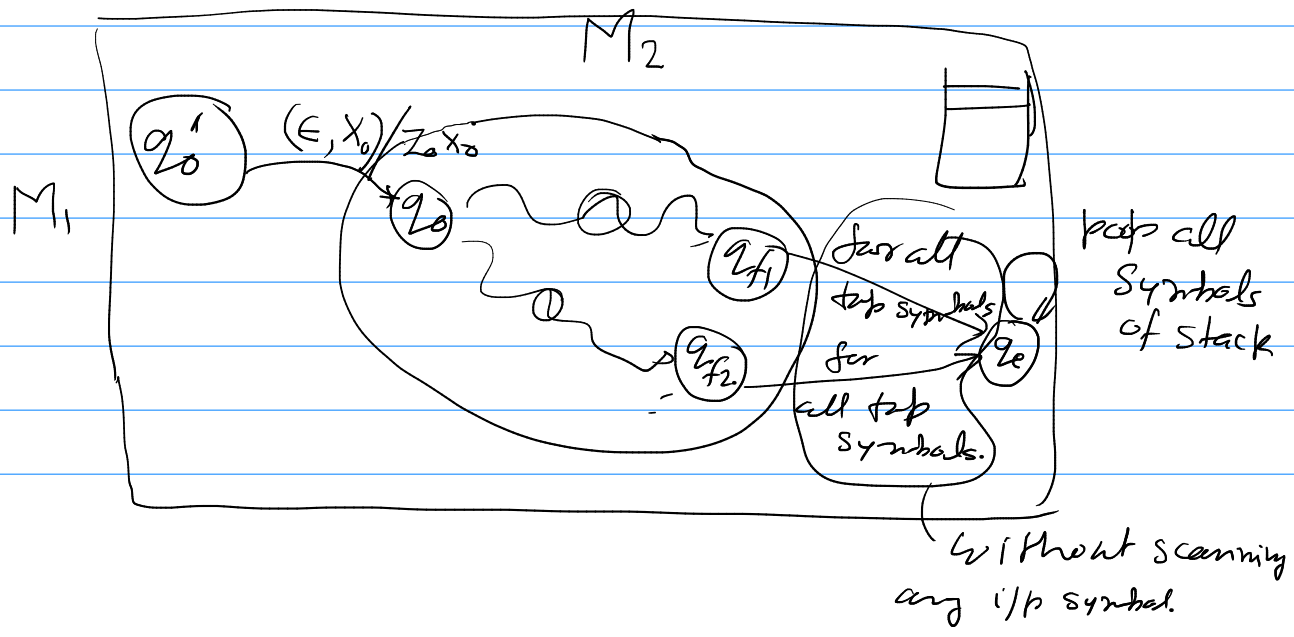
$\{w \mid (q_0, w, Z_0) \vdash^* (p, \epsilon, \gamma)$ for some $p \in F\}$

$N(M)$ Language accepted by empty stack.

$\{w \mid (q_0, w, Z_0) \vdash^* (p, \epsilon, \epsilon)$ for some $p \in Q\}$

Thm $L = L(M_2)$ for some PDA $M_2 \Rightarrow$

$L = N(M_1)$ for some PDA M_1



Thm $L = N(M_1)$ for some PDA $M_1 \Rightarrow$

$L = L(M_2)$ for some PDA M_2.

Add $q_f \in F$

whenever stack becomes empty and we have scanned the whole i/p string make transition to q_f.

Deterministic PDA

$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

1. $\forall q \in Q, \forall Z \in \Gamma, \forall a \in \Sigma \cup \{\epsilon\}$

$|\delta(q, a, Z)| \leq 1$

2 $\forall q \in Q \quad \forall Z \in \Gamma$

if $\delta(q, \epsilon, Z)$ is non empty

then $\delta(q, a, Z)$ must be empty $\forall a \in \Sigma$.

DPDA D

$L(M)$ $N(M)$

$L = \{ w w^R \mid w \in \{0,1\}^* \} \neq \{ w c w^R \mid w \in \{0,1\}^* \}$

L will be accepted by PDA but not by DPDA.

Equivalence of CFG and PDA

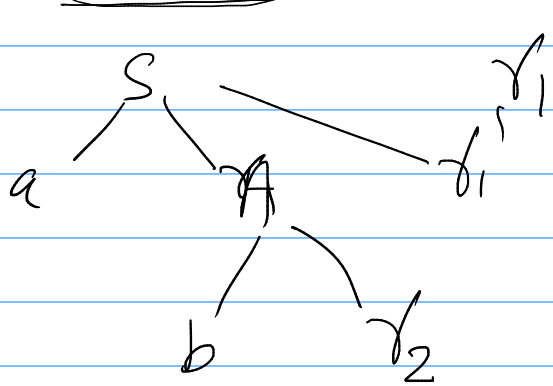
1. Given CFG G, construct a PDA which accepts L(G)

2. Given PDA M, construct a CFG G L(G) = N(M)

CFG G. which is in GNF, without any unreachable symbol.

$$\begin{cases} S \to a\gamma_1 \\ A \to b\gamma_2 \end{cases} \qquad b, a \in T \quad \gamma_2\gamma_1 \in V^*$$

abba $\qquad S \xrightarrow{*} abbq$?



$$\delta(q, a, A) = \{(q, \gamma)\}$$

when $A \to a\gamma$ is a production

$a \in T$

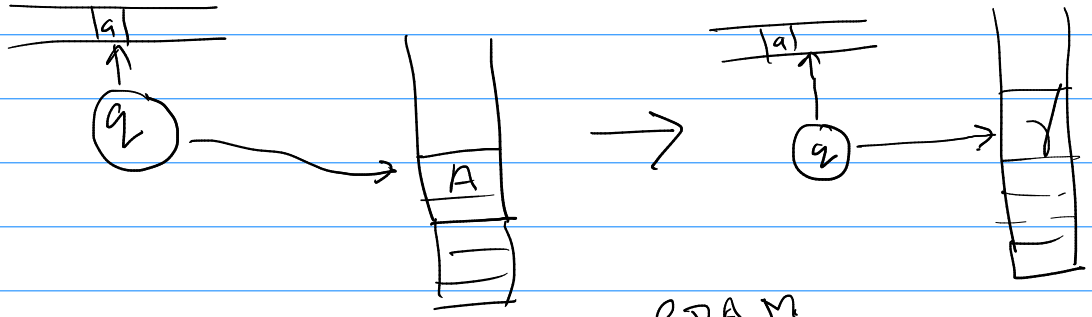
$\gamma \in V^*$

CFG $G \Rightarrow$ PDA. M $\quad L(G) = L(M)$

$G = (V, T, P, S)$ $\quad$ G in CNF without nullable symbols

$A \longrightarrow a\gamma$ $\quad \gamma \in V^*$, $a \in T$

PDA M: $\delta(q, a, A) = \{(q, \gamma)\}$



Left most derivation $\quad$ PDA M

$S \overset{*}{\Rightarrow} x\alpha \quad \Leftrightarrow \quad (q, x, S) \vdash^* (q, \epsilon, \alpha)$

$x \in T^*$, $\alpha \in V^*$ — CFG G

($\Leftarrow$) S1: $(q, x, S) \vdash^i (q, \epsilon, \alpha) \Rightarrow S \overset{i}{\Rightarrow} x\alpha$

Apply induction on i: $\quad i = 0$

(q, ϵ, S)

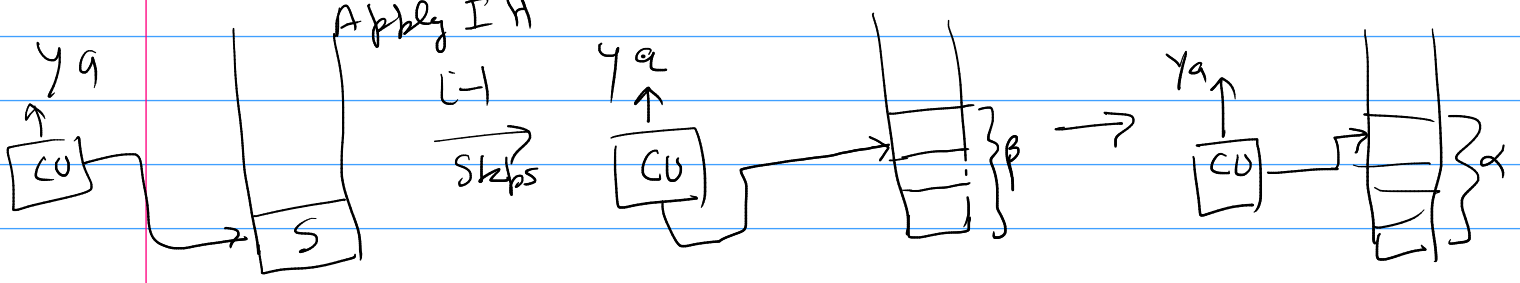
Assume that S1 is true for $\leq i-1$ steps.

Prove that S1 holds for $= i$

$x = ya$

apply IH

$$(q, ya, S) \overset{i-1}{\vdash} (q, a, \beta) \vdash (q, \epsilon, \alpha)$$



$$(q, ya, S) \overset{i-1}{\vdash} (q, a, \beta)$$

$$\Downarrow$$

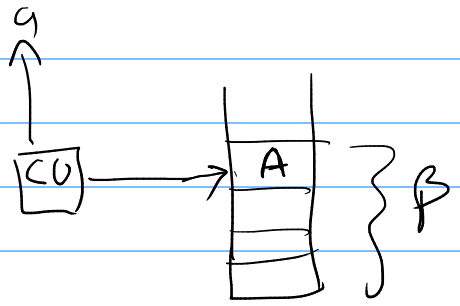
$$(q, y, S) \overset{i-1}{\vdash} (q, \epsilon, \beta)$$

Apply I.H.

$$S \overset{i-1}{\Longrightarrow} y\beta$$

$$(q, a, \beta) \vdash (q, \epsilon, \alpha)$$

$\beta \in V^*$



$\beta = A\gamma$

$$(q, a, A) \vdash (q, \epsilon, \alpha')$$

I.H.: $A \to a\alpha'$

$$(q, a, \underbrace{A\gamma}_{\beta}) \vdash (q, \epsilon, \underbrace{\alpha'\gamma}_{\alpha})$$

$S \xrightarrow{i-1} \gamma\beta = \gamma A \gamma \rightarrow \underbrace{\gamma a}_{*}\underbrace{\alpha'\gamma}_{\alpha} = x\alpha$

($\Rightarrow$) S2: $S \xrightarrow{*} x\alpha \iff (q, x, S) \vdash^{*} (q, \epsilon, \alpha)$ — Proved

Assume that S2: is true $\leq i-1$ steps.

Prove that S2 is true for i^{th} steps

Apply I.H.

$S \xrightarrow{i-1} \gamma A \gamma \rightarrow \gamma a \alpha' \gamma$

Apply IH — $\gamma a = x$ — $*$

$(q, \underbrace{x}_{\gamma a}, S) \vdash^{i-1} (q, a, A\gamma)$ — (1)

$\begin{cases} A \rightarrow a\alpha' \\ (q, a, A) \vdash (q, \epsilon, \alpha') \end{cases}$

$(q, a, A\gamma) \vdash (q, \epsilon, \underbrace{\alpha'\gamma}_{\alpha})$ — (2)

Combine (1) and (2)

$(q, x, S) \vdash^{i-1} (q, a, A\gamma) \vdash (q, \epsilon, \alpha)$

Given A

PDA M Construct CFG G

Such that $N(M) = L(G)$

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \phi)$$

Variables (qAp) $\forall\ q, p \in Q$ and $\forall\ A \in \Gamma$ of CFG G

Add production of the following form

1. $S \longrightarrow (q_0 Z_0 q)$ $\forall\ q \in Q$.

2. If $\delta(q, a, A)$ contains $(q_1, B_1 B_2 \cdots B_n)$ where $B_1 B_2 \cdots B_n \in \Gamma^*$

add

$(q A q_{n+1}) \longrightarrow a\,(q_1 B_1 q_2)(q_2 B_2 q_3)\cdots(q_n B_n q_{n+1})$

$\forall\ q_i \in Q$, $B_i \in \Gamma$

If $n = 0$ or $\delta(q, a, A) \ni (q_1, \epsilon)$ $a \in \Sigma \cup \{\epsilon\}$

$(q A q_1) \longrightarrow a$

$$M = (\{q_0, q_1\}, \{0, 1\}, \{X, Z_0\}, \delta, q_0, Z_0, \phi)$$

$$\delta(q_0, 0, Z_0) = \{(q_0, X Z_0)\}$$

$$V = \{q_0 X q_0, \ - \ - \ - \ - \ - \}$$

$$|V| = 2 \times 2 \times 2 = 8$$

S $\qquad g = |V|$

$$S \longrightarrow q_0 Z_0 q_0 \mid q_0 Z_0 q_1$$

$$\delta(q_0, 0, Z_0) = \{(q_0, X Z_0)\}$$

$$q_0 Z_0 q_0 \longrightarrow 0\,(q_0 X q_0)\,(q_0 Z_0 q_1) \mid 0\,(q_0 X q_1)\,(q_1 Z_0 q_0)$$

$$q_0 Z_0 q_1 \longrightarrow 0\,(q_0 X q_0)\,(q_0 Z_0 q_1) \mid 0\,(q_0 X q_1)\,(q_1 Z_0 q_0)$$

PDA M construct CFG G

$$L(G) = N(M)$$

S1: $\underbrace{(q\ A\ p) \overset{*}{\Rightarrow} x}_{G} \iff \underbrace{(q, x, A) \overset{*}{\vdash}_M (p, \epsilon, \epsilon)}_{M}$

Use induction to prove S1:

($\Leftarrow$) $(q, x, A) \overset{*}{\vdash}_M (p, \epsilon, \epsilon) \Rightarrow (qAp) \overset{*}{\Rightarrow} x$

induction parameter is number of steps in M

$$(q, x, A) \overset{i}{\vdash}_M (p, \epsilon, \epsilon) \Rightarrow (qAp) \overset{i}{\Rightarrow} x$$

Base case $i = 1$

$$(q, x, A) \vdash_M (p, \epsilon, \epsilon)$$

$(p, \epsilon) \in \delta(q, x, A)$ in M.

Apply construction of CFG.

$(q\ A\ p) \longrightarrow x$

Base case holds.

I.H. assume that S1 is true upto $i+1$ steps.

Now for i^{th} step.

$$(q, x, A) \overset{i}{\underset{M}{\vdash}} (p, \epsilon, \epsilon)$$

$$x = a Y_1 Y_2 \cdots Y_n \in \Sigma^*$$

$$(q, a Y_1 Y_2 \cdots Y_n, A) \vdash (q_1, Y_1 Y_2 \cdots Y_n, B_1 B_2 \cdots B_n) \overset{i-1}{\vdash} (p, \epsilon, \epsilon)$$

$$(q, a Y_1 Y_2 \cdots Y_n, A) \vdash (q_1, Y_1 Y_2 \cdots Y_n, B_1 B_2 \cdots B_n)$$

$(q_1, B_1 B_2 \cdots B_n) \in \delta(q, a, A)$

$(q\ A\ q_{n+1}) \longrightarrow a\,(q_1 B_1 q_2)(q_2 B_2 q_3) \cdots (q_n B_n q_{n+1})$

$$(q_1, Y_1 Y_2 \cdots Y_n, B_1 B_2 \cdots B_n) \overset{i-1}{\underset{M}{\vdash}} (p, \epsilon, \epsilon)$$

$Y_1 Y_2 \cdots Y_j$ $\quad$ $B_{j+1} \cdots B_n$ will not be effected.

$\exists\ q_2, q_3, \ldots, q_{n+1} = p$

$$(q_j, Y_j, B_j) \overset{*}{\vdash} (q_{j+1}, \epsilon, \epsilon) \quad \begin{matrix} Y_2 \cdots Y_n, \\ | \\ B_2 \cdots B_n \end{matrix} \quad 1 \le i \le n$$

$$(q_1,\ y_1 y_2 \cdots y_n,\ B_1 B_2 \cdots B_n) \overset{i-1}{\vdash} (p, \epsilon, \epsilon)$$

$$(q_1, y_1 y_2 \cdots y_n, B_1 \cdots B_n) \overset{*}{\vdash} (q_2, y_2 y_3 \cdots y_n, B_2 \cdots B_n)$$

$$\overset{*}{\vdash} (q_3, y_3 y_4 \cdots y_n, B_3 B_4 \cdots B_n) \vdash \cdots \vdash (p, \epsilon, \epsilon)$$

$$(q_1, y_1, B_1) \vdash (q_2, \epsilon, \epsilon)$$

$$(q_j, y_j, B_j) \overset{*}{\vdash} (q_{j+1}, \epsilon, \epsilon)$$

I.H.

$$(q_j\ B_j\ q_{j+1}) \overset{*}{\Longrightarrow} y_j \qquad 1 \le j \le n$$

p.

$$(q\ A\ q_{n+1}) \Rightarrow a\,(q_1 B_1 q_2)(q_2 B_2 q_3) \cdots (q_n B_n q_{n+1})$$

$$(q\ A\ p) \overset{*}{\Rightarrow} a\ y_1 y_2 \cdots y_n$$

Bound

($\Longleftarrow$) Reading assignment.

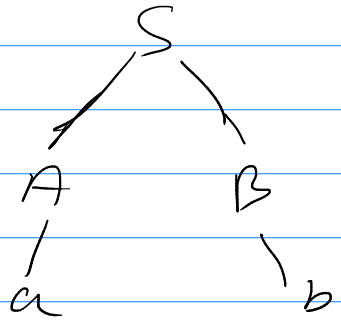
Pumping Lemma for CFL

Suppose CFG G in CNF and $L(G) = L \setminus \{\epsilon\}$

CNF $\begin{cases} A \to BC & B, C \in V \\ A \to a & a \in T \end{cases}$

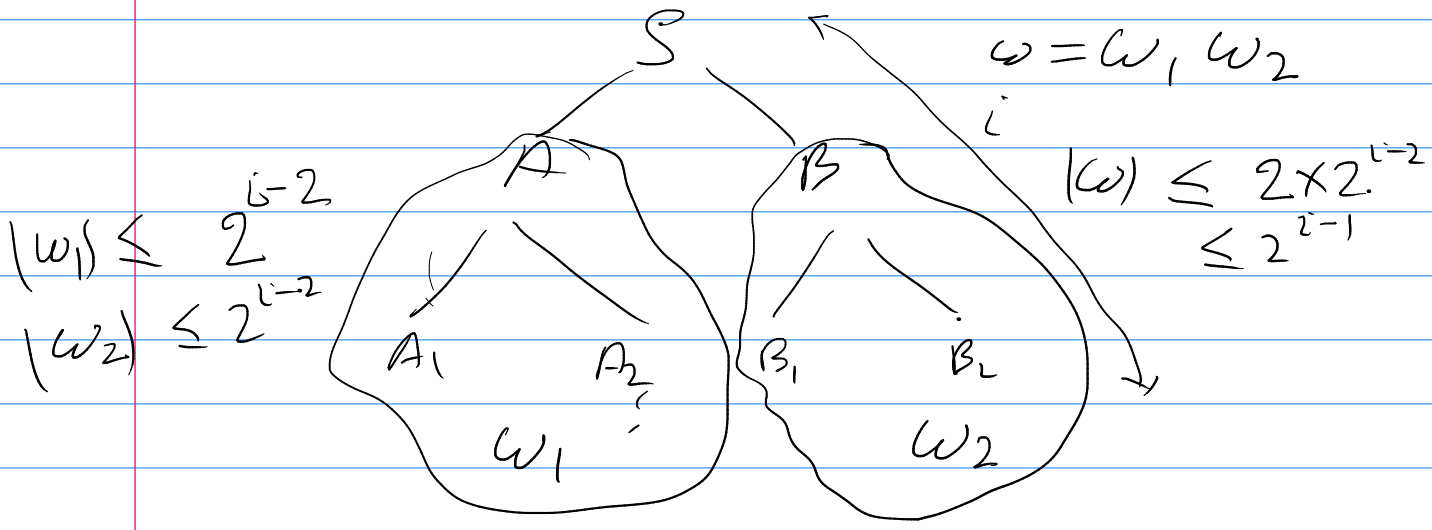
Suppose we have a parse tree which generate $w \in L(G)$ such that highest length of the path is R.

Then $|w| \leq 2^{R-1}$



2 = Length of the path.

$ab = 2^{2-1}$



$w = w_1 w_2$

$|w| \leq 2 \times 2^{i-2}$

$\leq 2^{i-1}$

$|w_1| \leq 2^{i-2}$

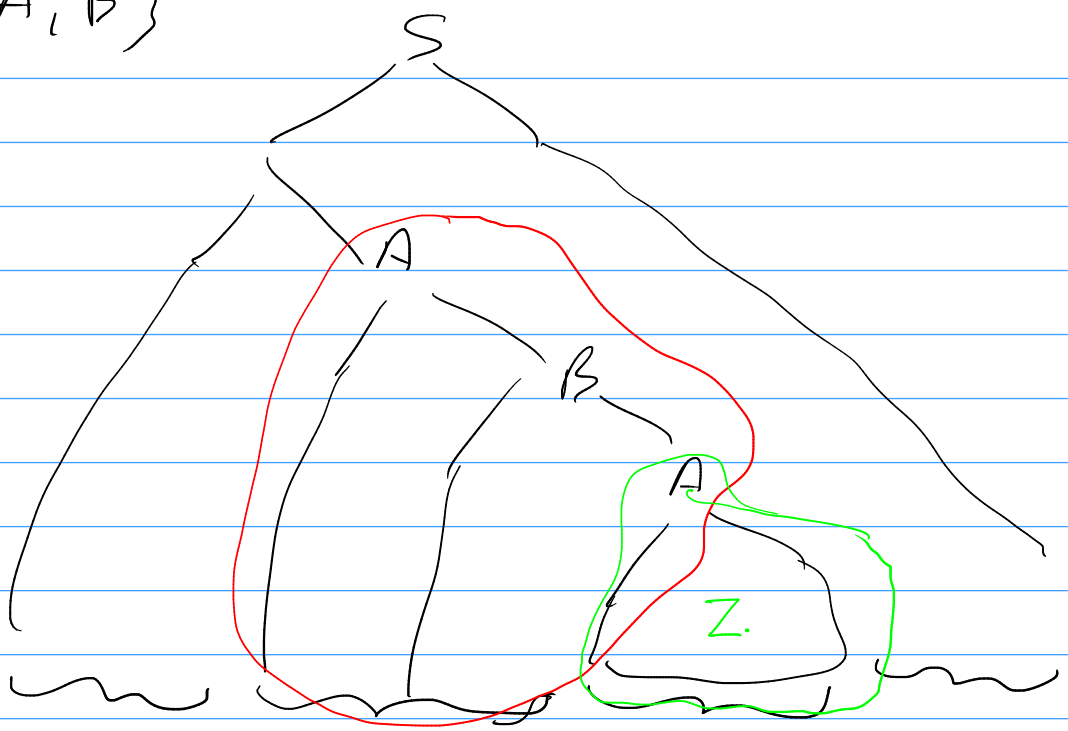
$|w_2| \leq 2^{i-2}$

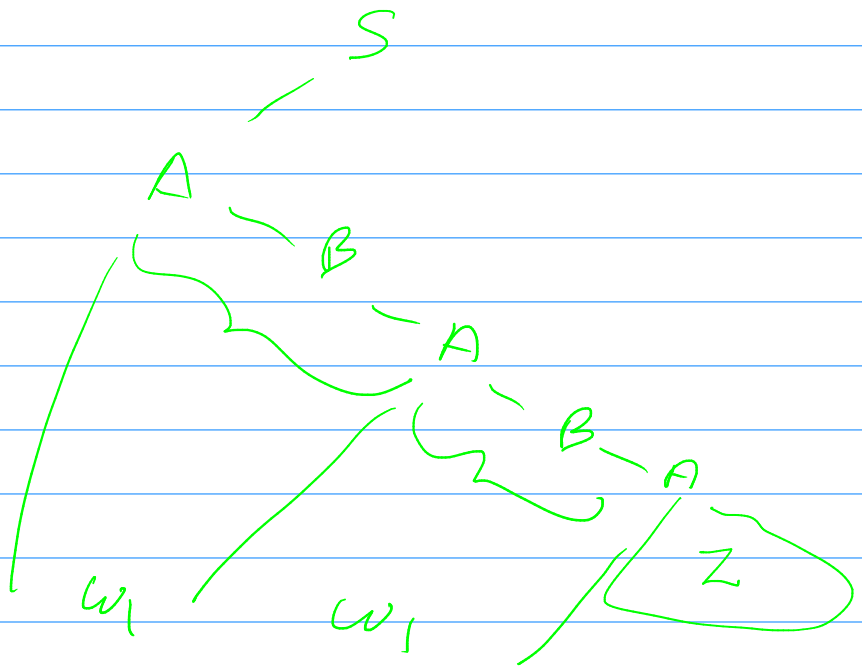
Suppose G has R variables.

Our base tree has path whose length is $\geq R+2$

Can we say that some variable has got repeated in the path of length $R+2$ or greater.

$\{S, A, B\}$

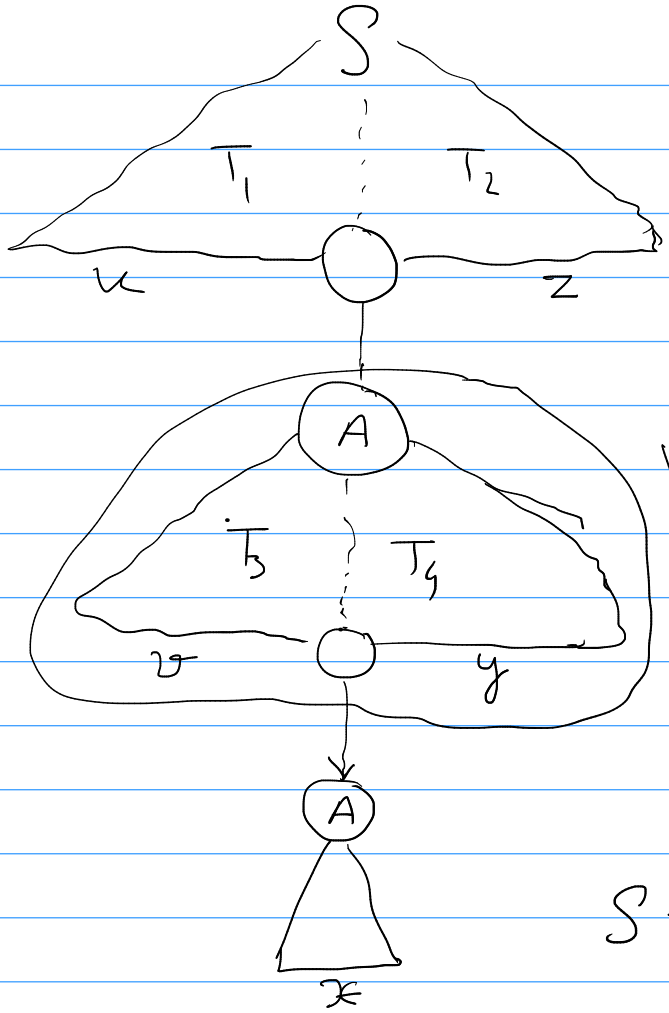




CFG $G = (V, T, P, S)$ $\quad |V| = k$.

$w \in L(G) \quad |w| \geq 2^k$

$\exists\ A \in V$ which has got repeated on the path of length $k+1$.



$uvxyz \in L(G)$

pump: $|vy| \geq 1$

$|vxy| \leq 2^k$

$uv^i x y^i z \in L(G)$

$\forall i \geq 0$

$S \rightarrow uAZ$

$A \rightarrow vAy$

$A \rightarrow x$

$S \rightarrow uAZ \rightarrow uvxyz$

$\hookrightarrow uvAyz \rightarrow uvvAyyz$

$S \rightarrow uAZ \rightarrow uxz$ $\qquad uv^2xy^2z$

$L = \{ a^n b^n c^n \mid n \geq 0 \}$ is not CFL.

pick m

$uvxyz = a^m b^m c^m \in L$

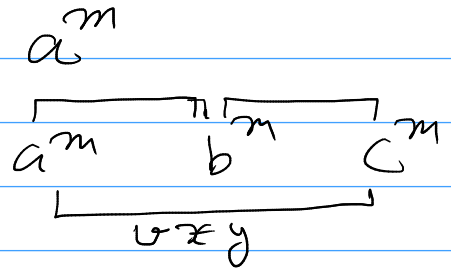
$|vy| \geq 1$

$|vxy| \leq m$

vxy must lie in a^m, b^m or c^m

$a^m b^m$ or $b^m c^m$

Since, $|vxy| \leq m$ vxy cannot contain b^m completely.



$a^m b^m c^m$

$uvxy = a^m$

$a^R b^m c^m$ $\quad R \neq m$

$a^R b^m c^m \notin L$

$uvxy = a^m b^p$ $\quad p < m$

$|u| < m$

$vxy = a^{m_1} b^{b_1}$

$y = b^{l_1}$ $\quad vx = a^{l_2}$

$v = a^{m_1} b^{b_1}$

a

$L = \{ ww \mid w \in \{a,1\}^* \}$ Not CFL

$L = \{ a^{n!} \mid n > 0 \}$ Not CFL.

Closure properties of CFL

CFL L_1, L_2

Prove that $L_1 \cup L_2$, $L_1 L_2$, L_1^* will be CFL.

L_1 $\quad G_1 = (V_1, T_1, P_1, S_1)$

L_2 $\quad G_2 = (V_1, T_1, P_2, S_2)$

$L_1 \cup L_2$ $\quad G_U = (V_1 \cup V_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \to S_1 | S_2\}, S)$

$L_1 L_2$ $\quad$ Add $\quad S \to S_1 S_2$

L_1^* $\quad S \to S_1 S | \epsilon$

Exp :- $L = \{ w \in \{a,b\}^* \mid$ number of a in w = number of b in w $\}$

$h(a) = 0^3 1^3$
$h(b) = 0110$

$S \to aSbS | bSaS | \epsilon$

$L_a = \{ 0^n 1^n \mid n \geq 0 \}$ $\quad S \to S_a S S_b S | S_b S S_a S | \epsilon$

$L_b = \{ w w^R \mid w \in \{0,2\}^* \}$

$S_a \to 0 S_a 1 | 01$

$S_b \to 0 S_b 0 | 2 S_b 2 | \epsilon$

$abab \in L$

$0^3 1^3 | 0220 | 0^2 1^2 | 022220$

Prove that CFL is closed w.r.t. substitution, and homomorphism.

(I), $L_1 \cap L_2$ — Not CFL

$\overline{L_1 \cup L_2} = L_1 \cap L_2$?

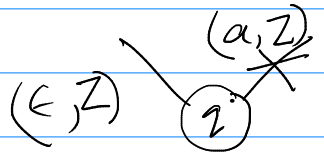
$L_1 = \{ a^i b^i c^i \mid i \geq 1 \}$ Not CFL

$L_2 = \{ a^i b^i c^j \mid i, j \geq 1 \}$ CFL ?

$L_3 = \{ a^j b^i c^i \mid i, j \geq 1 \}$ CFL ?

$L_1 = L_2 \cap L_3$?

DPDA if L is accepted by DPDA then $\overline{L}$ will also be accepted by DPDA.



abaabcabc

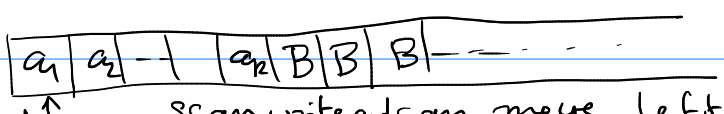
Reading Assignment

CYK Algorithm

CFL ∩ R = CFL

Turing Machine

infinite tape (Large Memory)



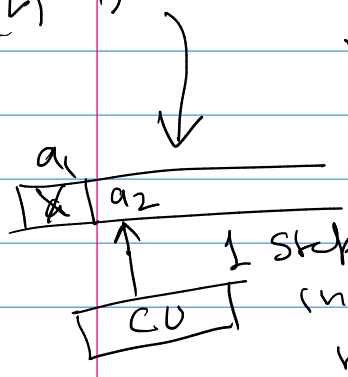
scan, write and can move left or right by one position

head

CU

control unit

$\delta(q_1, a_1) = (q_1, X, R)$



1 step right in black day has written X

Similar to our normal computing M/C.

Set of states — tape Symbols — initial state — Blank Symbols — Set of final states

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

Σ: i/p symbol. δ: transition function.

$$B \in \Gamma \qquad \Sigma \subseteq \Gamma$$

$$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$$

L left movement
R right movement.

I.D.

$$\alpha_1 q \alpha_2 \qquad \alpha_1, \alpha_2 \in (\Gamma \cup \Sigma)^*$$

$$q a_1 a_2 a_3 \cdots \vdash X q a_2 a_3 \cdots \vdash q_1 X a_2 a_3 \cdots$$

$$\delta(q, a_1) = (q, X, R)$$

$$\delta(q, a_2) = (q_1, a_2, L)$$

TM M

$$L(M) = \{ \omega \mid \omega \in \Sigma^* \quad q_0 \omega \vdash^* \alpha_1 p \alpha_2 \quad p \in F \wedge \alpha_1, \alpha_2 \in (\Sigma \cup \Gamma)^* \}$$

halt

$\omega \in L(M)$: ω is accepted by M if it halts at final state.

$w \notin L(M)$ if M on w either reach non final state and halt or it does not halt.

Find n such that n is odd and it is perfect.

A number n is perfect if $n = \sum_{d|n} d$

27 1013

Example Design a TM for $L = \{a^n b^n \mid n \geq 1\}$

aaabbbB

Xaa Y...

XXa YY

XXX YY

XXXYYYB

a X

b Y

Left most a

a a b

Enter final state

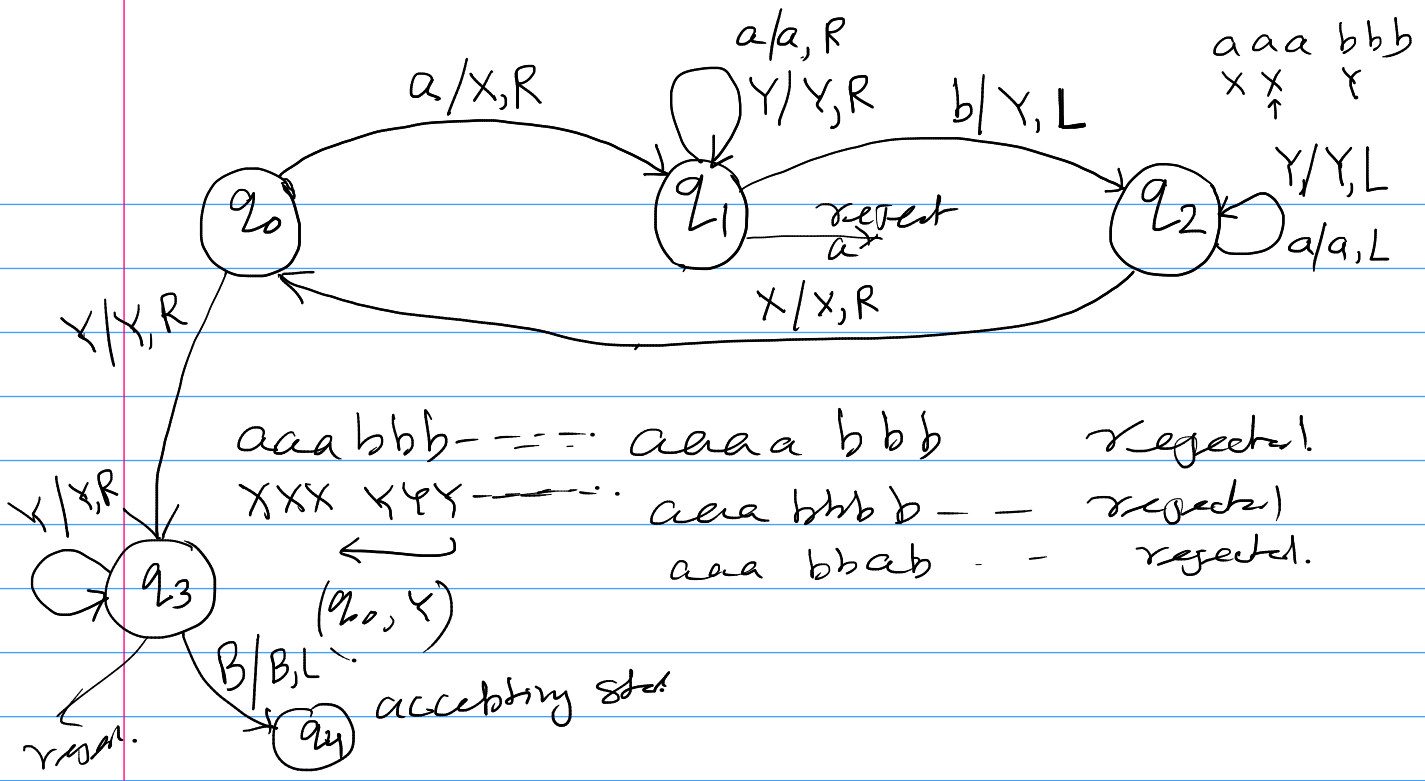
$\Leftarrow\Rightarrow$ B, a, b reject or reach non final state

aaabbb

X → Y

X ← → Y

X ← → Y



A Language accepted by TM, is called recursively enumerable Language.

A Language accepted by TM which halts on all i/ps called recursive Language.

$$\begin{cases} w \in L(M)^{TM} & \text{accept and halt} \\ w \notin L(M)^{TM} & \text{reject " "} \end{cases}$$

Turing Machine which computes a function

$$f(m,n) = \begin{cases} m-n & m \geq n \\ 0 & \text{otherwise.} \end{cases}$$

i/p $0^m 1 0^n$

o/p $\begin{cases} 0^{m-n} & \text{if } m \geq n \\ B^\infty & \text{otherwise.} \end{cases}$

0000 1000

X Y

X Y

X Y

X R

XXX 0 1 YYY

BBB (0) B B B $\quad f(4,3) = 1$

$f(i_1, i_2, i_3, \ldots i_k) = m.$ defined for all i/p

TM i/p $0^{i_1} 1 0^{i_2} \ldots 1 0^{i_k}$ such function we call it total recursive

o/p 0^m

halt

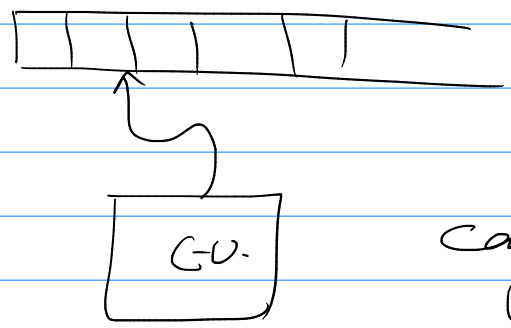
recursive Language

A function $f(i_1, i_2, \ldots i_k)$ is computed by TM is called partial recursive function.

recursive enumerable Language.

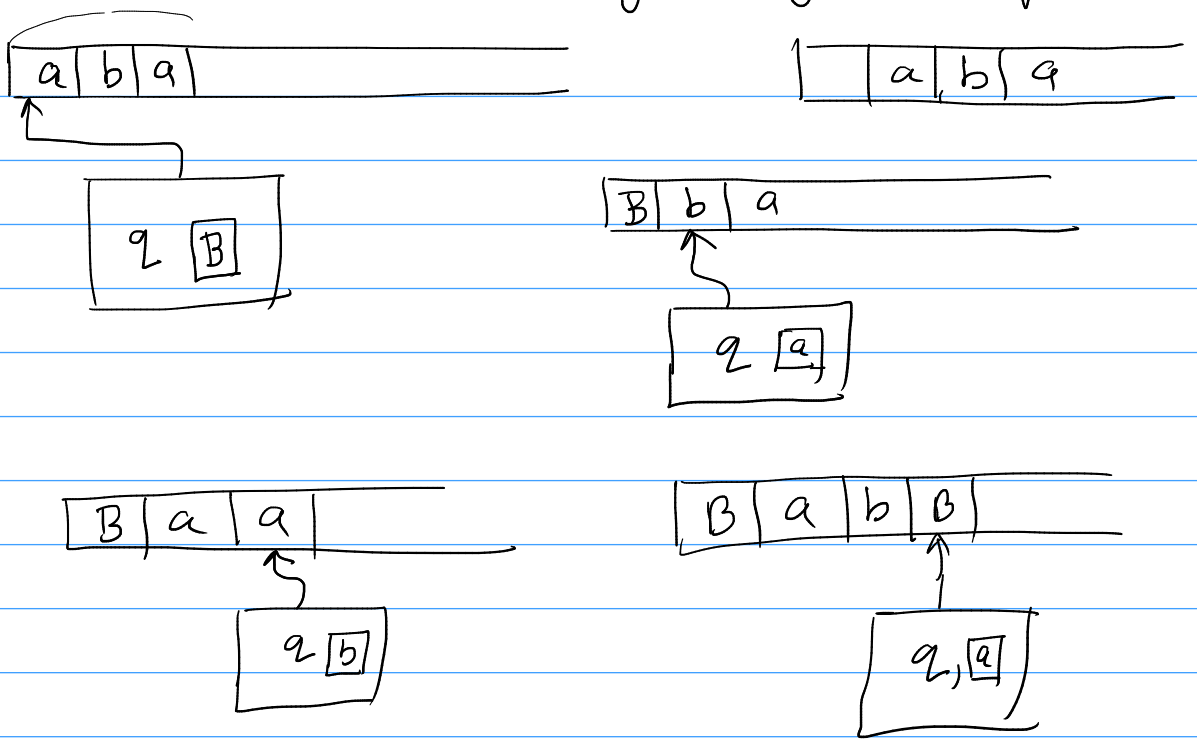
$L = \{ 0^{i_1} 1 0^{i_2} \ldots 0^{i_k} 1 1 0^m \}$

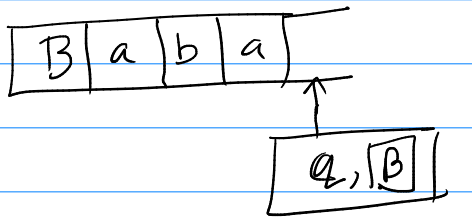
Techniques for designing T.M.



Can store some finite length string.

right by one position





accepted

$Q = \{q\}$ Γ

$Q' = Q \times \Gamma$

Construct a TM which computes the following function

$f(m,n) = mn$

i/p $0^m 1 0^n$ in tape

o/p 0^{mn} in tape.

i/p 000100

0001001

B001001 00

BB01 00 10000

BBB1001000000

BBBBBBB000000 o/p

B 00 1 q_0 00 1

B 00 1 X 0 1 0

B 00 1 X X 1 00

B 00 1 00 100

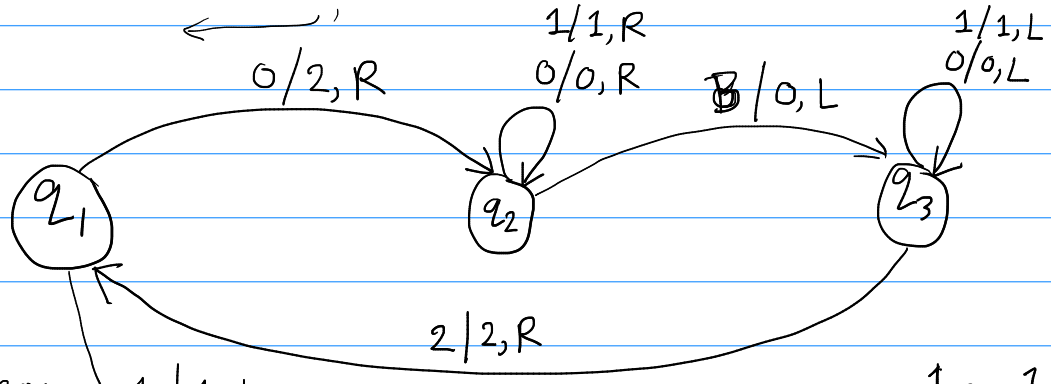
Copy Procedure

I.D. I.D.

1: $q_0 0^m 1 0^n \vdash^* B 0^{m-1} 1 q_1 0^n 1$ ✓ Copy

2: $0^m 1 q_1 0^n 1 0^i \vdash^* 0^m 1 q_5 0^n 1 0^{n+i}$

3: $B^i 0^{m-i} 1 q_5 0^n 1 0^{ni} \vdash^* B^{i+1} 0^{m-i-1} 1 q_1 0^n 1 0^{ni}$ ✓

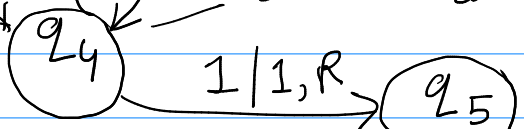


00 1 2 q_4 0 000

1 2 2 q_4 1

1 2 q_4 2 1

1/1,L 2/0,L 2 erased with 0



000 1 00 1

000 q_4 00 1

000 1 2 0 1 0 ← Copy has finished

00 0 1 2 2 q_4 1 00

B 00 1 q_5 00 1 00

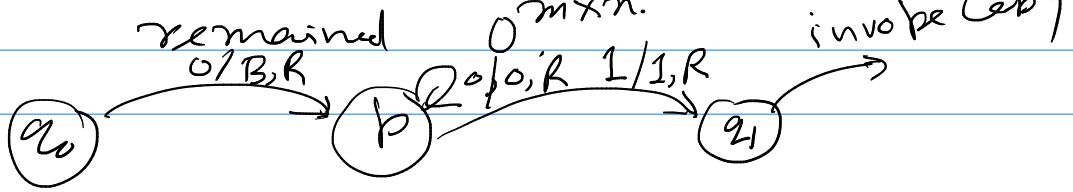
Our goal is to TM

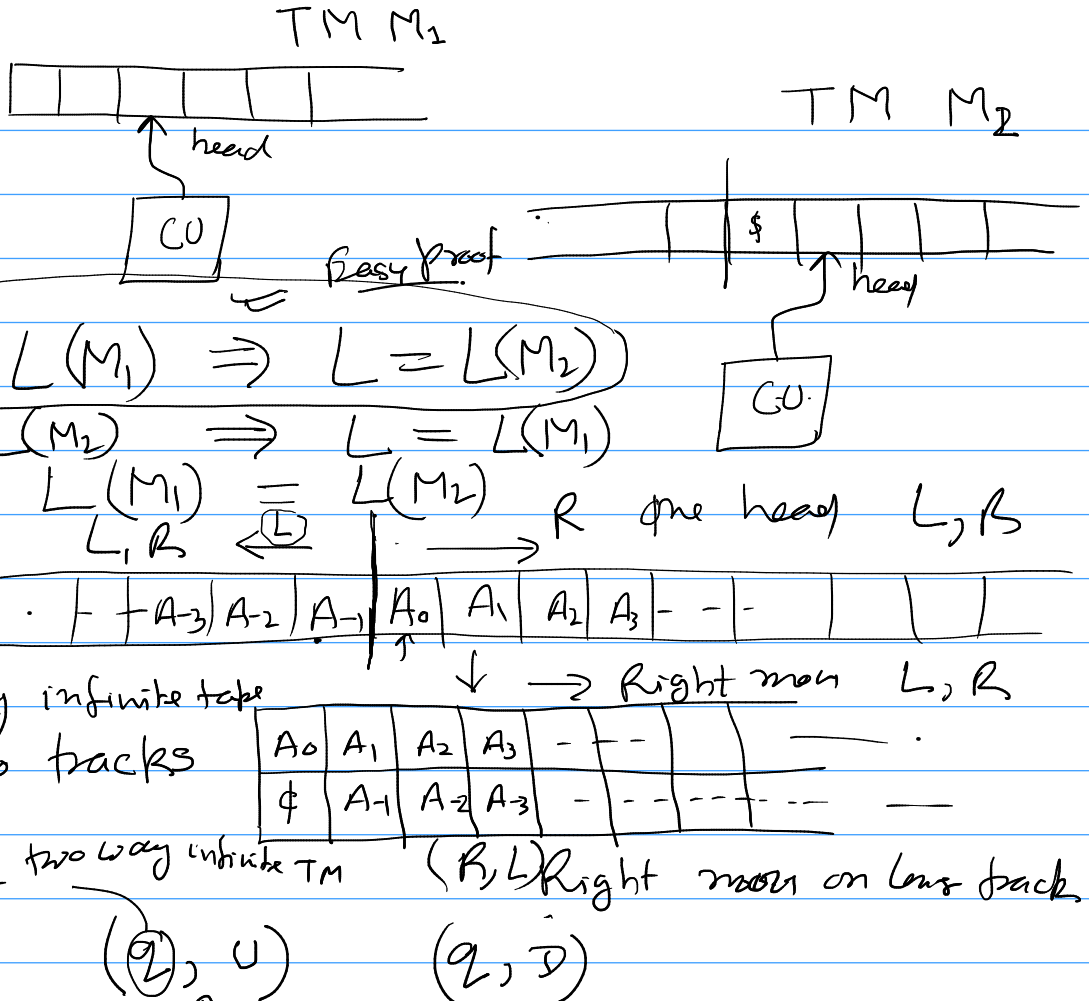
B B 0 1 q_1 00 1 00

invoke Copy

invoke m/c for 2 and 3 repeatedly

when $i = m$, Erase $10^n 1$.





Two way infinite tape TM

$M_2 = (Q_2, \Sigma_2, \Gamma_2, \delta_2, q_2, F_2, B)$

Two track one sided infinite TM

$M_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_1, F_1, B)$

q_1	q_2
B	B

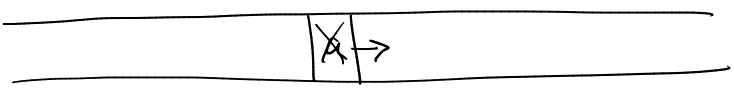
$Q_1 = Q_2 \times \{U, D\}$

$\Gamma_1 = \Gamma_2 \times \Gamma_2$

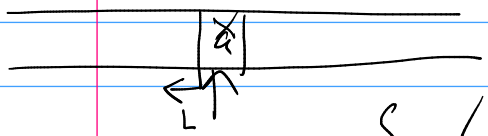
$\Sigma_1 = \Sigma_2 \times (\Sigma_2 \cup \{B, ¢\})$

$F_1 = \{(q, U), (q, D) \mid q \in F_2\}$

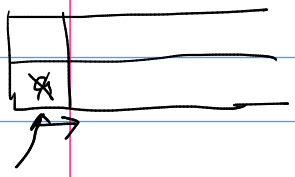
$M_2 : \quad \delta_2(q_2, a) = (q, X, R)$



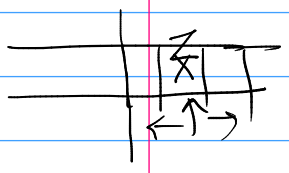
$$\delta_1(q_1, [a, B]) = ((q, U), [X, B], R)$$



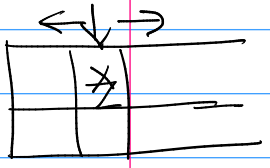
$$\delta_2(q_2, a) = (q, X, L)$$



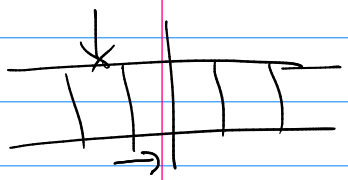
$$\delta_1(q_1, [a, B]) = ((q, D), [X, \$], R)$$



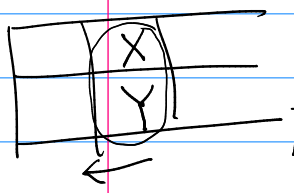
$$\delta_2(q, X) = (P, Z, A) \quad \leftarrow \{L, R\}$$



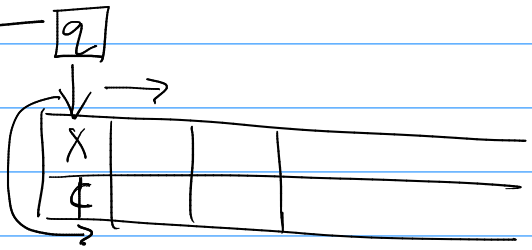
$$\delta_1((q, U), [X, Y]) = ((P, U), [Z, Y], A)$$



$$\delta_2(q, X) = (P, Z, A) \quad L \; R$$



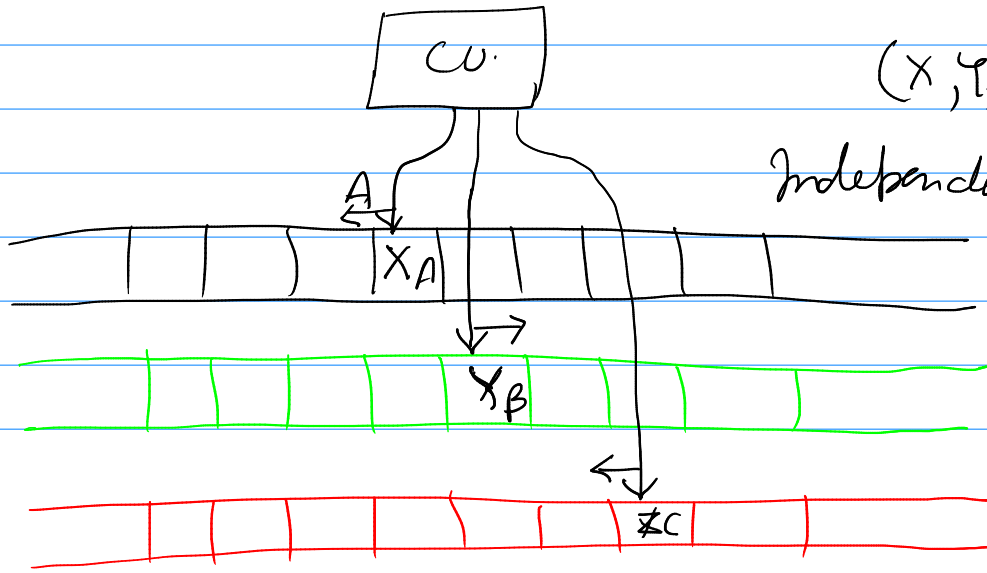
$$\delta_1((q, D), [X, Y]) = ((P, D), [X, Z], \begin{matrix} L \\ R \\ A \end{matrix})$$



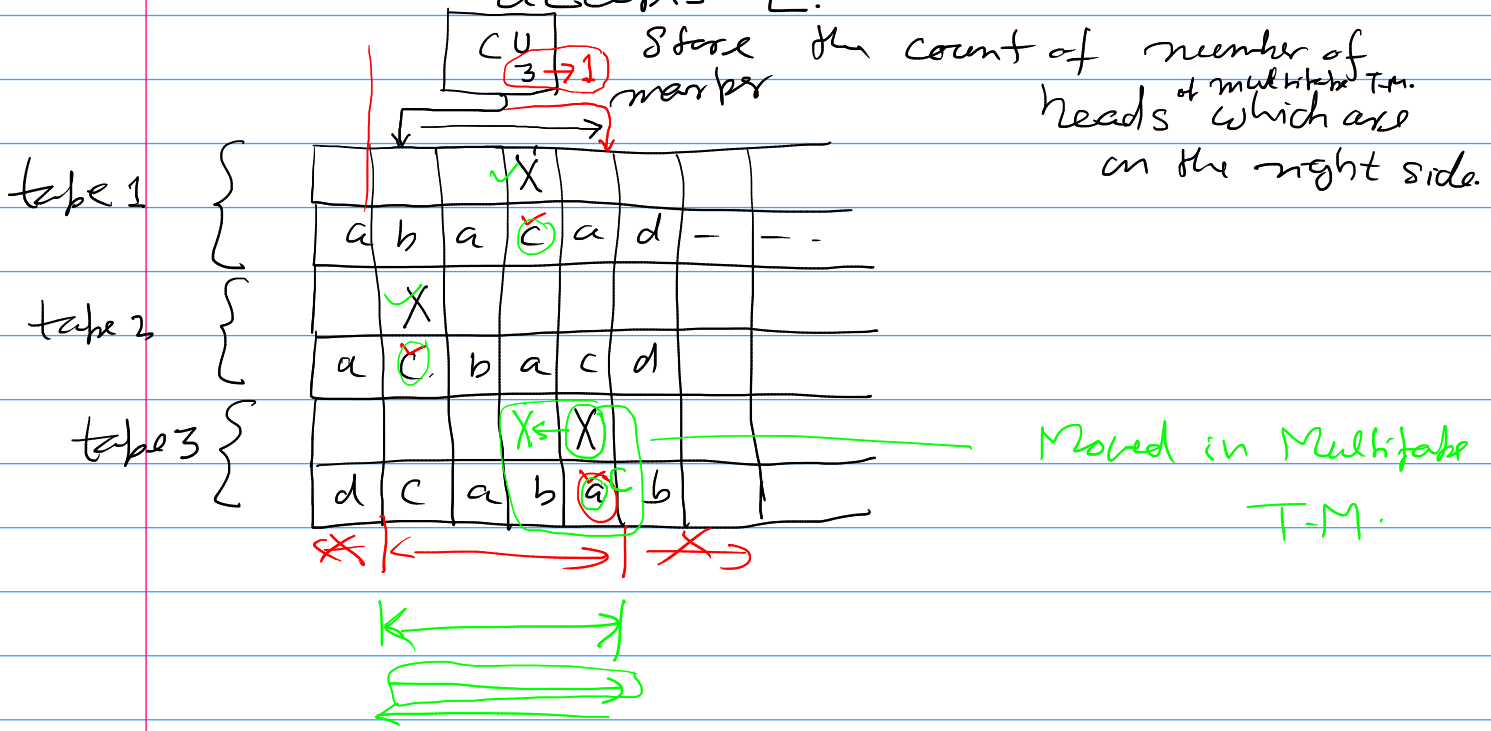
$$\delta_2(q, X) = (P, Y, A)$$

( )

Multitape Turing Machine

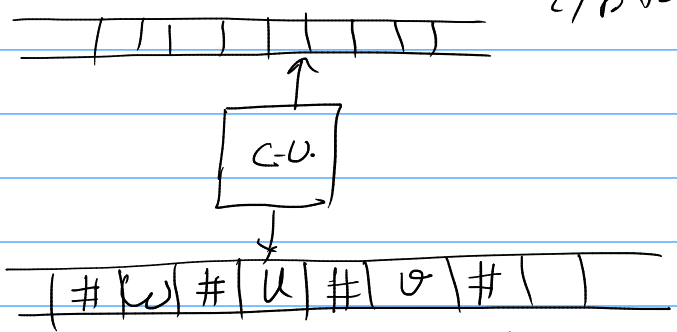


Thm If L is accepted by Multitape TM then $\exists$ a Standard T.M. which accepts L.



Turing Machine as Enumerator

Generator Machine.



Modification in o/p tape is not allowed, and head position of o/p tape will always move right.

generator machine

Thm $L = G_1(M_1) \Leftrightarrow \exists$ TM M_2 $L = L(M_2)$

$G_1(M_1) = L = \{w, u, v, \dots\}$

$L = G(M_1) \Rightarrow \exists$ TM M_2 $L = L(M_2)$

M_1 will generate $w \in L$

M_2 i/p $w \in L$

1. run M_1 generate x
2. if $(x == w)$ accept
   otherwise go to step 1

L is r.e. $\Rightarrow \exists M$ such that $L = G(M)$

TM M_1 which accepts L.

Σ input alphabet

Machine which will generate words of Σ^*?

$\Sigma = \{0, 1\}$

Σ^*

$\epsilon, 0, 1, 00, 011, 10, 11, 0x,$
$1x$ — — — —

generate word in L

M which generates words from Σ^*...
w_1, w_2, w_3 — — —

$M_1(w_i)$ — accepts write in o/p tape.
— otherwise again run M

Enters in loop and step for other program

generate a pair $(i, j) \in |\mathbb{N}| \times |\mathbb{N}|$

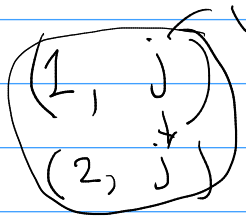
run M which generate words from Σ^*

w_i

run M_1 on w_i for j steps.

if M_1 accepts write o/p

otherwise go for next pair (i, j)



w_1 w_2 w_3 ...

$i=1$ $(1,1), (1,2), (1,3), \ldots$

$i+j = n \in |\mathbb{N}|$

$i+j = 3$ $(1\ 2)$ ✓ $(2,1)$ ✓

$i+j = 4$ $(1,3), (2,2), (3,1)$

$w_1, w_2, w_3 \ldots$

0^{i+j} $0^2 1 0^2$

1 3

Reading Assignment

recursion $\Leftrightarrow$ generating word

canonical order.

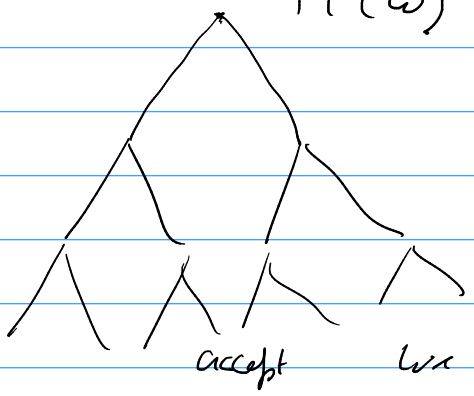
$\Sigma = \{0, 1\}$

$0, 1, 00, 01, 10, 11 \ldots$

Non deterministic TM:

$$|\delta(q,a)| \geq 1$$

$$\delta(q,a) = \{(q', b, L), (q, a, R)\}$$



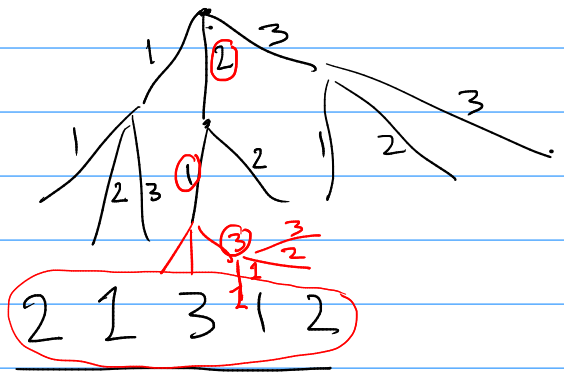
if leaf nodes end at reject state
reject ω.

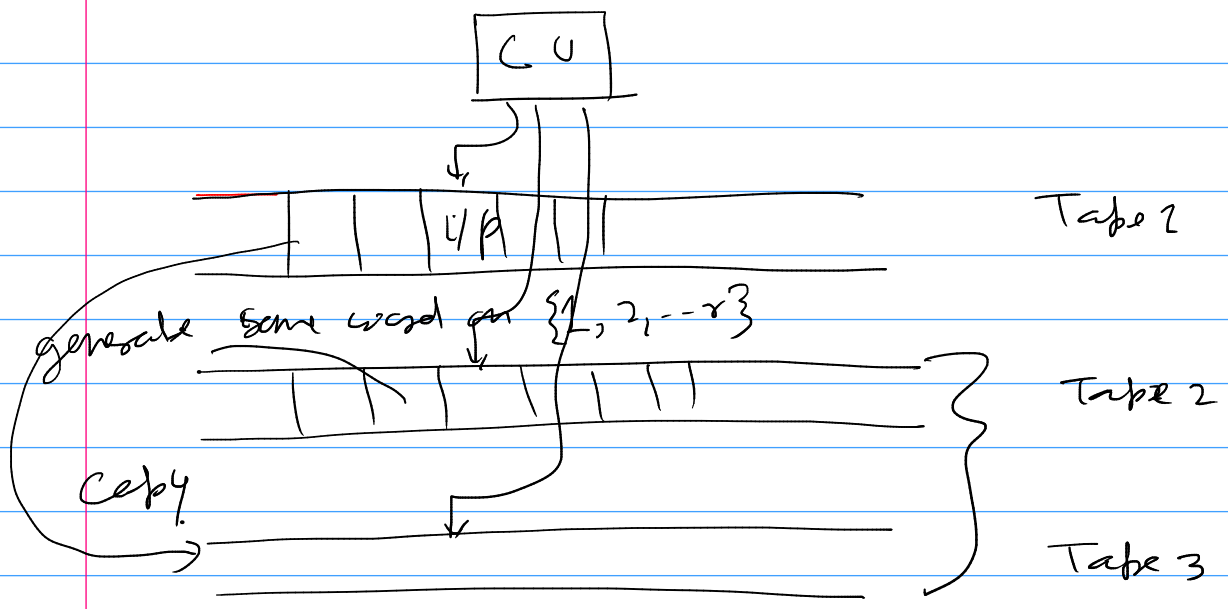
NTM = DTM: w.r.t. Power.

Thm $L = L(NTM) \Rightarrow \exists DTM \quad L = L(DTM)$

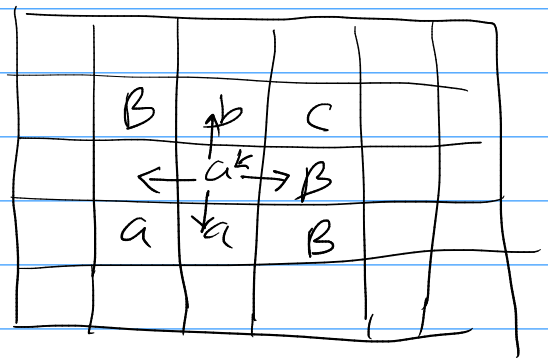
$\omega \in L$

$$|\delta(q,a)| \leq r = 3$$





Multidimensional T.M: Reading Assignment.



Same as standard T.M.

Decidability:

Problem (x, y, z, i)

$$x^i + y^i = z^i \quad i \geq 3, \quad x, y, i, z \in \mathbb{N}$$

Does the above equation possess a solution?

↑ Yes, No version of the above problem.

Language.

$$L = \{ (x, y, z, i) \in \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \mid x^i + y^i = z^i, i \geq 3 \}$$

$w \in L$?

CFG: Is a given CFG ambiguous.?

$G_1 = (V_1, T_1, P_1, S_1)$ $\leq$ Yes / No

Solvable / decidable: Then I an algorithm such that when it takes G_1 as input then it tells whether G_1 is ambiguous or not.

G_1 is an instance of the above problem.

$L = \{ G_1, G_2, \text{- - - - -} \}$ (Yes instances of above problem)

TM M(halts on all i/p)

$M(G)$ — accepts if CFG is ambiguous (CFG.)

reject otherwise

we say that problem is decidable. (or Solvable)

L is recursive

otherwise not recursive

Reduction: NP-Completeness

AMB: Is a given CFG ambiguous?

FIND: For a given CFG, Produce a word with two or more parse trees if it exist, otherwise, output No.

(1.) AMB $\Rightarrow$ FIND

(2.) FIND $\Rightarrow$ AMB.

FIND(G) {

2. Run AMB(G) — Yes → write new lines which generate such a w.

No → No

}

AMB(G) {

Run FIND(G) — w → Yes

No → No

}

P = { Set of all languages for which ∃ a deterministic polynomial time TM }

NP = { Set of all languages for which ∃ a non deterministic polynomial time scanning machine }

$NP \stackrel{?}{=} P$

Hilbert: ICM — Field Medal

Can we write a program which generates proof for a given mathematical statement?

dream

His ^ has got breaked.

Godel's: { has proved that } a true mathematical statement in the context of $(|\mathbb{N}|, +, \times, \{\text{logical Symbols}\})$ which has no proof. }

$(|\mathbb{N}|, +, \{\text{logical Symbols}\})$

decidable

{ It is not possible to prove the consistency of a mathematical system from within the " "

$0 = 1$ inconsistent

Properties of Recursive and R.E. Languages

L is recursive $\Rightarrow$ $\overline{L}$ is recursive.

L_1 is recursive
L_2 " "
$\Rightarrow$ $L_1 \cup L_2$ is recursive

L_1 M_1
L_2 M_2

$L_1 \cup L_2$ M

$M(w) = \{$ $M_1(w)$ — accept → acc.
reject — $M_2(w)$ acc / rej.

L_1 is R.E M_1

L_2 is R.E M_2

$\downarrow$

$L_1 \cup L_2$ is R.E. M.

$L_1 \cup L_2$

$M(\omega) = \{ (M_1(\omega), M_2(\omega))$

$M_1(\omega)$ — accept — accept

No accept

$M_2(\omega)$ — accept — accept

Not accept.

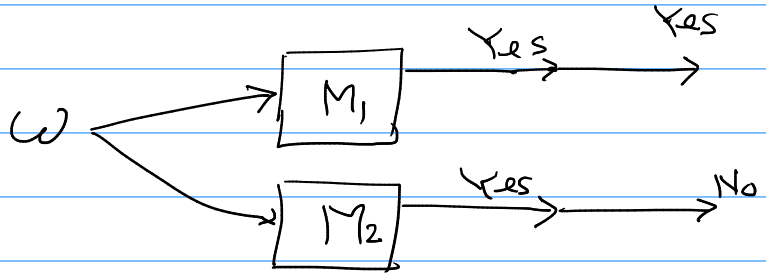
Both run basically.

Thm if L and $\bar{L}$ are both r.e.

then L is recursive.

L M_1

$\bar{L}$ M_2



L is recursive.

flip the o/p then we get a halting TM which accepts $\bar{L}$.

Encoding of TM

initial state, final state

$$M = (Q, \Sigma, \delta, \Gamma, q_1, q_2)$$

Thm: TM M can be represented as a TM M' such that $\Sigma' = \{0, 1\}$

$\Gamma' = \{0, 1, B\}$.

$0 = X_1'$ $\quad L = D_1'$

$1 = X_2'$ $\quad R = D_2'$

$B = X_3'$

$Q' = \{q_1', q_2', \ldots q_n'\}$ (initial state, final state)

$$\delta(q_i', X_j') = (q_k', X_l', D_m')$$

$Code_1 = 0^i 1 0^j 1 0^k 1 0^l 1 0^m$ = Code of one transition function

$Code_2 =$ ⋮

$Code_3 =$

$\langle M' \rangle = 111\ Code_1\ 11\ Code_2\ 11 \cdots 111$

$x \in \{0, 1\}^*$

(whether it represent encoding of a TM.

$\langle M, w \rangle = \langle M \rangle w$

Construct a Language which will not be accepted by any TM.

$w \in \{0,1\}^*$

TM M can be represented as finite length binary string.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	------
w_1	0	1	1	- - -
w_2	0	1	1	- - .
w_3	0	1	0	- -
w_4	!	!	!	
!	!	!	!	

If M_j accepts w_i put 1

otherwise 0

L_d consist of w_i such that entry $(i,i) = 0$

L_d will not accepted by any TM.

Assume that L_d is accepted by TM M_j

$L = \overline{\Sigma^*}$

$L_d = L(M_j)$

$w_j \in L_d$? $(\langle M_j \rangle, w_j) = 1$

$w_j \notin L_d$ $(\langle M_j \rangle, w_j) = 1$

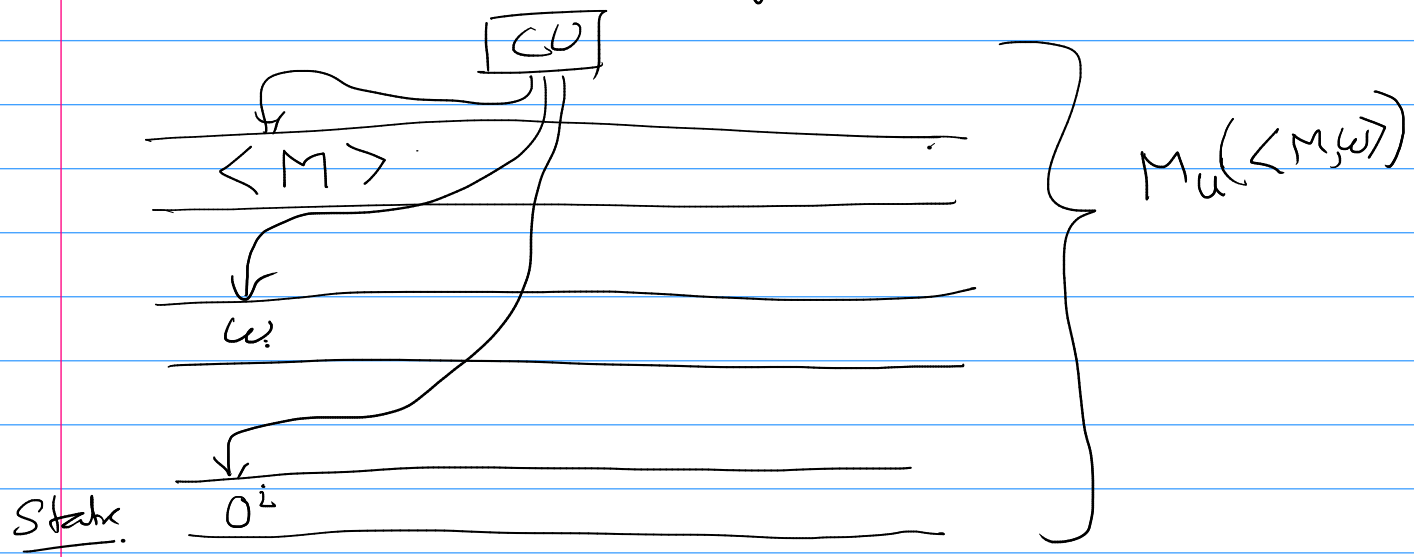
Universal Language

$$L_u = \{ \langle M, \omega \rangle \mid M \text{ accept } \omega \}$$

Universal T.M. M_u

$$L(M_u) = L_u$$

3 tape turing machine.



first check whether $\langle M, \omega \rangle$ is in correct format.

L_d, L_u → not r.e.

L_u → r.e.

$\overline{L_d}$ → L_d is not r.e. ⇒ L_d is not recursive ⇒ $\overline{L_d}$ is not recursive

$\overline{L_u}$ →

L_u is r.e.

Thm: L_u is not recursive.

$$L_u = \{ \langle M, \omega \rangle \mid \text{TM } M \text{ accepts } \omega \}$$

Reduction strategy to prove the above theorem.

$\overline{L_d}$ is not recursive.

Assume that L_u is recursive.

By using halting TM M such that $L(M) = L_u$ we can design a halting TM for $\overline{L_d}$.

We know that no such TM can exist for $\overline{L_d}$ hence our assumption is wrong.

L_u is not recursive.

TM M $L(M) = L_u$	TM M $L(M) = \overline{L_d}$
Yes	Yes
No	No

$$L_d = \{ \langle M_i, \omega_i \rangle \mid M_i \text{ does not accept } \omega_i \}$$

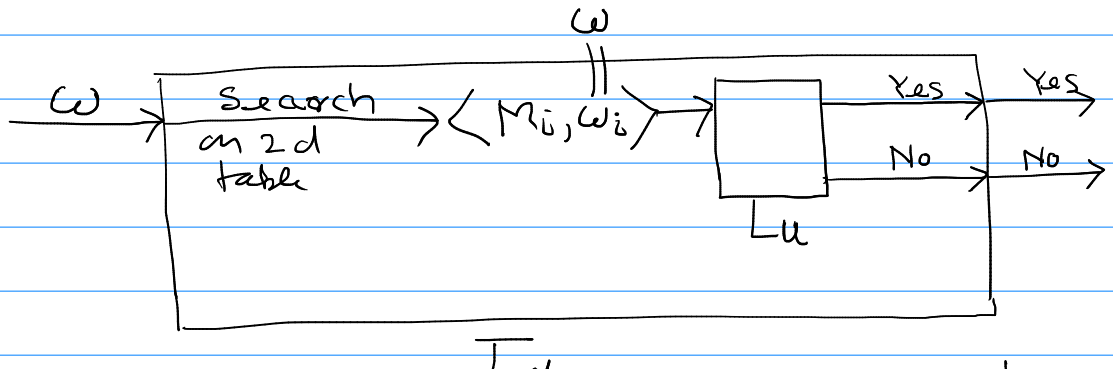
TM A for L_u

↓

TM M for $\overline{L_d}$

$$\overline{L_d} = \{ \omega_i \mid \text{TM } M_i \text{ accepts } \omega_i \}$$

$$L_u = \{ \langle M, \omega \rangle \mid \text{TM } M \text{ accepts } \omega \}$$



$$\overline{L_d} \leq_{TM} L_u$$

Turing Machine reduction

P – polynomial time

Thm $E_{TM} = \{ \langle M \rangle \mid L(M) = \phi \}$

$\exists$ no halting TM for E_{TM}.

$L_u = \{ \langle M, \omega \rangle \mid M \text{ accepts } \omega \}$ not recursive

Assume that E_{TM} is decidable

halting TM M $\quad L(M) = E_{TM}$

design a halting TM which accepts L_u.,

Hence L_u is recursive which is not so

our assumption is wrong, E_{TM} is undecidable.

$L_u \leq_{TM} E_{TM}$

TM R is a decider for E_{TM}

By using R we will make decider TM S for L_u

$R(\langle M \rangle)$ — Yes $\quad L(M) = \phi$

No definately M accept some ω

$S(\langle M, \omega \rangle)$:

1. get M_1

$M_1(x)$:

1. if $x \neq \omega$ reject
2. if $x = \omega$ run M on ω and accept if M does.

2. $R(\langle M_1 \rangle)$

accepts $\Downarrow$ $L(M_1) = \phi$ $\Downarrow$ Reject — M does not accept ω

Reject $\Downarrow$ $L(M_1) \neq \phi$ $\Downarrow$ Accept — M accept ω

$$L_r = \{ \langle M \rangle \mid L(M) \text{ is recursive} \}$$

Thm L_r is not r.e.

$\overline{L_u}$ is not r.e. — L_u is r.e but not recursive

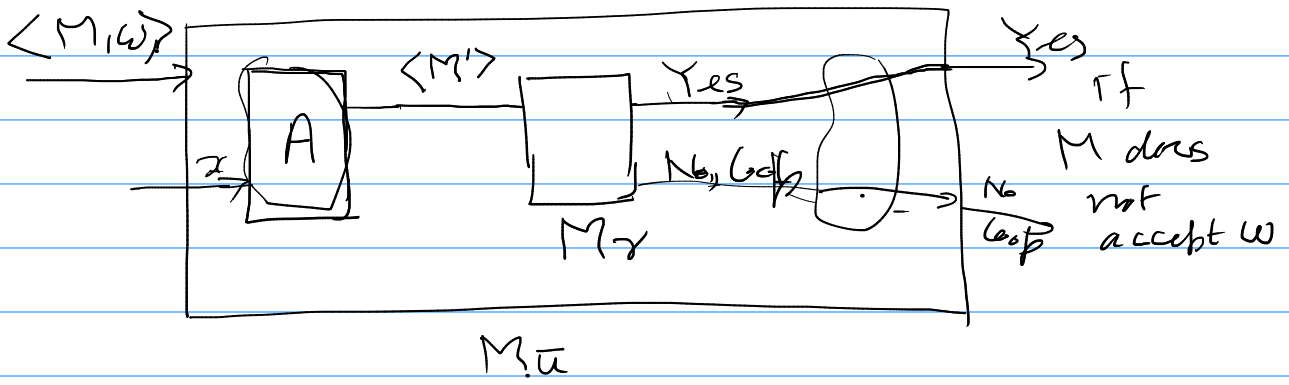
Assume that L_r is r.e.

$\exists$ a TM M_r which accepts L_r.

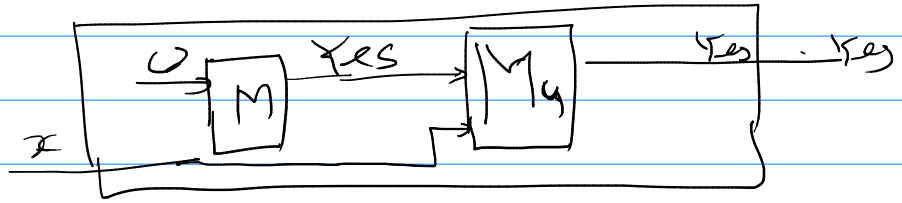
By using TM M_r construct a TM M such that $L(M) = L_{\bar{u}}$

$$L_{\bar{u}} = \{ \langle M, w \rangle \mid M \text{ does not accept } w \}$$

$L_{\bar{u}}$



$$L(M') = \begin{cases} \phi & M \text{ does not accept } w \quad \text{recursive} \\ L_u & M \text{ accepts } w \quad \text{non recursive} \end{cases}$$



A will write encoding of M'

Rice Theorem

RE: Set consist of all r.e languages of $\{0,1\}^*$

$$RE = \{ L \mid \exists \text{ TM } M \quad L(M) = L, \; L \subseteq \{0,1\}^* \}$$

Property

$$P \subseteq RE$$

Examples

$$P = \{ L \mid L \text{ is r.e} \quad L \text{ is infinite} \}$$

$$P' = \{ L \mid L \text{ is r.e.} \quad L \text{ is finite} \}$$

$$P'' = \{ L \mid L \text{ is r.e} \quad L \text{ is CFL} \}$$

$$\vdots \qquad (\text{r.e.})$$

A Language L has property P if $L \in P$.

$P = \{\phi\}$ or $P = RE$ trivial property.

otherwise non trivial property.

P : Property.

$$L_P = \{ \langle M \rangle \mid L(M) \in P \}$$

We will not learn any halting TM M such that $L(M) = L_P$

Rice Theorem

Any non trivial property P of r.e. Languages is undecidable.

Proof $\phi \notin P$ w.l.o.g.

P is non empty then

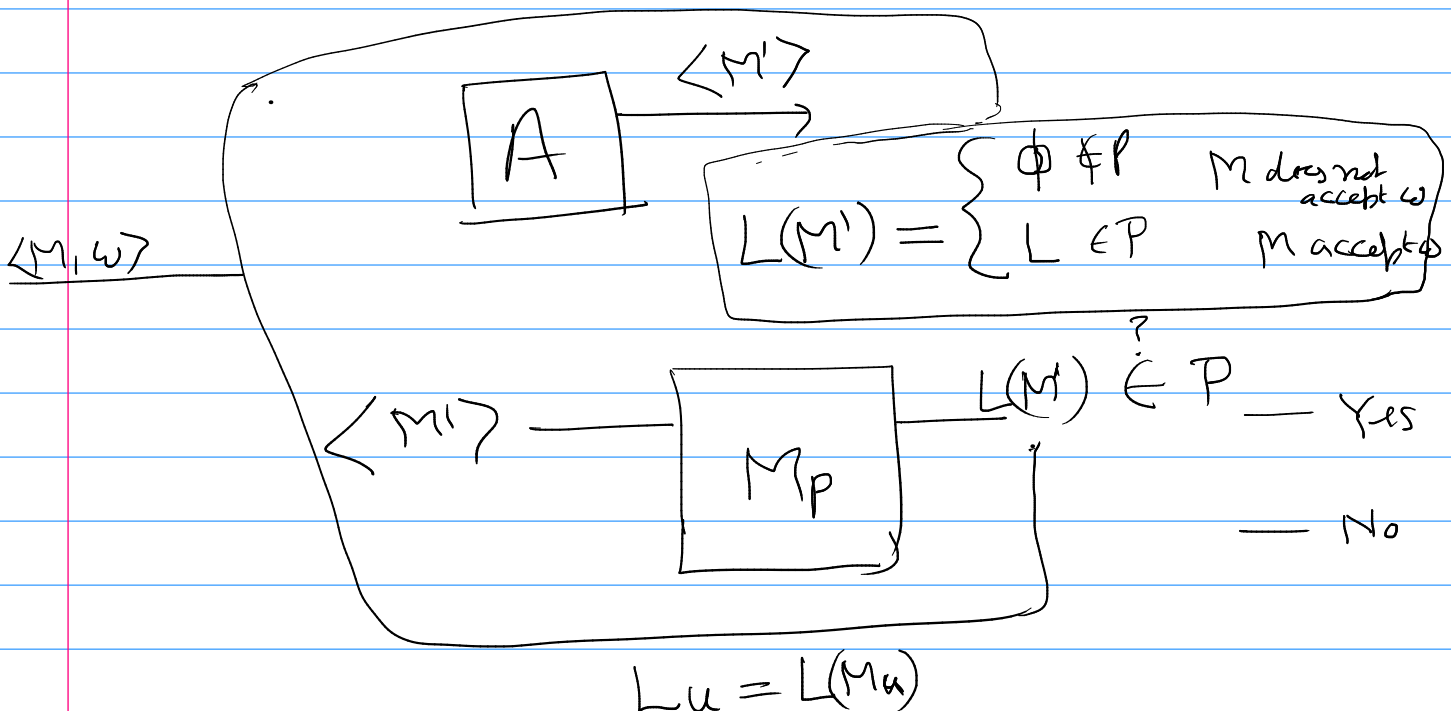
$\exists\ L \in P.$

$L(M_L) = L$

Assum that L_P is decidable.

By using TM M_P which accept L_P, we design a decider for L_U

We know that L_U is not recursive, hence our assumption is wrong.



Unrestricted grammar

$G = (V, T, P, S)$

$\alpha \longrightarrow \beta$

$\alpha, \beta \in (V \cup T)^*$

and α must contain a variable V

Thm Let L be r.e. language then $\exists$ an unrestricted grammar G such that

$S \xrightarrow{*} q_0(BB)(aa)(bb)(aa)(BB)$ $L(G) = L$.

P : $q_0(BB) \longrightarrow (BB)q_1$ $q_1(aa) \longrightarrow (aB)q_2$

String which we get from G

$q_0(BB)(aa)(bb)(aa)(BB)$

$(BB)q_1(aa)(bb)(aa)(BB)$

$\longrightarrow (BB)(aB)q_2(bb)(aa)(BB)$

Add $q_2(bb) \longrightarrow (bb)q_3$.

$(BB)(aB)(bb)q_3(aa)(BB)$

Add $(bb)q_3(aa) \longrightarrow q_4(bb)(aB)$

$(BB)(aB)q_4(bb)(aB)(BB)$

$\downarrow$ Add more production

aba

$\vdash q_0Baba$ $\delta(q_0, B) = (q_1, B, R)$

$\vdash Bq_1aba$

$\delta(q_1, a) = (q_2, B, R)$

$\vdash BBq_2ba$

$\delta(q_2, b) = (q_3, b, R)$

$\vdash BBbq_3a$

$\delta(q_3, a) = (q_4, B, L)$

$\vdash BBq_4bB$

when M accepts aba

Once we reach final state in TM.
we will erase all symbols
except the first copy of input
symbols.

$(BB)(aB)(bB)f(aB)(BB) \vdash (BBBfB)$ (B a b a)

$f(aB) \longrightarrow f(aB)f$

$(BB)^*(aB)^*(bB)f(aB)f(BB)$

P: $\begin{cases} (bB)f \longrightarrow f(bB)f & (BB)f \to f(BB)f \\ (aB)f \longrightarrow f(aB)f \end{cases}$

$f(BB)f(aB)f(bB)f(aB)f(BB)$

Add
$f(BB) \longrightarrow B$
$f(aB) \longrightarrow a$
$f(bB) \longrightarrow b$

$BabaB$ G has generated aba

$q_0(BB)(a_1a_1)(a_2a_2)\cdots(a_na_n)(BB)(BB)(BB)(BB)$

$S \longrightarrow S(BB) \mid T$

$T \longrightarrow T(\sigma\sigma) \mid q_0(BB)$

$\forall\ \sigma \in \Sigma$

$$\delta(p,a) = (q,b,R)$$

$$p(\sigma a) \to (\sigma b)\, q$$
$$\forall\, \sigma \in \Sigma \cup \{B\}$$

$$\delta(p,a) = (q,b,L)$$

$$(\sigma_1 \sigma_2)\, p\, (\sigma_3 a) \to q\,(\sigma_1 \sigma_2)(\sigma_3 b)$$
$$\forall\, \sigma_1, \sigma_3 \in \Sigma \cup \{B\}$$
$$\forall\, \sigma_2 \in \Gamma \cup \{B\}$$

$$f(\sigma_1 \sigma_2) \to f(\sigma_1, \sigma_2)\, f$$
$$\forall\, \sigma_1 \in \Sigma \cup \{B\}$$
$$\forall\, \sigma_2 \in \Gamma \cup \{B\}$$

$$f(\sigma_1, \sigma_2) \to \sigma_1 \quad \text{Erase}$$

CSG Context Sensitive Grammar.

$$G = (V, T, P, S)$$

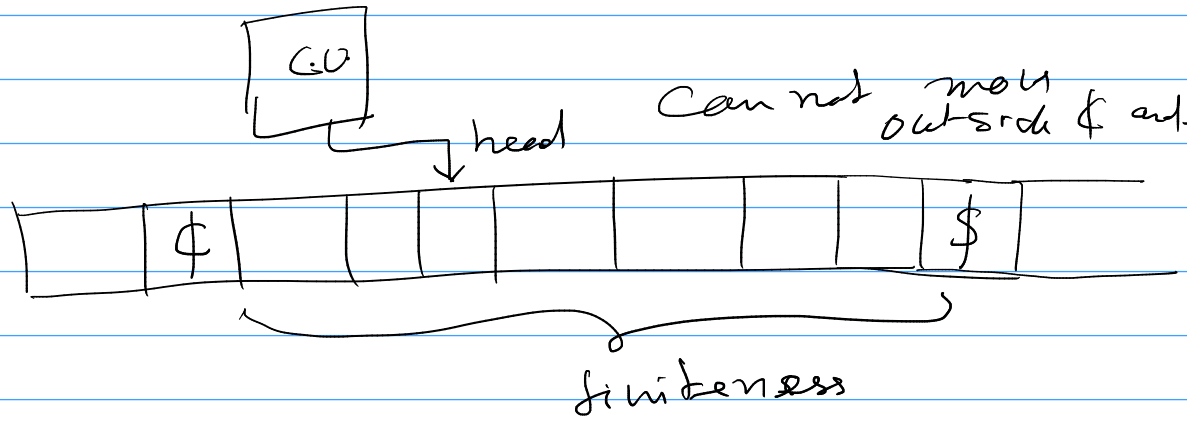
$$\alpha \to \beta$$
$$\alpha, \beta \in (V \cup T)^*$$
α must contain a variable

$$|\alpha| \leq |\beta|$$

CSG G

$L = L(G)$ Context Sensitive Language

Linear Bounded Automata.



CSL ⇒ recursive.