NORMAL FORMS FOR CONTEXT FREE GRAMMAR

Chomsky Normal Form

↳ Terminal $A \to a$

↳ Variable Variable $A \to BC$

1. $S \to AS \mid a$, $A \to SA \mid b$ ✓
2. $S \to AS \mid AAS$, $A \to SA \mid aa$ ✗

Greinbach Normal Form

↳ Terminal 1 or 0 $S \to a$

↳ Terminal, infinite variables $S \to aABCD$

↳ infinite variables $S \to ABC$

1. $S \to cAB$, $A \to aA \mid bB \mid b$, $B \to b$ ✓
2. $S \to abSb \mid aa$ ✗

Convert to Chomsky Normal Form

↳ shouldn't contain λ → else do substitution

↳ add new variables for terminals

0) $S \to ABa$
$A \to aab$
$B \to Ac$

Introducing I, J, K

$S \to ABI$
$A \to IIJ$
$B \to AK$
$I \to a$
$J \to b$
$K \to c$

Introducing X, Y

$S \to AX$
$X \to BI$
$A \to IY$
$Y \to IJ$
$B \to AK$
$I \to a$
$J \to b$
$K \to c$

$S \to AX$
$X \to BI$
$A \to IY$
$Y \to IJ$
$B \to AK$
$I \to a$
$J \to b$
$K \to c$

→ Chomsky

Convert to Greinbach Normal Form

↳ Change Terminal symbols to variables

0) $S \to abSb$
$S \to aa$

Introducing I, J

$S \to IbSJ$
$S \to aI$
$I \to a$
$J \to b$

$S \to IbSJ$
$S \to aI$
$I \to a$
$J \to b$

→ Greinbach

CYK Algorithm → only applicable for chomsky normal form

↳ checks wether string belongs to a CFG or not

1. convert to CNF if not
2. fill 1st diagonals with ther derivatives
3. fill 2nd diagonals with concatenated derivatives

⋮

CHOMSKY NORMAL FORM

$A \to \overset{V}{B}\overset{V}{C}$ $A \to \overset{T}{a}$

Q) $S \to AB$
$A \to BB|a$
$B \to AB|b$

abbb valid member or not

1	2	3	4
a	b	b	b
A	B	B	B
AB	BB	BB	
SB	A	A	

→ diagonal 1

→ diagonal 2

→ if S in (1,4) then valid
if no S in (1,4) then invalid

	4	3	2	1
1	SB	A	S,B	A
2	S,B	A	B	
3	A	B		
4	B			

valid

1,3 → diagonal 3 ← 2,4

1|2|3 — 11,23: AA ✗ — 12,33: SB B / SB BB ✗ A

2|3|4 — 22,34: BA ✗ — 23,44: AB / SB

1,4 → diagonal 4

1|2|3|4 — 11,24: A SB / As ✗ — 12,34: SB A / AB SB ✗ — 13,44: A B / SB / SA BA ✗ ✗

Pumping Lemma for CFL

1. Assume A is a regular language → for contradiction

   $|s| \geq P$ → assume
   Divide S into 5 pieces

   $S = uvxyz$

2. Show 3 pumping conditions unsatisfied

   1. $uv^i x y^i z$, $i \geq 0$
   2. $|vy| > 0$
   3. $|vxy| \leq P$

3. S cannot be pumped = CONTRADICTION

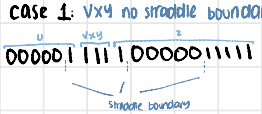
8) $L = \{ww \mid w \in \{0,1\}^*\}$ is NOT context free · SHOW

assume L is context free

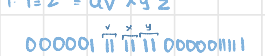
$S \geq 5$ assume

$S = \underbrace{01}_{w}\underbrace{01}_{w}$

case 1: vxy no straddle boundary

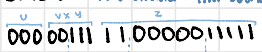


1. $i=2$ (assume) $= uv^2xy^2z$ ✗

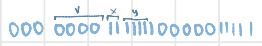


$0^5 0 1^5$ first half ≠ second half as wrong pattern

case 2a: vxy straddle first boundary

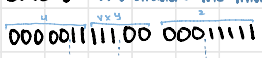


1. $i=2 = uv^2xy^2z$ ✗

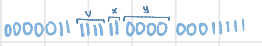


$0^3 1^3 0^5 1^5$ first half ≠ second half as wrong pattern

case 3 : vxy straddle the midpoint

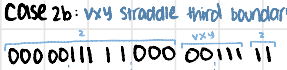


1. $i=2 = uv^2xy^2z$ ✗

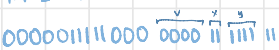


$0^5 1^3 0^3 1^5$ first half ≠ second half as wrong pattern

case 2b: vxy straddle third boundary



1. $i=2 = uv^2xy^2z$ ✗



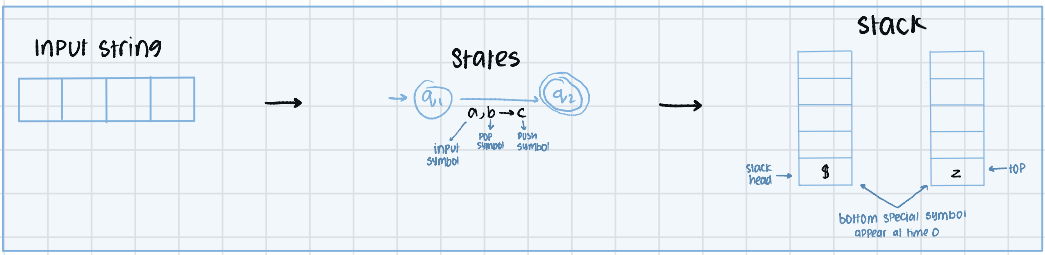
$0^5 1^5 0^3 1^3$ first half ≠ second half as wrong pattern

HENCE NOT CONTEXT FREE

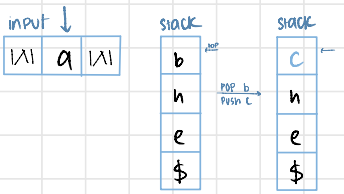
Pushdown Automata (PDA)

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

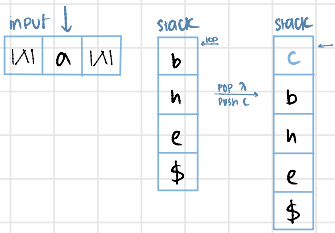
- Q: states
- Σ: input alphabets
- Γ: stack
- δ: transition function
- q_0: initial state
- z: stack start symbol
- F: accepted states



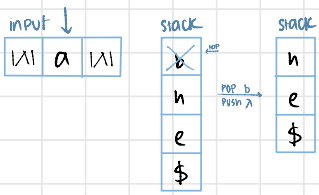
1. $(q_1) \xrightarrow{a,b \to c} (q_2)$



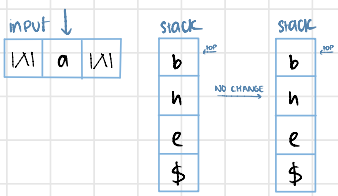
2. $(q_1) \xrightarrow{a,\lambda \to c} (q_2)$



3. $(q_1) \xrightarrow{a,b \to \lambda} (q_2)$



4. $(q_1) \xrightarrow{a,\lambda \to \lambda} (q_2)$



Instantaneous Description

(q, u, s)

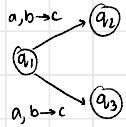
- q: current state
- u: remaining input
- s: current stack contents

* stack contents dont matter

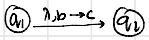
Non Deterministic PDAs

↳ PDAs are non deterministic

↳ allowed non deterministic transitions



λ - transition



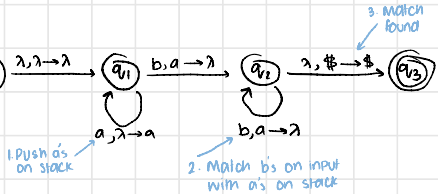
↳ A string is accepted if

↳ all the input is consumed

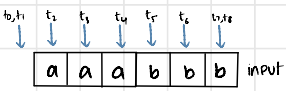
↳ the last state is an accepting state

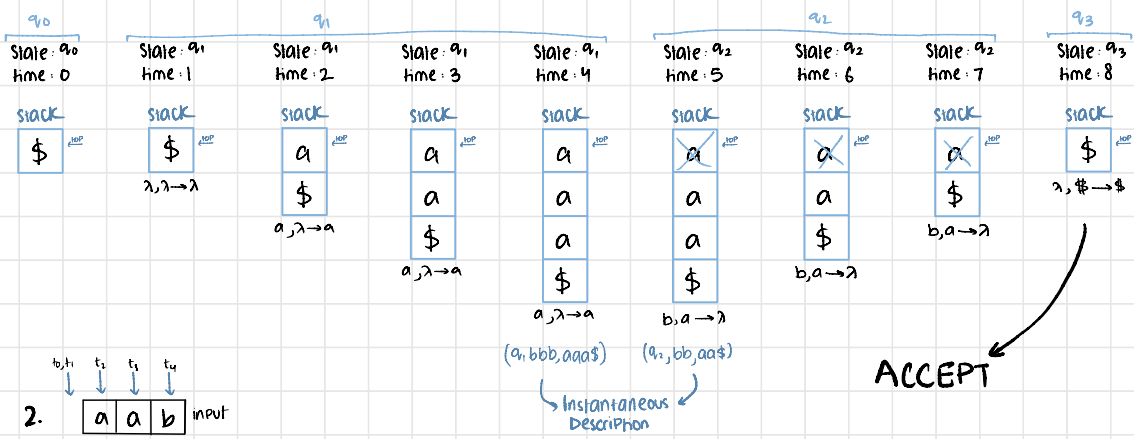
} both conditions to be fulfilled

Q) PDA M : $L(M) = \{a^n b^n : n \geq 0\}$



1.



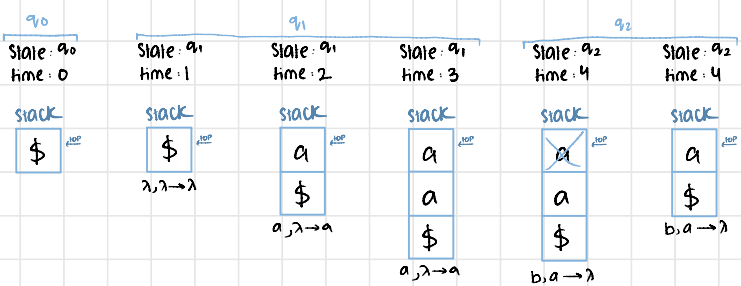


(q1, bbb, aaa$) (q2, bb, aa$)

↳ Instantaneous Description

ACCEPT

2.



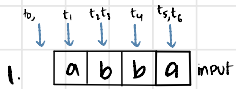
↳ REJECT

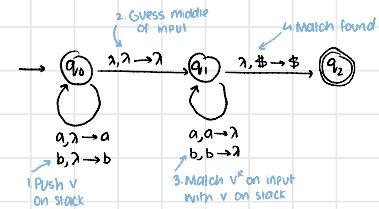
as no accepted state reached and inputs all consumed

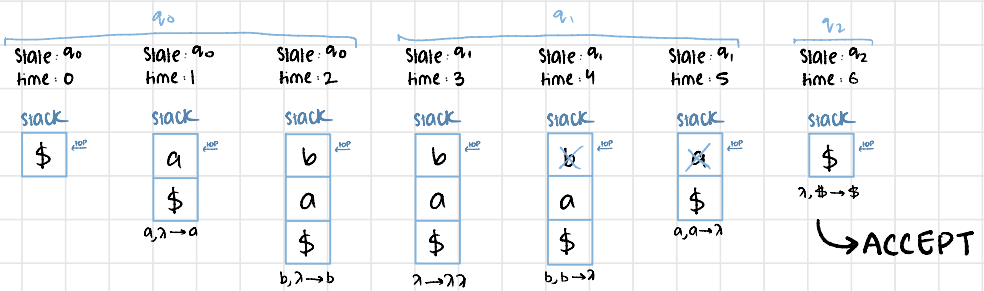
hence no accepting computation for aab

Q) PDA M: $L(M) = \{ vv^R : v \in \{a,b\}^* \}$ → Palindrome

1.

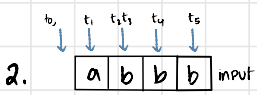


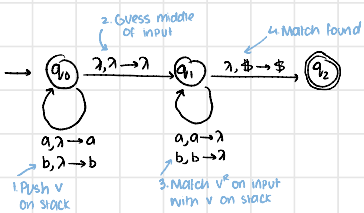


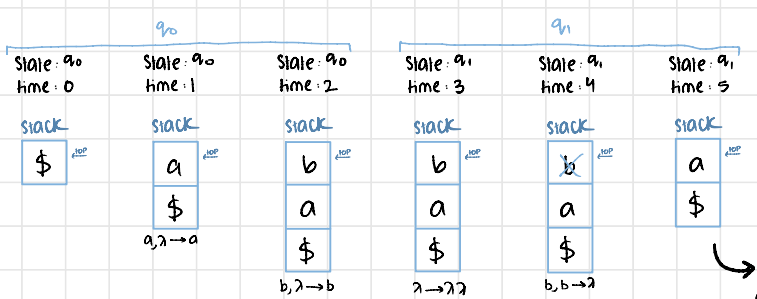


↳ ACCEPT

2.



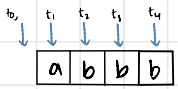


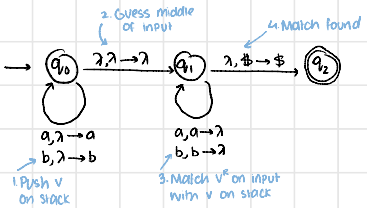


↳ REJECT

as can't consume input

3.





State: q_0 Time: 0 — stack: \$ (top)

State: q_0 Time: 1 — stack: a, \$ (top); $a,\lambda \to a$

State: q_0 Time: 2 — stack: b, a, \$ (top); $b,\lambda \to b$

State: q_0 Time: 3 — stack: b, b, a, \$ (top); $b,\lambda \to b$

State: q_0 Time: 4 — stack: b, b, b, a, \$ (top); $b,\lambda \to b$

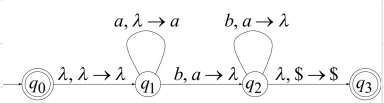
State: q_1 Time: 5 — stack: b, b, b, a, \$ (top); $\lambda \to \lambda\lambda$

↳ REJECT

as no accepted state reached and inputs all consumed

A computation:

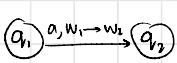
$(q_0, aaabbb, \$) \vdash (q_1, aaabbb, \$) \vdash$
$(q_1, aabbb, a\$) \vdash (q_1, abbb, aa\$) \vdash (q_1, bbb, aaa\$) \vdash$
$(q_2, bb, aa\$) \vdash (q_2, b, a\$) \vdash (q_2, \lambda, \$) \vdash (q_3, \lambda, \$)$



For convenience we write:

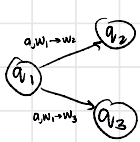
$(q_0, aaabbb, \$) \overset{*}{\vdash} (q_3, \lambda, \$)$

Formalities for PDA's



Transition function

$\delta(q_1, a, w_1) = \{(q_2, w_2)\}$

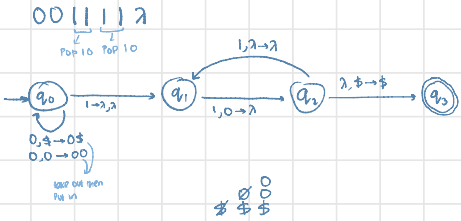


Transition function

$\delta(q_1, a, w_1) = \{(q_2, w_2), (q_3, w_3)\}$

Construct a PDA

Q) $L_1 = \{0^n 1^{2n}, n>0\}$



Transition states

$q_0, 0, \$ \rightarrow q_0, 0\$$

$q_0, 0, 0 \rightarrow q_0, 00$

$q_0, 1, \lambda \rightarrow q_1, \lambda$

$q_1, 1, 0 \rightarrow q_2, \lambda$

$q_2, 1, \lambda \rightarrow q_1, \lambda$

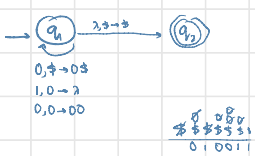
$q_2, \lambda, \$ \rightarrow q_3, \$$

Q) $L_1 \{ w \in (0,1)^* \mid n_0(w) = n_1(w)$

↳ no of 0,1s same

01, 10, 0011, 010011

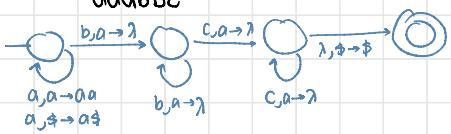
010011



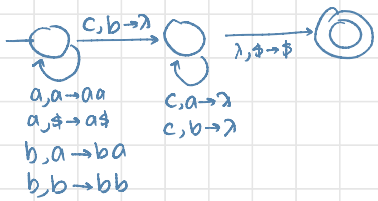
what about 1, $ → 1$

Q) $L_1 = \{a^{m+n} b^m c^n\}$

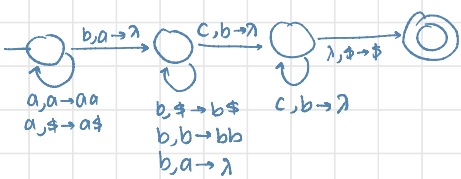
aaabbc



Q) $L_1 = \{a^n b^m c^{n+m}\}$

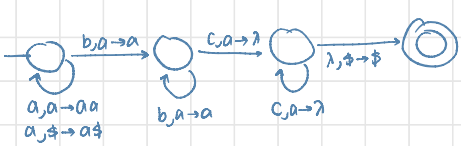


Q) $L_1 = \{a^n b^{m+m} c^m\}$

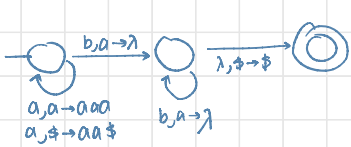


Q) $L_1 = \{a^n b^m c^n\}$ $n, m \geq 1$

aba abba aabaa ✓



Q) $L_1 = \{a^n b^{2n}\}$



Language of PDA

$L(M) = \{w : (q_0, w, z) \overset{*}{} (q_f, \lambda, s)\}$

q_0: initial state; q_f: accepted state

Convert CFG to PDAs

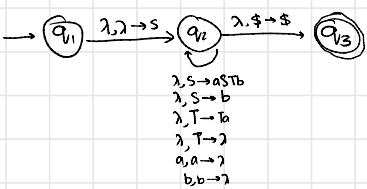
↳ $L(G) = L(M)$

↳ for each production, add transitions $A \to w \Rightarrow \lambda, A \to w$

↳ for each terminal, add transitions $a \Rightarrow a, a \to \lambda$

1.

Grammar	PDA	
$S \to aSTb$	$\lambda, S \to aSTb$	
$S \to b$	$\lambda, S \to b$	
$T \to Ta$	$\lambda, T \to Ta$	$a, a \to \lambda$
$T \to \lambda$	$\lambda, T \to \lambda$	$b, b \to \lambda$



In general, it can be shown that:

Grammar G generates string w $\overset{*}{S \Rightarrow w}$

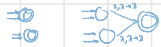
If and Only if

PDA M accepts w $(q_0, w, \$) \overset{*}{\vdash} (q_2, \lambda, \$)$

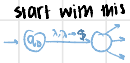
Therefore $L(G) = L(M)$

convert PDAs to CFG

↳ L(M) = L(G)

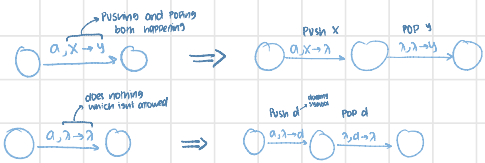
1. 1 final state 

2. start and end with empty stack

start with this 

end with this 

3. assure each transition Pushes/Pops, but not both



4. NO underflow or overflow stack

↙ Popping on empty stack

↘ Pushing when stack is full

So far we have shown:

$L(G) \subseteq L(M)$

With a similar proof we can show

$L(G) \supseteq L(M)$

Therefore: $L(G) = L(M)$

Positive closure for CFL

↳ context free

↳ has new start variable S

↳ has new additional production $S \to S_1 | S_2$ / $S \to S_1 S_2$ / ...

↳ UNION → CONTEXT FREE

$L_1 = \{a^n b^n\}$ $\quad S_1 \to aS_1 b | \lambda$

$L_2 = \{ww^R\}$ $\quad S_2 \to aS_2 | bS_2 b | \lambda$

$L_1 \cup L_2 = \{a^n b^n\} \cup \{ww^R\}$ $\quad S \to S_1 | S_2$

↳ CONCATENATION → CONTEXT FREE

$L_1 = \{a^n b^n\}$ $\quad S_1 \to aS_1 b | \lambda$

$L_2 = \{ww^R\}$ $\quad S_2 \to aS_2 | bS_2 b | \lambda$

$L_1 L_2 = \{a^n b^n\}\{ww^R\}$ $\quad S \to S_1 S_2$

↳ STAR OPERATION → CONTEXT FREE

$L = \{a^n b^n\}$ $\quad S \to aSb | \lambda$

$L = \{a^n b^n\}^*$ $\quad S_1 \to SS_1 | \lambda$

Negative closure for CFL

↳ not necessarily context free

↳ INTERSECTION → NOT NECESSARILY CONTEXT FREE

$L_1 = \{a^n b^n c^m\}$ $\quad S \to AC$, $A \to aAb | \lambda$, $C \to cC | \lambda$

$L_2 = \{a^n b^m c^m\}$ $\quad S \to AB$, $A \to aA | \lambda$, $B \to bBc | \lambda$

$L_1 \cap L_2 = \{a^n b^n c^n\}$ NOT context free

↳ COMPLEMENT → NOT NECESSARILY CONTEXT FREE

$L_1 = \{a^n b^n c^m\}$ $\quad S \to AC$, $A \to aAb | \lambda$, $C \to cC | \lambda$

$L_2 = \{a^n b^m c^m\}$ $\quad S \to AB$, $A \to aA | \lambda$, $B \to bBc | \lambda$

$\overline{L_1 \cup L_2} = L_1 \cap L_2 = \{a^n b^n c^n\}$ NOT context free

Intersection of Context Free Language and Regular Language

L_1 context free
L_2 regular
} $L_1 \cap L_2$ → context free

Machine M_1
PDA for L_1
context free

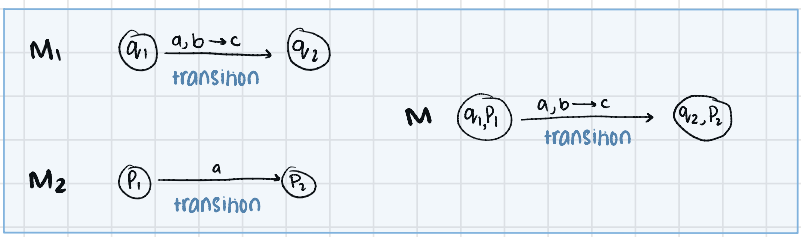
Machine M_2
DFA for L_2
regular

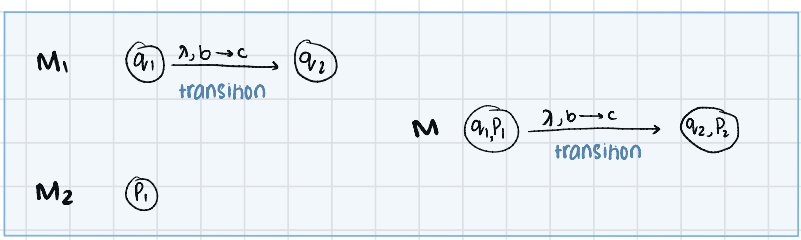
M simulates in parallel M_1 and M_2

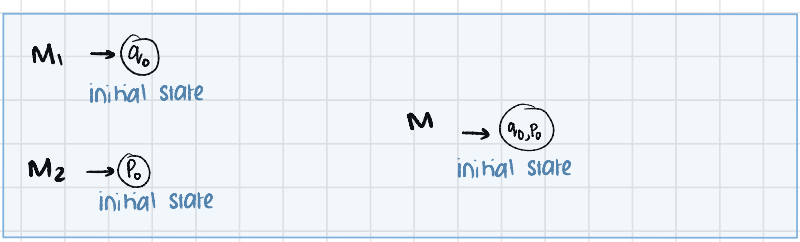
M accepts string w if and only if

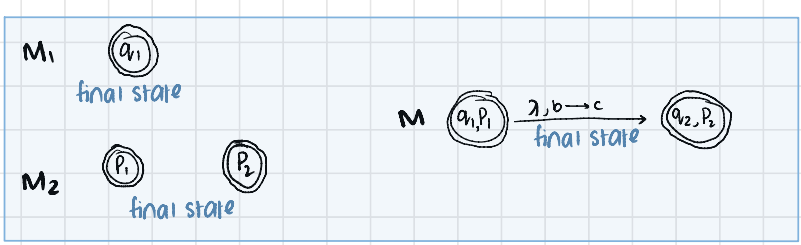
M_1 accepts string w and
M_2 accepts string w

$$L(M) = L(M_1) \cap L(M_2)$$









Therefore:

M is PDA

↓

$L(M_1) \cap L(M_2)$ is context-free

↓

$L_1 \cap L_2$ is context-free

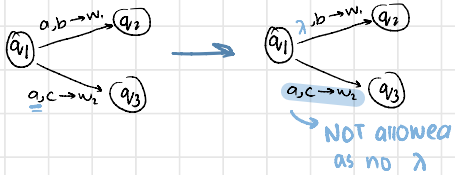
Deterministic PDA (DPDA)

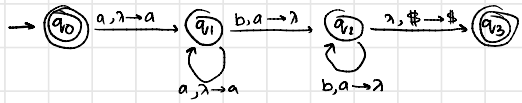
↳ λ must exist

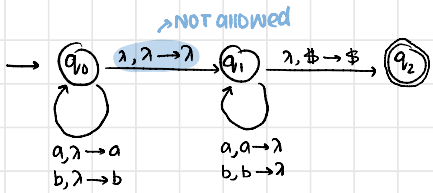
↳ λ, λ → λ ✗

1. 

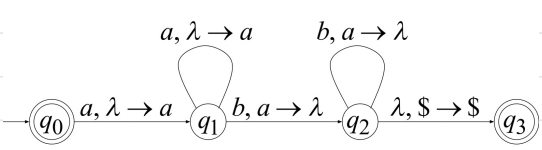
2. 

3. 

4. 

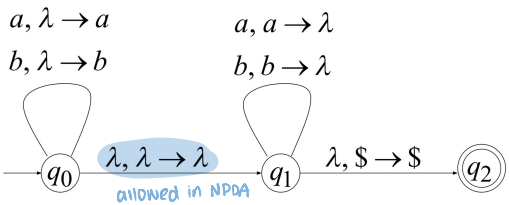
5. 

$$L(M) = \{a^n b^n : n \geq 0\}$$



NON Deterministic PDA (NPDA)

$$L(M) = \{ww^R\}$$



NPDAs are more powerfull than DPDAs

CHECK SLIDES

Proof L is non deterministic CFL
there is no DPDA that accepts L

↳ Proof by contradiction

$a^n b^n c^n$ → PDA fails as more than 2 comparisons

Turing Machines (TM)

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

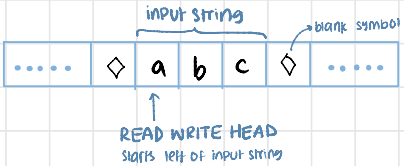
- Q — states
- Σ — input alphabets
- Γ — TAPE alphabet
- δ — transition function
- q_0 — initial state
- $\Diamond$ — blank
- F — final states

Standard Turing Machine

- Deterministic
- Infinite Tape in both directions
- Tape is the in the input/output file

TAPE

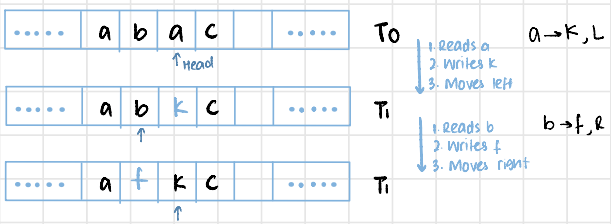
No boundaries -- infinite length



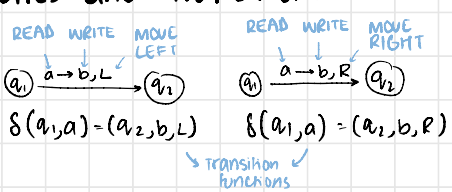
head at each transition
1. Reads a symbol
2. Writes a symbol
3. Moves left/right

CONTROL UNIT

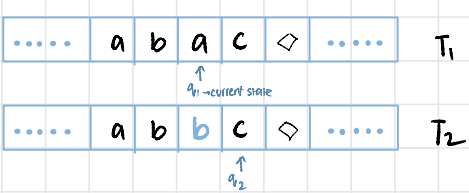
1.



States and transitions



1. q_1 a → b, L q_2

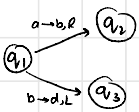


2. q_1 ◇ → g, R q_2

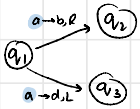
Turing Machines are deterministic

↳ NO same read symbols

ALLOWED



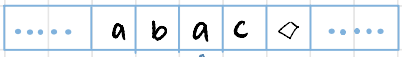
NOT ALLOWED

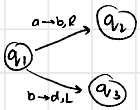


Halting

↳ when no possible transitions

1.





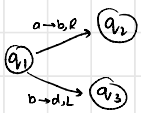
no possible transition

HALT

Partial Transition function

↳ NO transition for an input symbol

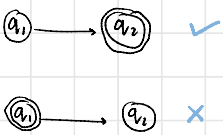
1.



No transition for input symbol c

Final states

↳ NO transitions from final state



Accept input

↳ if machine halts in final state

we call it

↳ Turing Recognizable

↳ Turing Acceptable

↳ Recursively Enumerable

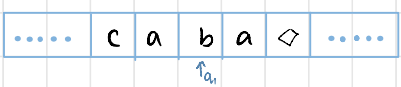
Reject input

↳ if machine halts in nonfinal state

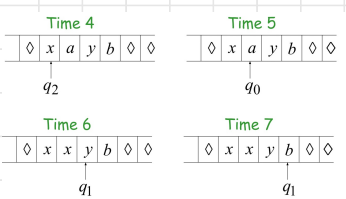
OR

↳ if machine enters infinite loop

Instantaneous Description



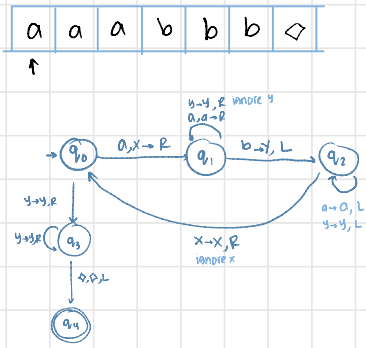
ID = Ca q_1 ba



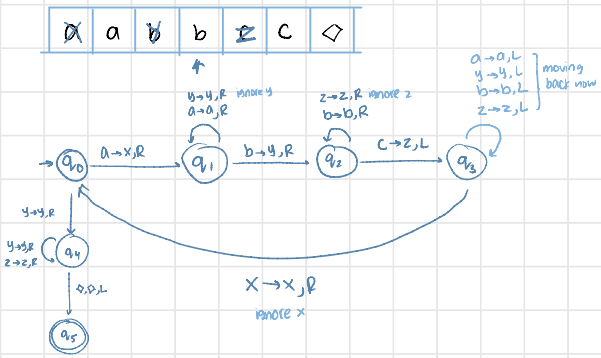
Equivalent notation: q_2 xayb $\overset{*}{\vdash}$ xxy q_1 b

Design Turing Machine

Q) $\{a^n b^n \mid n \geq 1\}$

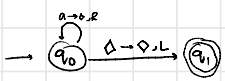


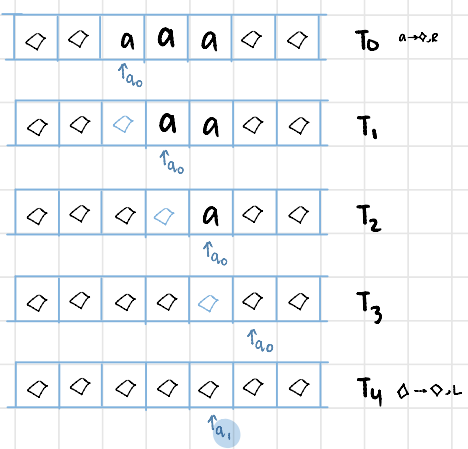
Q) $\{a^n b^n c^n \mid n \geq 1\}$



Turing Machine Examples

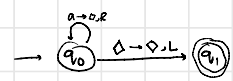
1. A turing machine that accepts the language a^*





Halt and Accept

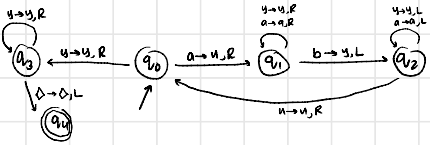
2. A turing machine that accepts the language a^*



NO possible transition

Halt and Reject

3. A turing machine for language $\{a^n b^n\}$



Basic Idea:

Match a's with b's:
Repeat:
  replace leftmost a with x
  find leftmost b and replace it with y
Until there are no more a's or b's

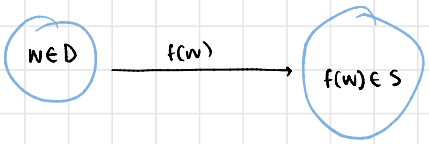
If there is a remaining a or b reject

◇ a a b b ◇ ◇ $\uparrow a_0$	T_0	$a \to u, R$
◇ u a b b ◇ ◇ $\uparrow a_1$	T_1	$a \to a, R$
◇ u a b b ◇ ◇ $\uparrow a_1$	T_2	$b \to y, L$
◇ u a y b ◇ ◇ $\uparrow a_2$	T_3	$a \to a, L$
◇ u a y b ◇ ◇ $\uparrow a_2$	T_4	$u \to u, R$
◇ u a y b ◇ ◇ $\uparrow a_0$	T_5	$a \to u, R$
◇ u u y b ◇ ◇ $\uparrow a_1$	T_6	$y \to y, R$
◇ u u y b ◇ ◇ $\uparrow a_1$	T_7	$b \to y, L$
◇ u u y y ◇ ◇ $\uparrow a_2$	T_8	$y \to y, L$
◇ u u y y ◇ ◇ $\uparrow a_2$	T_9	$u \to u, R$
◇ u u y y ◇ ◇ $\uparrow a_0$	T_{10}	$y \to y, R$
◇ u u y y ◇ ◇ $\uparrow a_3$	T_{11}	$y \to y, R$
◇ u u y y ◇ ◇ $\uparrow a_3$	T_{12}	$\diamond \to \diamond, L$
◇ u u y y ◇ ◇ $\uparrow a_4$	T_{13}	Halt and Accept

Computing Functions with Turing Machines

A function $f(w)$ has

Domain D — Result Region S



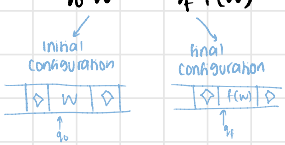
Decimal: 5

Binary: 101

Unary: 11111 → preferd as easier to manipulate with Turing machine

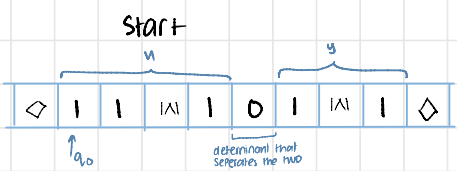
↳ A function f is computable if there is a Turing Machine M such that

$q_0\, w \;*\; q_f\, f(w)$

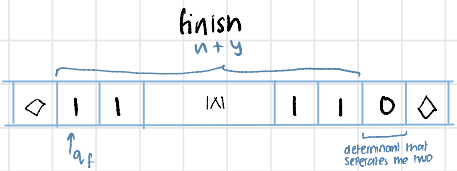


1. $f(n,y) = n + y$

input string: n 0 y unary

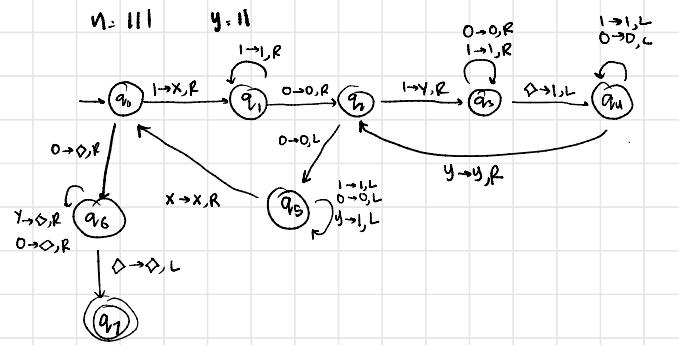


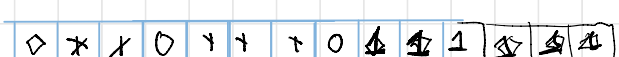
output string: n+y 0 unary



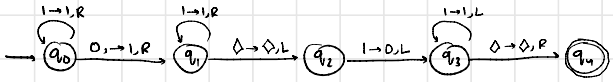
0) $f(n,y) = ny$

n: 111 y: 11

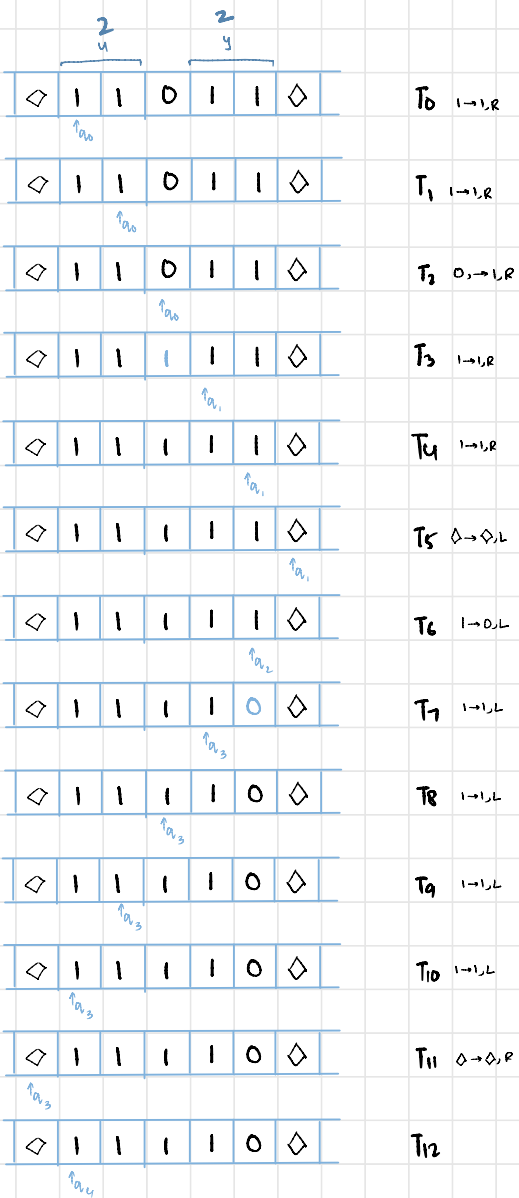




2. $f(x,y) = x+y$



$x = 11$, $y = 11$



Halt and Accept

combining TUring machine

↓ see SLIDES

Expected Final Exam Questions

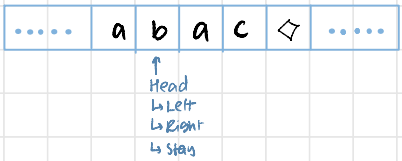
Design TM and explain the working of each state or combination of related states

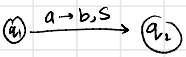
1. Binary Addition
2. Binary Subtraction $q_0\ waw \boxtimes^{*} q_f\ w$
3. Binary Multiplication

Give basic idea and Design TM over unary Alphabet and explain the working of each state or combination of related states

4. Primality Checking $q_0\ w \boxtimes^{*} q_f\ a$
5. Factorial Calculation $q_0\ w \boxtimes^{*} q_f\ w!$

Turing Machine with Stay Option

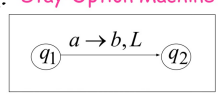




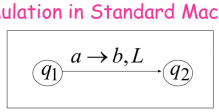
↳ Stay Option Machines are atleast as powerful as Standard Machines

1. a Stay option machine that never uses S move = Standard Machine
2. a Standard machine can simulate a Stay option machine

1. Stay-Option Machine

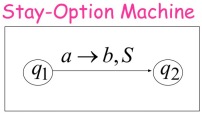


Simulation in Standard Machine

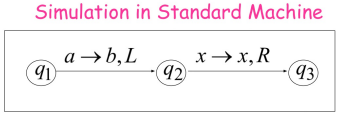


Similar for Right moves

2. Stay-Option Machine



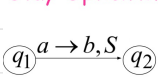
Simulation in Standard Machine

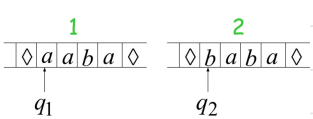


For every tape symbol x

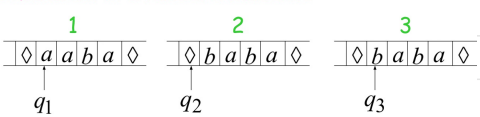
Example

Stay-Option Machine:

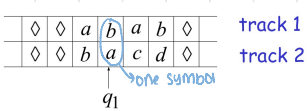


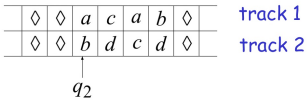


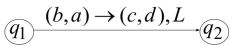
Simulation in Standard Machine:



Turing Machine with Multiple Track Tape



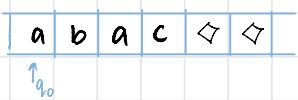




Courtesy Costas Busch - RPI 21

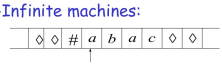
Turing Machine with Semi-Infinite Tape

- head extend infinitely only to the right
- Initial position is left most cell
- when head moves left from border, it returns to same position



- Semi Infinite Machies have the same power as standard Machines
  1. a standard machine can stimulate a Semi Infinite machine
  2. a Semi Infinite machine can stimulate a standard machine

1. Standard Turing machines simulate Semi-Infinite machines:



Standard Turing Machine

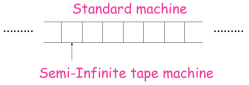
a. insert special symbol # at left of input string

b. Add a self-loop to every state (except states with no outgoing transitions)

Courtesy Costas Busch - RPI 3

2. Semi-Infinite tape machines simulate Standard Turing machines:

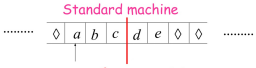
Standard machine



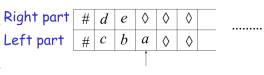
Semi-Infinite tape machine

Squeeze infinity of both directions in one direction

Standard machine

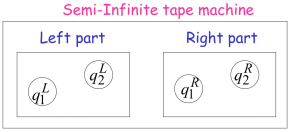


Semi-Infinite tape machine with two track

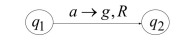


Standard machine

Semi-Infinite tape machine

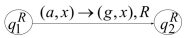


Standard machine

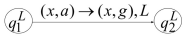


Semi-Infinite tape machine

Right part



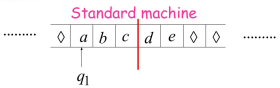
Left part



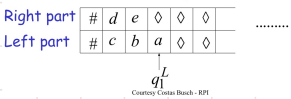
For all tape symbols X

Time 1

Standard machine

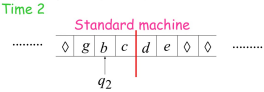


Semi-Infinite tape machine

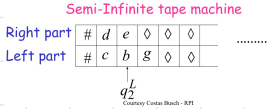


Time 2

Standard machine



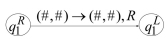
Semi-Infinite tape machine



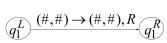
At the border:

Semi-Infinite tape machine

Right part

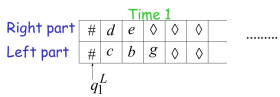


Left part

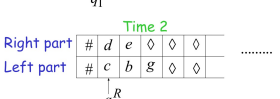


Semi-Infinite tape machine

Time 1



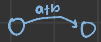
Time 2



END OF PROOF

GENRALIZED TRANSITION GRAPH (GTG)

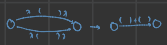
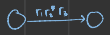
- ↳ initial state >0
- ↳ final state ≥0
- ↳ finite states, input letters
- ↳ Directed edges connecting regular expression states

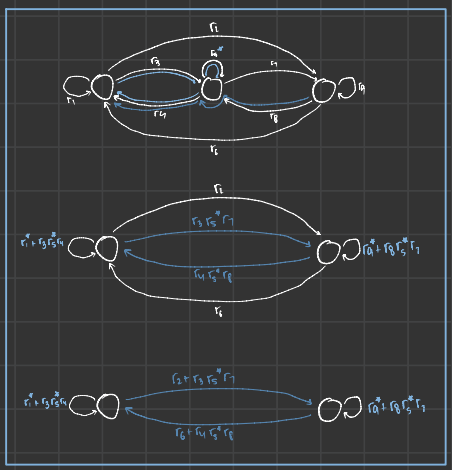


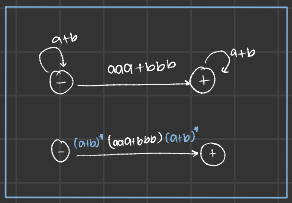
NON DETERMINISIM

- ↳ TG, GTG
- ↳ There may exist none or more than one path for a certain string

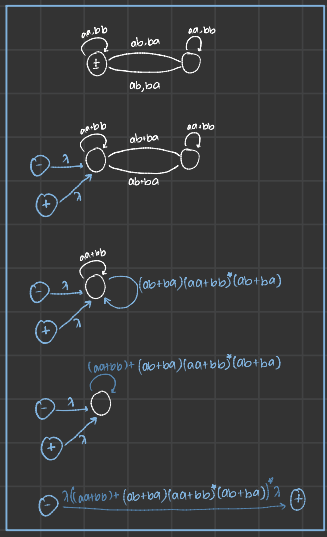
TG TO GTG CONVERSION

1. more than 1 start states = add new start state and connect old states with λ 
2. more than 1 final states = add new final state and connect old states with λ 
3. if a state has two or more edges = convert to 1 transition 
   - ↳ loops = * 
   - ↳ 2 possibilities = + 
4. Bypass and state elimination 
   - ↳ if 3 states connected in sequence : eliminate mid state and join the other two 





Since start, final state same
no other final state
add seperate start, final state and connect old state

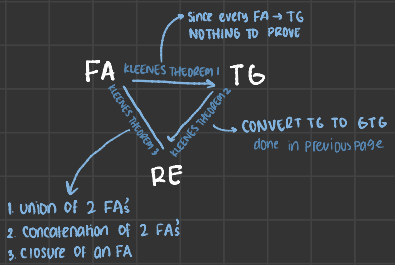


KLEENES THEOREMS

Regular Expressions

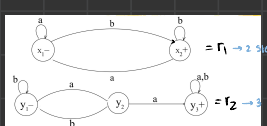
↳ FA
↳ TG
↳ RE

if can be expressed by one then the other and can be expressed aswell



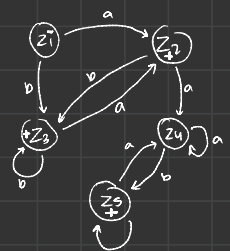
1. union of 2 FA's

↳ sum of 2 FA's = $r_1 + r_2$
↳ make transition table
↳ r_3 initial state = r_1, r_2 initial state
↳ r_3 final state = r_1, r_2 final state
↳ take $z_1 \equiv \{r_1 \text{ initial}^-, r_2 \text{ initial}^-\}$
↳ r_3 final = r_1 and r_2 finals
↳ total no of possible states = states of r_1 x states of r_2



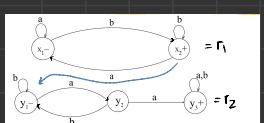
$r_3 = r_1 + r_2 \rightarrow$ 6 possible states

Old states	a	b
$z_1^- \equiv \{x_1^-, y_1^-\}$	$z_2 \equiv \{x_1^-, y_2\}$	$z_3 \equiv \{x_2^+, y_1^-\}$
$z_2 \equiv \{x_1^-, y_3^+\}$	$z_4 \equiv \{x_1^-, y_3^+\}$	$z_3 \equiv \{x_2^+, y_1^-\}$
$z_3^+ \equiv \{x_1^+, y_1^-\}$	$z_2 \equiv \{x_1^-, y_2\}$	$z_3 \equiv \{x_2^+, y_1^-\}$
$z_4^+ \equiv \{x_1^-, y_3^+\}$	$z_4 \equiv \{x_1^-, y_3^+\}$	$z_5 \equiv \{x_2^+, y_3^+\}$
$z_5^+ \equiv \{x_2^+, y_3^+\}$	$z_4 \equiv \{x_1^-, y_3^+\}$	$z_5 \equiv \{x_2^+, y_3^+\}$



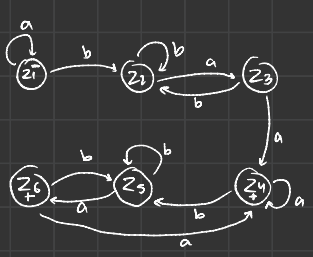
2 concatenation of 2 FA's

↳ $r_1 r_2$
↳ make transition table
↳ r_3 initial state = r_1 initial state
↳ r_3 final state = r_2 final state
↳ r_1 final state = r_2 initial state
↳ take $z_1 \equiv \{r_1 \text{ initial}^-\}$
↳ r_3 final = r_2 finals



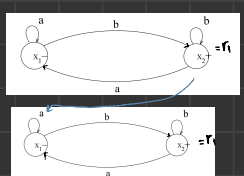
$r_3 = r_1 r_2$

Old states	a	b
$z_1^- \equiv \{x_1^-\}$	$z_1^- \equiv \{x_1^-\}$	$z_2 \equiv \{x_2^+, y_1^-\}$ (r1 final, r2 initial)
$z_2 \equiv \{x_2^+, y_1^-\}$	$z_3 \equiv \{x_2^-, y_2\}$	$z_2 \equiv \{x_2^+, y_1^-\}$
$z_3 \equiv \{x_2^-, y_2\}$	$z_4 \equiv \{x_2^-, y_3^+\}$	$z_2 \equiv \{x_2^+, y_1^-\}$
$z_4^+ \equiv \{x_2^-, y_3^+\}$	$z_4 \equiv \{x_2^-, y_3^+\}$	$z_5 \equiv \{x_2^+, y_2^-, y_3^+\}$
$z_5^+ \equiv \{x_2^+, y_2^-, y_3^+\}$	$z_6 \equiv \{x_2^-, y_2, y_3^+\}$	$z_5 \equiv \{x_2^+, y_2^-, y_3^+\}$
$z_6^+ \equiv \{x_2^-, y_2, y_3^+\}$	$z_4 \equiv \{x_2^-, y_3^+\}$	$z_5 \equiv \{x_2^+, y_2^-, y_3^+\}$



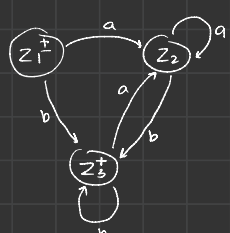
3. closure of an FA

↳ r^*
↳ make transition table
↳ r_1 final state = r_1 initial state
↳ r_3 initial = r_3 final ±
↳ take $z_1 = \{r_1 \text{ initial}^\pm\}$
↳ r_3 final = r_1 finals



$r_3 = r_1^*$

Old states	a	b
final → $z_1^\pm \equiv \{x_1^\pm\}$	$z_2 \equiv \{x_1^-\}$	$z_3 \equiv \{x_2^+, x_1^-\}$ (r1 final, r1 initial)
non final ← $z_2 \equiv \{x_1^-\}$	$z_3 \equiv \{x_1^-\}$	$z_3 \equiv \{x_2^+, x_1^-\}$
$z_3^+ \equiv \{x_2^+, x_1^-\}$	$z_2 \equiv \{x_1^-\}$	$z_3 \equiv \{x_2^+, x_1^-\}$

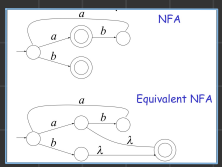


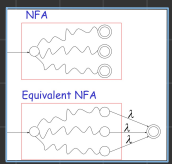
LECTURE 8

Properties of Regular Languages

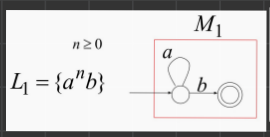
- ↳ Union $L_1 \cup L_2$
- ↳ Concatenation $L_1 L_2$
- ↳ Star L_1^*
- ↳ Reversal L_1^R
- ↳ Complement $\overline{L_1}$
- ↳ Intersection $L_1 \cap L_2$

Any NFA converted to equivalent NFA with single final state

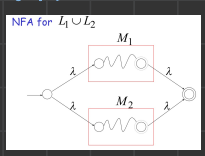


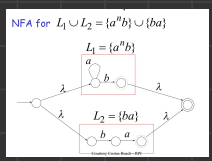


Show Regular Languages closed under

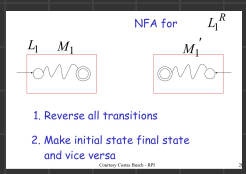


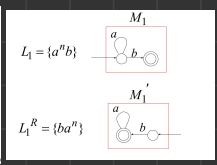
Union



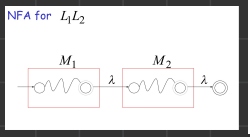


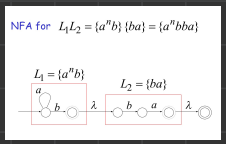
Reversal



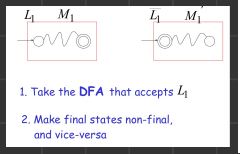


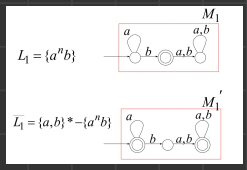
Concatenation



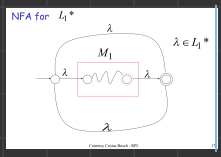


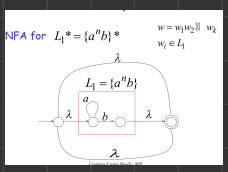
Complement



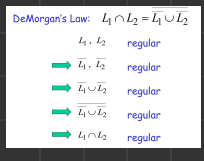


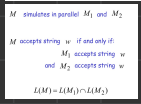
Star





Intersection





LECTURE 9

MINIMISING DFA's by PARTITION

Why Minimise DFA's

↳ Efficiency

↳ Better understanding of the language

↳ Compare similarity b/w the two FA's

↳ Determine if two DFA's recognise the same language

Minimization Process

↳ Remove inaccesible states

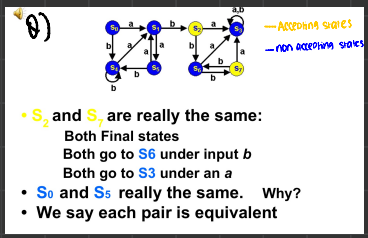
↳ Group equivalent states

↳ 2 states equal if all behaviors same

Minimization Steps

1. Places states of DFA into 2 equivalent classes
   - ↳ final states
   - ↳ non final states

   } 2 partitions
2. Repeat until no more change
   - ↳ for each state in same partition
     - if transition diff than those in it's group
     - Partition to new eqivalence class



1 final, non final partitions

	a	b
S_0	S_1	S_4
S_1	S_0	S_5
S_3	S_3	S_3
S_4	S_1	S_4
S_5	S_1	S_4
S_6	S_3	S_7
*S_2	S_3	S_6
*S_7	S_3	S_6

S_1, S_6 born end on final states

	a	b
S_0	S_1 I	S_4 I
S_1	S_0 I	S_5 II
S_3	S_3 I	S_3 I
S_4	S_1 I	S_4 I
S_5	S_1 I	S_4 I
S_6	S_3 I	S_7 II
*S_2	S_3 I	S_6 I
*S_7	S_3 I	S_6 I

S_0, S_4, S_5 same partition unlike S_1

	a	b
S_0	S_1 III	S_4 I
S_3	S_3 I	S_3 I
S_4	S_1 III	S_4 I
S_5	S_1 III	S_4 I
S_1	S_0 I	S_5 II
S_6	S_3 I	S_7 II
*S_2	S_3 I	S_6 III
*S_7	S_3 I	S_6 III

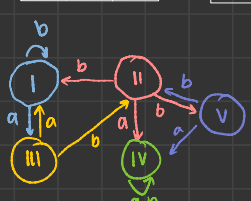
S_1, S_6 not same partion for a

	a	b
S_0	S_1 III	S_4 I
S_4	S_1 III	S_4 I
S_5	S_1 III	S_4 I
S_3	S_3 IV	S_3 IV
S_1	S_0 I	S_5 II
S_6	S_3 IV	S_7 II
*S_2	S_3 IV	S_6 III
*S_7	S_3 IV	S_6 III

	a	b
S_0	S_1 III	S_4 I
S_4	S_1 III	S_4 I
S_5	S_1 III	S_4 I
S_3	S_3 IV	S_3 IV
S_1	S_0 I	S_5 II
S_6	S_3 IV	S_7 II
*S_2	S_3 IV	S_6 V
*S_7	S_3 IV	S_6 V

	a	b
I	III	I
II	IV	V
III	I	II
IV	IV	IV
V	IV	II

5 partitions in order

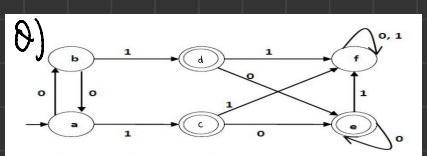


MINIMISING DFA's by Myphill-Nerode Theorem TF ALGORITHM

Minimization Steps

1. Draw table for all pairs of state P,Q
2. Mark every pair in DFA where P→final, Q→not final, vice versa
   ↳ dosent have to be connected
3. Repeat until unable to mark more
   ↳ check for unmarked pairs $\delta(Q,x), \delta(P,x)$
   UM ↳ unmarked pair → leave it as is
   DE ↳ pair dosent exist → leave it as is
   M ↳ marked pair → mark Q,P
4. Combine unmarked pairs and make single state

Q)



Step 1, 2

a	a	b	c	d	e	f
b						
c	✓	✓				
d	✓	✓				
e	✓	✓				
f	✓	✓	✓	✓	✓	

Step 3

unmarked Pairs

(a,b) → $\delta(a,0)=b$ UM, $\delta(a,1)=c$ UM
$\delta(b,0)=a$ $\delta(b,1)=d$

(c,d) → $\delta(c,0)=e$ DE, $\delta(c,1)=f$ DE
$\delta(d,0)=e$ $\delta(d,1)=f$

(e,c) → $\delta(e,0)=e$ DE, $\delta(e,1)=f$ DE
$\delta(c,0)=e$ $\delta(c,1)=f$

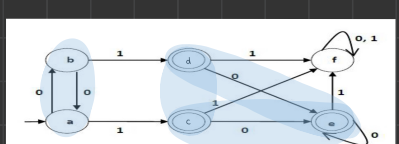
(e,d) → $\delta(e,0)=e$ DE, $\delta(e,1)=f$ DE
$\delta(d,0)=e$ $\delta(d,1)=f$

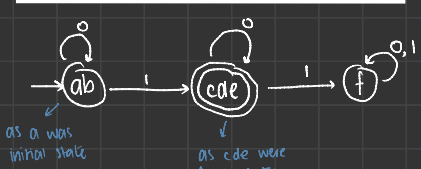
(f,a) → $\delta(f,0)=f$ UM, $\delta(f,1)=f$ M
$\delta(a,0)=b$ $\delta(a,1)=c$ M
so mark f,a

(f,b) → $\delta(f,0)=f$
$\delta(b,0)=a$ M

Step 4

(a,b) (c,d) (e,c) (e,d)





LECTURE 10

PUMPING LEMMA

Used to prove that a language is not regular by contradiction

Pumping Lemma steps

1. Assume A is a regular language → for contradiction
2. Pumping length = ... → assume
3. Find string 's' in A such that $|s| \geq P$ → assume
4. Divide S into x y z
5. Consider cases where S can be divided to xyz
6. Show 3 Pumping conditions unsatisfied
   - ↳ 1. $xy^iz \in A$ for every $i \geq 0$
   - ↳ 2. $|y| > 0$
   - ↳ 3. $|xy| \leq P$
7. S cannot be pumped = CONTRADICTION

CAN NOT BE USED TO PROVE A LANGUAGE IS REGULAR

0) Prove $A = \{a^n b^n \mid n \geq 0\}$ is not regular

↳ Assume A is regular → for contradiction

Pumping length = P

$S = a^P b^P$ taking P = 7

x y z

aaaaaaa bbbbbbb

case 1: y is in the 'a' part

aaaaaaa bbbbbbb

x y z

$xy^2z \Rightarrow xy^2z$

aa aaaaaaaa abbbbbbb

(repeat 2 for y^2)

a = b → as a^nb^n

1. $11 \neq 7$ ✗
2. $|y| = 4$ ✓
3. $|xy| = 6$ P = 7 ✓

case 2: y is in the 'b' part

aaaaaaa bbbbbbb

x y z

$xy^2z \Rightarrow xy^2z$

aaaaaaa bb bbbbbbbb b (y^2)

a = b

3. $7 \neq 11$ ✗
2. $|y| = 4$ ✓
3. $|xy| = 13$ P = 7 ✗

case 3: y is in the 'a' and 'b' part

aaaaa aabbbbbbb

x y z

$xy^2z \Rightarrow xy^2z$

aaaaa aabb aabb bbbbb (y^2)

Pattern should be a's followed by b's

1. Pattern not followed ✗
2. $|y| = 4$ ✓
3. $|xy| = 9$ P = 7 ✗

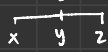
HENCE NOT REGULAR

Q) Prove $A = \{ yy \mid y \in \{0,1\}^* \}$ is not regular

Assume A is regular

Pumping length = P

$S = 0^P 1 0^P 1$ → choose any string



taking P = 7 → choose any y

case 1: y is in the '0' part

y^2

00 0000 0000 0100000001

1. doesn't follow pattern yy ✗
   - start and end pattern should be the same ✓
2. $|y| = 4$ ✓
3. $|xy| > 6$ P = 7 ✓

Hence not regular

DFA WITH OUTPUTS

Output → depends on input / doesn't depends on input

Mealy State Machines

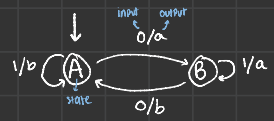
↳ Output depends on current state and input

↳ inputs = outputs

↳ no final state

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

- Q: finite set of states
- Σ: inputs
- δ: transition function $Q \times \Sigma \to Q$
- λ: output function $\Sigma \times Q \to \Delta$
- q_0: initial state

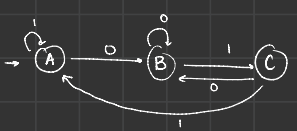


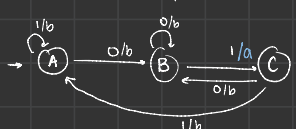
e.g. 1010

Q) Construct mealy machine that produces a when 01 is input string

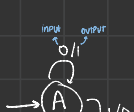
$\Sigma = \{0,1\}$

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





Q) Construct mealy machine that produces 1's complement



1 0 1 0 0

0 1 0 1 1

Moore State Machines

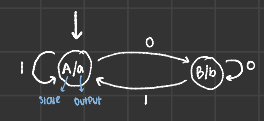
↳ Output depends on current state

↳ inputs = outputs + 1

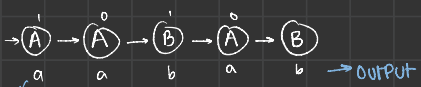
↳ no final state

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

- Q: finite set of states
- Σ: inputs
- δ: transition function $Q \times \Sigma \to Q$
- λ: output function $Q \to \Delta$
- q_0: initial state



e.g. 1010

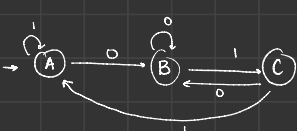


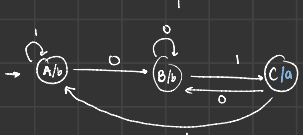
↙ directly comes without any input

Q) Construct moore machine that produces a when 01 is input string

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

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





context free languages

context free grammars

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

- V: variables
- T: terminal symbols
- S: start variable
- P: production rule
  - $A \to a$
  - $a \in \{V \cup \Sigma\}^*$
  - $A \in V$

Q) generate equal no of a's and b's in the form $a^n b^n$ (not regular lang)

$G = \{ (S,A), (a,b), (\underbrace{S \to aAb,\ A \to aAb \mid \epsilon}_{P}) \}$

ε: epsilon, empty symbol

$S \to a\underline{A}b$ (replace $A \to aAb$) $V = \{S, A\}$

$\to aa\underline{A}bb$ (replace $A \to aAb$) $T = \{a, b\}$

$\to aaa\underline{A}bbb$ (replace $A \to \epsilon$)

$\to aaabbb$

$\to a^3b^3 \to a^nb^n$

NOTATION

- beg lower case letters: a,b,c ... Terminal symbols
- end lower case letters: w,x,y... strings of terminals
- beg upper case letters: A,B,C ... variables
- end upper case letters: X, Y, ... Terminals/strings
- lower case Greek letters: α,β,γ... Strings of Terminals/variables

Derivation order

1. $S \to AB$ 2. $A \to aaA$ 3. $A \to \lambda$
4. $B \to Bb$ 5. $B \to \lambda$

↳ Leftmost derivation order of string

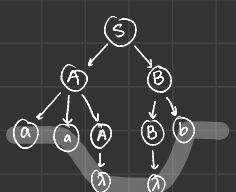
$S \xrightarrow{1} AB \xrightarrow{2} aaAB \xrightarrow{3} aaB \xrightarrow{4} aaBb \xrightarrow{5} aab$

↳ Rightmost derivation order of string

$S \xrightarrow{1} AB \xrightarrow{4} ABb \xrightarrow{3} Ab \xrightarrow{2} aaAb \xrightarrow{3} aab$

Derivation Trees

$S \to AB$ $A \to aaA \mid \lambda$ $B \to Bb \mid \lambda$



= aab

↳ when 2 or more derivation trees then context free grammar is ambiguous

is this coming?

ishma hafeez notes