AFLL

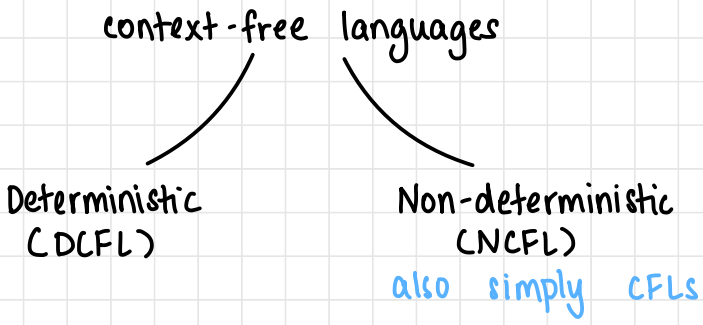
UNIT - 4

CLASS NOTES

feedback/corrections: vibha@pesu.pes.edu

Vibha Masti

CLOSURE PROPERTIES OF CONTEXT-FREE LANGUAGES



Closure Property

A set is closed under an operation if the operation can always be completed with elements in the set

CLOSURE PROPERTIES OF CFLs

Closed under
- union
- concatenation
- Kleene closure
- Reversal

Not closed under
- intersection
- complement

CLOSURE PROPERTIES

UNION

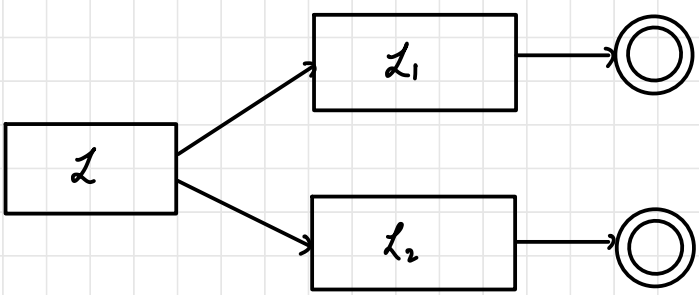
If L_1 is a CFL and L_2 is a CFL, then $L_1 \cup L_2$ is a CFL

- or operation ($\mathcal{L}_1$ or $\mathcal{L}_2$)

Question 1

$\mathcal{L}_1 = \{ a^n b^n c^m \mid m \geq 0 \text{ and } n \geq 0 \}$
$\mathcal{L}_2 = \{ a^n b^m c^m \mid m \geq 0 \text{ and } n \geq 0 \}$

Show that they are closed under union as CFLs



$S \rightarrow S_1 \mid S_2$
$S_1 \rightarrow (\mathcal{L}_1)$
$S_2 \rightarrow (\mathcal{L}_2)$

context free

CONCATENATION

If L_1 is a CFL and L_2 is a CFL, then $L_1 L_2$ is a CFL

$$S \rightarrow S_1 S_2$$
$$S_1 \rightarrow (\mathcal{L}_1)$$
$$S_2 \rightarrow (\mathcal{L}_2)$$

Question 2

$$\mathcal{L}_1 = \{a^n b^n \mid n \geq 0\}$$
$$\mathcal{L}_2 = \{c^m d^m \mid m \geq 0\}$$

$$\mathcal{L} = \mathcal{L}_1 \mathcal{L}_2 = \{a^n b^n c^m d^m \mid n \geq 0, m \geq 0\}$$

PDA:

- push a's to stack & pop b's once b reached
- once z_0 reached, push c's to stack until a d is seen
- resultant PDA $\longrightarrow$ CFL

KLEENE CLOSURE

If $\mathcal{L}$ is a CFL, $\mathcal{L}^*$ is a CFL

Question 3

$$\mathcal{L} = \{a^n b^n \mid n \geq 0\}$$

$$\mathcal{L}^* = \{(a^n b^n)^* \mid n \geq 0\}$$

$$S \rightarrow AS \mid \lambda$$
$$A \rightarrow aAb \mid \lambda$$

REVERSAL

If $\mathcal{L}$ is a CFL, $\mathcal{L}^R$ is a CFL

NON-CLOSURE PROPERTIES

INTERSECTION

- Intersection of a CFL with an RG is a CFL
- Intersection of a CFL with a CFL is not a CFL

Question 4

$$\mathcal{L}_1 = \{a^m b^n c^n \mid m,n \geq 0\}$$
$$\mathcal{L}_2 = \{a^n b^n c^m \mid m,n, \geq 0\}$$

$\mathcal{L}_1 \cap \mathcal{L}_2 = \{a^n b^n c^n \mid n \geq 0\}$ → not context free
cannot construct PDA

RE and CFL

- cross product of PDA and DFA
- run paths parallelly from single start state
- if both reach final state, accepted

COMPLEMENT

$\mathcal{L}_1 \rightarrow$ CFL , $\overline{\mathcal{L}_1} \rightarrow$ not CFL

- According to DeMorgan's Law, $\mathcal{L}_1 \cap \mathcal{L}_2 = \overline{\overline{\mathcal{L}_1} \cup \overline{\mathcal{L}_2}}$
- Assume complement to be closed and let $\mathcal{L}_1$ & $\mathcal{L}_2$ be CFLs
- Then, $\overline{\overline{\mathcal{L}_1} \cup \overline{\mathcal{L}_2}}$ should also be a CFL. However, the LHS $\mathcal{L}_1 \cap \mathcal{L}_2$ is not a CFL.
- Therefore, complement is not closed under CFLs

CLOSURE PROPERTIES OF DCFLS

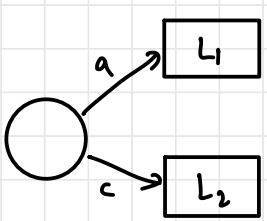
- only closed under complement
- sure about when to push/pop (single path)

NON-CLOSURE PROPERTIES

UNION

$\mathcal{L}_1 = \{a^n b^n \mid n \geq 0\}$
$\mathcal{L}_2 = \{c^m d^m \mid m \geq 0\}$

$\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2$ is DCFL

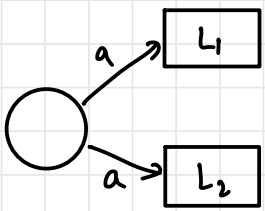


DCFL CLOSED

Question 5

$$L_1 = \{a^n b^m c^m \mid m, n \geq 0\}$$
$$L_2 = \{a^n b^n c^m \mid m, n \geq 0\}$$

$$L = L_1 \cup L_2$$



not DCFL NOT CLOSED

- Not necessarily closed

INTERSECTION

- Not necessarily closed under intersection

$$L_1 = a^n b^n c^m$$
$$L_2 = a^n b^m c^m$$

$$L = L_1 \cap L_2 \longrightarrow \text{not DCFL}$$

CONCATENATION

- Not necessarily closed under intersection

$$L_1 = \{a^n b^n \mid n \geq 0\}$$
$$L_2 = \{c^m d^m \mid m \geq 0\}$$

$L = L_1 L_2$ is DCFL can make PDA

Question 6

$$L_1 = \{a^n b^m \mid m < n\}$$
$$L_2 = \{w c w^R\}$$

$$L = L_1 L_2 = \{a^n b^m w c w^R\}$$

for example $w = abb \uparrow bbbacabbb$

don't know to start w from here

non-determinism

L is not a DCFL

REVERSAL

If L is a DCFL, L^R is not necessarily a DCFL

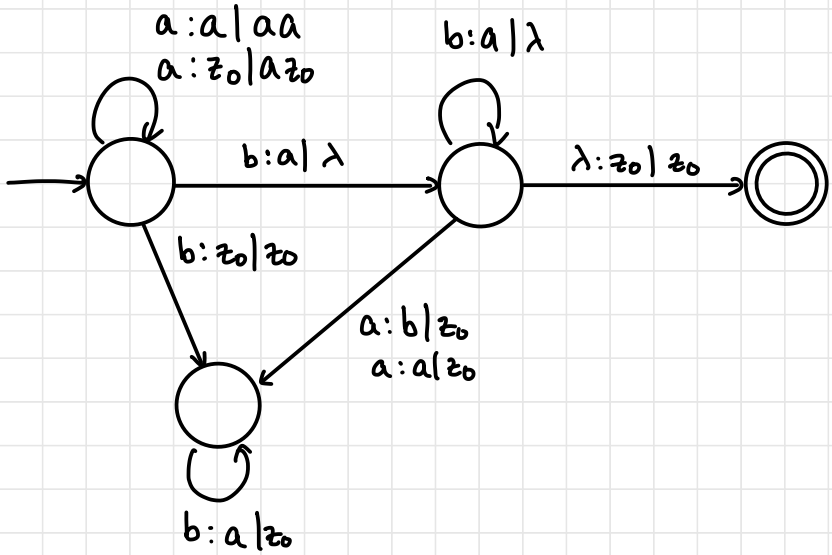
CLOSURE PROPERTIES

COMPLEMENT

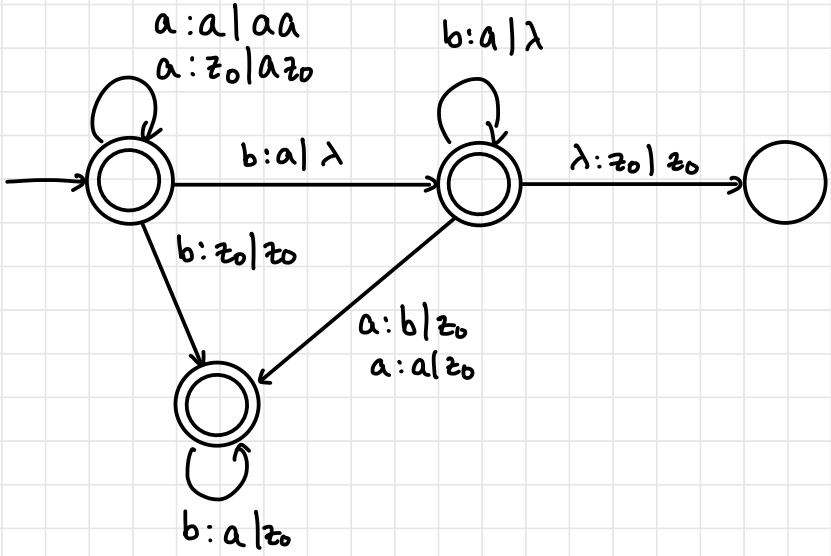
$L_1 \rightarrow$ DCFL , $\overline{L_1} \rightarrow$ DCFL (closed under complement)

Question 7

$$L = \{a^n b^n \mid n \geq 0\}$$



- flipped states



ANSWERING QUESTIONS ABOUT CFLS

1. Is a CFL empty?

- if start state S is non-generating

  S → non generating

- if start symbol is non-terminating

  S → aA | bB | C
  A → aA | Aa | bB
  B → bB | Ba | aA
  C → bA | aB | aSb | bSa

  non-terminating

- emptiness is a decidable property

2. Is a CFL finite?

- if there is a loop, the language is infinite

(a) S → aSb | λ infinte

(b)
S → aA | bB | C
A → Ab | bB
B → bB | aS

infinite

- decidable

3. Is a string w a member of the CFL?

- derivable strings → can use CYK algorithm
- decidable

4. Are two CFLs equal?

- no algorithm to prove equality of CFLs
- algorithm exists for RLs (if start states indistinguishable)

5. Is the CFL ambiguous?

- not decidable; no algorithm

PUMPING LEMMA

- Have to consider state as well as the stack contents
- Pumping lemma cannot be used in the same way as a regular language
- Work with CNF grammars and we work with long strings greater than number of non-terminals in the grammar
- Unlike RLs where we focus on number of states
- Divide into 5 parts
- Only can prove if L is not CFL (contradiction)

Question 8

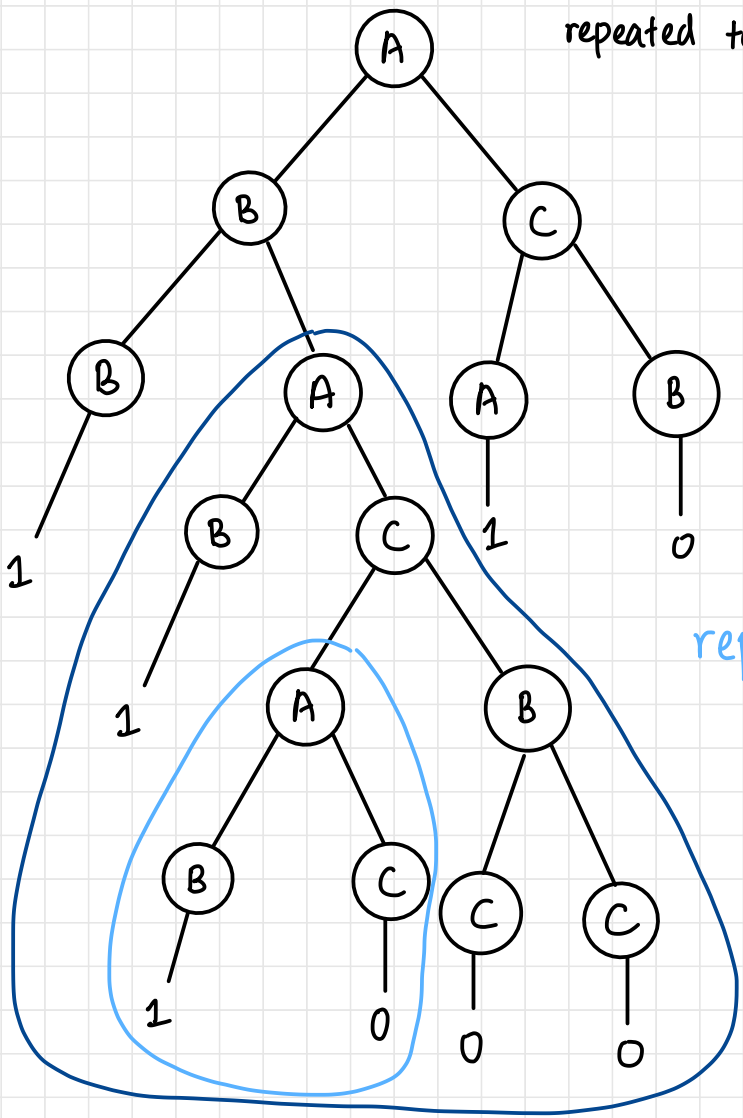
$A \rightarrow Bc \mid 0 \mid 1$
$B \rightarrow BA \mid CC \mid 1 \mid 0$
$C \rightarrow AB \mid 0$

pumping using parse tree

non-terminals = 3
$|w| \geq 2^3 = 8$

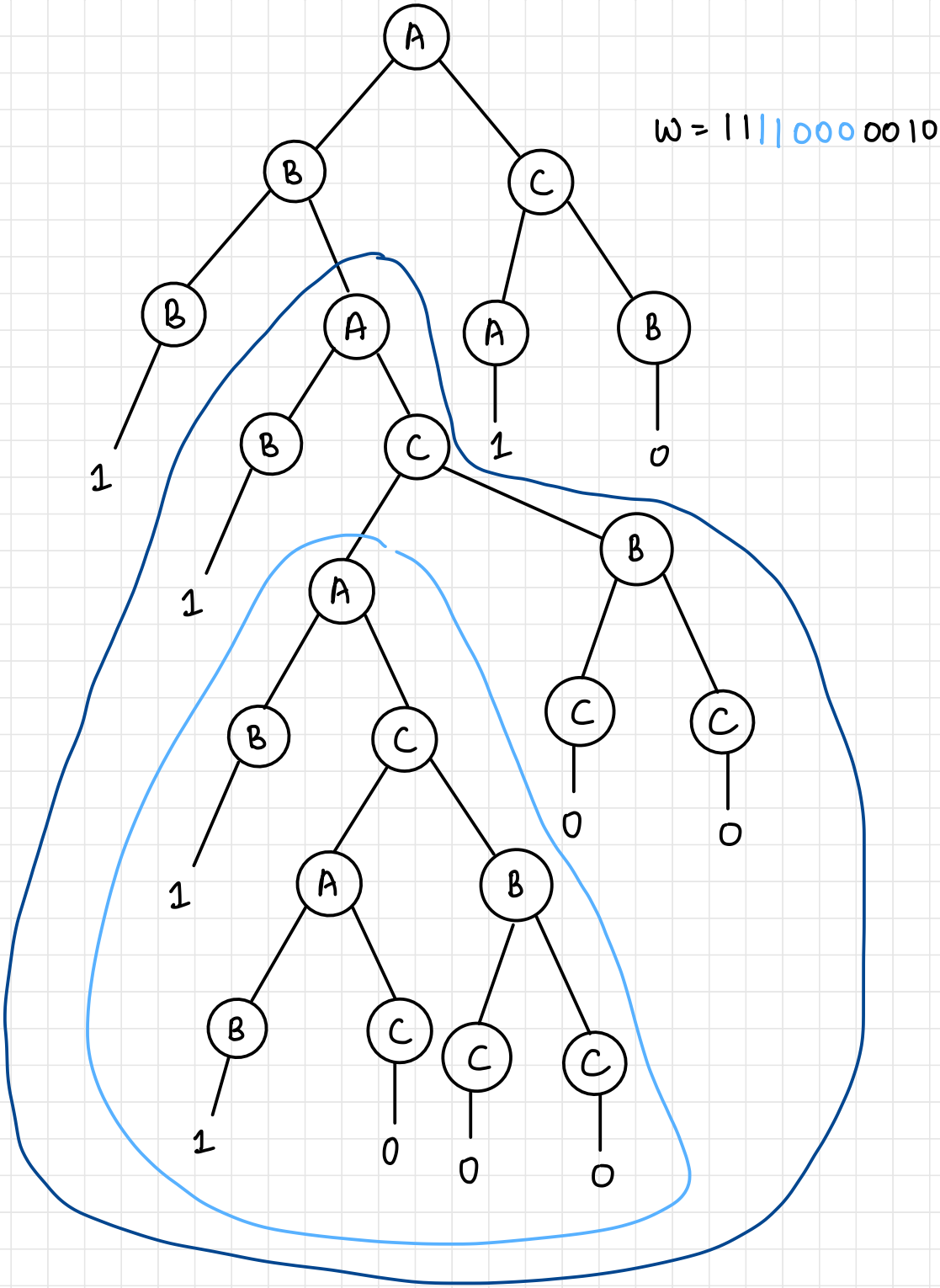
$w = 11100010$ parse tree

repeated terminal ⟶ infinite



replace inner A with outer subtree

$w = 11100010$


w = 11|1|000|00|10
A
B
C
B
A
A
B
1
B
C
1
0
1
A
B
C
B
C
C
0
0
1
A
B
B
C
C
C
1
0
0
0

$w = \underbrace{1}_{u}\underbrace{1}_{v}\underbrace{10}_{w}\underbrace{00}_{x}\underbrace{10}_{y}$ first iteration

$w = \underbrace{1}_{u}\underbrace{11}_{v}\underbrace{10}_{w}\underbrace{0000}_{x}\underbrace{10}_{y}$ second iteration

$w = \underbrace{1}_{u}\underbrace{111}_{v}\underbrace{10}_{w}\underbrace{000000}_{x}\underbrace{10}_{y}$ third iteration

- only v & x keep changing

$|word| \geq n$ n = no. of variables

$|vwx| \leq m$

$|vx| \geq 1$

- note: not used to prove $\mathcal{L}$ is CFL; only used when we cannot construct a PDA

Question 9

$\mathcal{L} = \{a^n b^n\}$

word → $w = a^{20} b^{20}$ $m = 20$

word, not segment w → $|w| \geq m$

$\underbrace{a^{19}}_{u}\underbrace{a}_{v}\underbrace{\lambda}_{w}\underbrace{b}_{x}\underbrace{b^{19}}_{y}$

Question 10

$$\mathcal{L} = \{a^n b^n c^n \mid n \geq 0\}$$

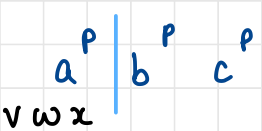
- Assume $a^n b^n c^n$ is a CFL
- Let $S = a^p b^p c^p$ p = pumping constant/ no. of variables

$$|S| \geq p$$
$$|vwx| \leq p$$
$$|vx| \geq 1$$

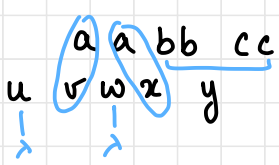
How to split

Case 1

- vwx is made only of a's



- vx together must contain at least one a



Case 2

- vwx is made only of b's

$$\underset{u}{a^p}\quad \underset{vwx}{b^p}\quad \underset{y}{c^p} \qquad \notin L$$

Case 3

- vwx is made only of c's

$$\underbrace{a^p\ b^p}_{u}\quad \underset{vwx}{c^p}\quad \underset{y}{\lambda} \qquad \notin L$$

Case 4

- vwx is made of some a's and some b's

$$\underset{u}{a^{p-2}}\ \underset{v}{a}\ \underset{w}{ab}\ \underset{x}{b}\ \underset{y}{b^{p-2}c^p} \qquad \notin L$$

Case 5

- vwx is made of some b's and some c's

$$\underset{u}{a^p b^{p-2}}\ \underset{v}{b}\ \underset{w}{bc}\ \underset{x}{c}\ \underset{y}{c^{p-2}} \qquad \notin L$$

- cannot span over all (a,b,c) as $|vwx| \leq p$

Question 11

$$L = \{ ss \mid s \in \{a,b\}^* \}$$

- cannot construct PDA (only for ss^R)
- Let $s = a^m b\, a^m b$, m = no. of variables
- $|s| \geq m$ ✓
- divide into u, v, w, x, y

Placement of u, v, w

Case 1

- vwx contains only a's

$$v^i w x^i \quad \text{for } i = 2$$

$$\underset{u}{\lambda} \quad \underset{v}{a}\, \underset{w}{a^{m-2}}\, \underset{x}{a} \quad b\, \underset{y}{a^m}\, b \qquad \notin L$$

- pump in, more a's in the first part

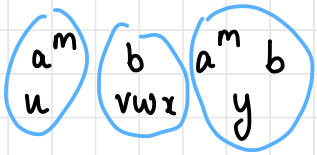
Case 2

- vwx contains both a's and b's

$$\underset{u}{a^{m-1}}\, \underset{vwx}{ab}\, \underset{y}{a^m}\, b \qquad \notin L$$

Case 3

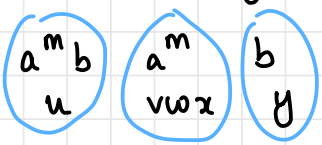
- vwx contains only b's



$\notin L$

Case 4

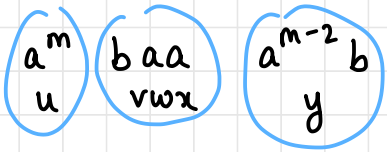
- vwx contains only a's



$\notin L$

Case 5

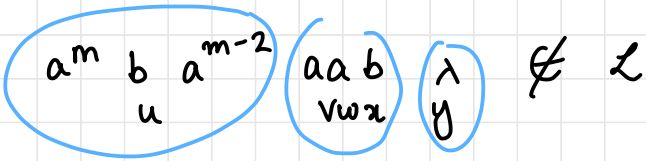
- vwx contains b's and a's



$\notin L$

Case 6

- vwx contains both a's and b's



$\notin L$

Question 12

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

Assume L is CFL, p = no. of variables in CNF

w, $|w| \geq p$

$uvxyz = w$

loop

$|vxy| \leq p$ cannot span entire string

$|vy| \geq 1$

$w = a^p b^p c^p$

Case 1

$\underbrace{a^p}_{uvxy} \mid \underbrace{b^p a^p}_{z}$

$a^{p+2} b^p a^p \notin L$

Case 2

$\underbrace{a^p}_{u} \quad \underbrace{b^p}_{vxy} \quad \underbrace{a^p}_{z}$

$a^p b^{p+2} a^p \notin L$

Case 3

$a^p b^p | a^p | \lambda \quad \notin L$

$u \quad vxy \quad z$

Case 4

$a^p b^p a^p$

$aaa \ldots a\,ab\,b \ldots bbaa \ldots aa$

$u \quad vxy \quad z$

Case 5

$a^p b^p a^p$

$aa \ldots aa\,bb \ldots bb\,aa \ldots aa$

$u \quad vxy \quad z$

Question 13

$L = \{a^n b^m c^n d^{n+m} \mid n, m \geq 0\}$

Assume L is CFL, p = no. of variables in CNF

$w, \quad |w| \geq p$

$uvxyz = w$

loop

$|vxy| \leq p$

$|vy| \geq 1$

$w = a^p b^q c^p d^{p+q}$

Case 1

vxy in only a's $\longrightarrow$ $n_a(w) \neq n_c(w)$

Case 2

$\underbrace{aa\ldots}_{u}\underbrace{aabb}_{vxy}\underbrace{\ldots bb\,cc\ldots cc\,dd\ldots dd}_{z}$

- d will not match n+m
- $a \neq c$

Case 3

only b's $\longrightarrow$ d's will not match

Case 4

bc $\longrightarrow$ d's not matched

Case 5

only c's $\longrightarrow$ a & c not match

Case 6

vxy over (abc) $\longrightarrow$ ruled out

Case 7

vxy over cd $\longrightarrow$ $a \neq c$

Case 8

only d's $\longrightarrow$ won't be n+m

* Question 14

$$L = \{a^{n^2} \mid n \geq 0\}$$

perfect squares

$$L = \{\lambda, a, a^4, a^9, a^{16}, \ldots\}$$

Assume L is CFL, k = no. of variables in CNF (pumping constant)

w, $|w| \geq k$

$uvxyz = w$

loop

$|vxy| \leq k$

$|vy| \geq 1$

Let $n = k^2$

Let $w = a^{k^4}$

$\forall\ i \geq 0$, $w \in L$, $w = uv^i x y^i z$

w contains only a's

$vy \rightarrow a^p$ (pump p times)

$w = a^{k^4+p} = s$ (new string)

- to get the next perfect square of $n^2 \rightarrow (n+1)^2$
- for a^{k^4}, next string is $a^{(k^2+1)^2}$

$$a^{(k^4 + 2k^2 + 1)}$$

if $p = 2k^2+1$, only then the language is accepted

$\therefore$ $|vy|$ would be $2k^2+1$

but $|vy| \leq k$

* Question 15

$L = \{a^{n!} \mid n \geq 0\}$

$w = a^{m!}$

$u\ v\ x\ y\ z$ ($m!$)

$v = a^{k_1}$

$y = a^{k_2}$

$ak_1 + ak_2 = k$

$u\ v\ x\ y\ z$

$m! + ak_1 + ak_2 = m! + k$

$m! + k \leq m! + m \leq (m+1)!$

$a^{n!}$ not CFL

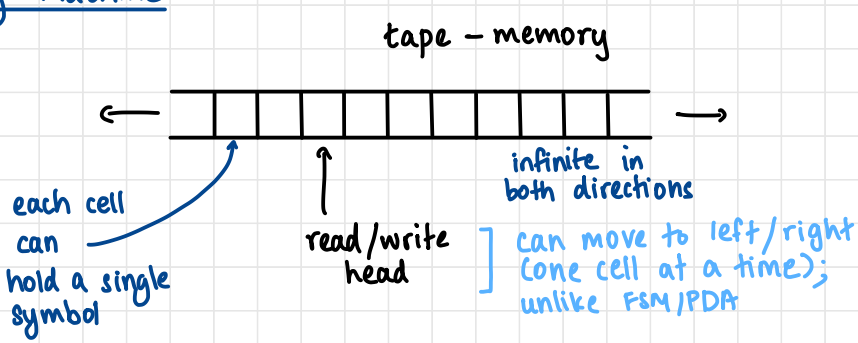
turing MACHINE

- More powerful than CFLs
- Languages — recursively enumerable languages
- Turing award — highest award given

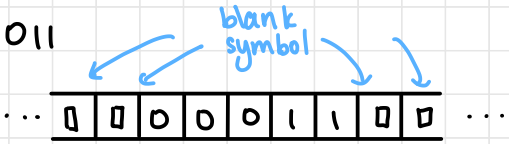
Alan Turing, 1936

- Created an abstract machine (paper on Computable Numbers — 1936)
- Computing done on paper using symbols
- Imagine paper to be tape divided into square boxes and the tape is an abstract machine
- Behaviour of a computer (person) determined by the symbols they are using and their state of mind
- Transition depends on what state one is in and what one is looking at on the tape

Turing Machine



For w = 00011



special symbols called blank symbols (not empty)

B or ␣ or □

- Any computation that a modern computer is capable of performing can be performed (computed) by a Turing Machine
- If a computation cannot be performed by a Turing Machine, then it cannot be performed by a modern computer
- Tape data structure acts as input, output and stores intermediate memory (infinite memory)

Formal Definition of Turing Machine

7-tuple, $M = (Q, \Gamma, \square, \Sigma, \delta, q_0, F)$

gamma (toe)

- Q — finite set of states
- Γ — tape alphabets
- $\square$ — blank symbol
- Σ — alphabet
- δ — transition function
- q_0 — start state
- F — set of final states

state, input symbol, read/write

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

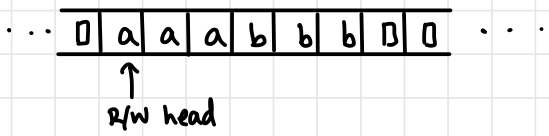
move to the left or right

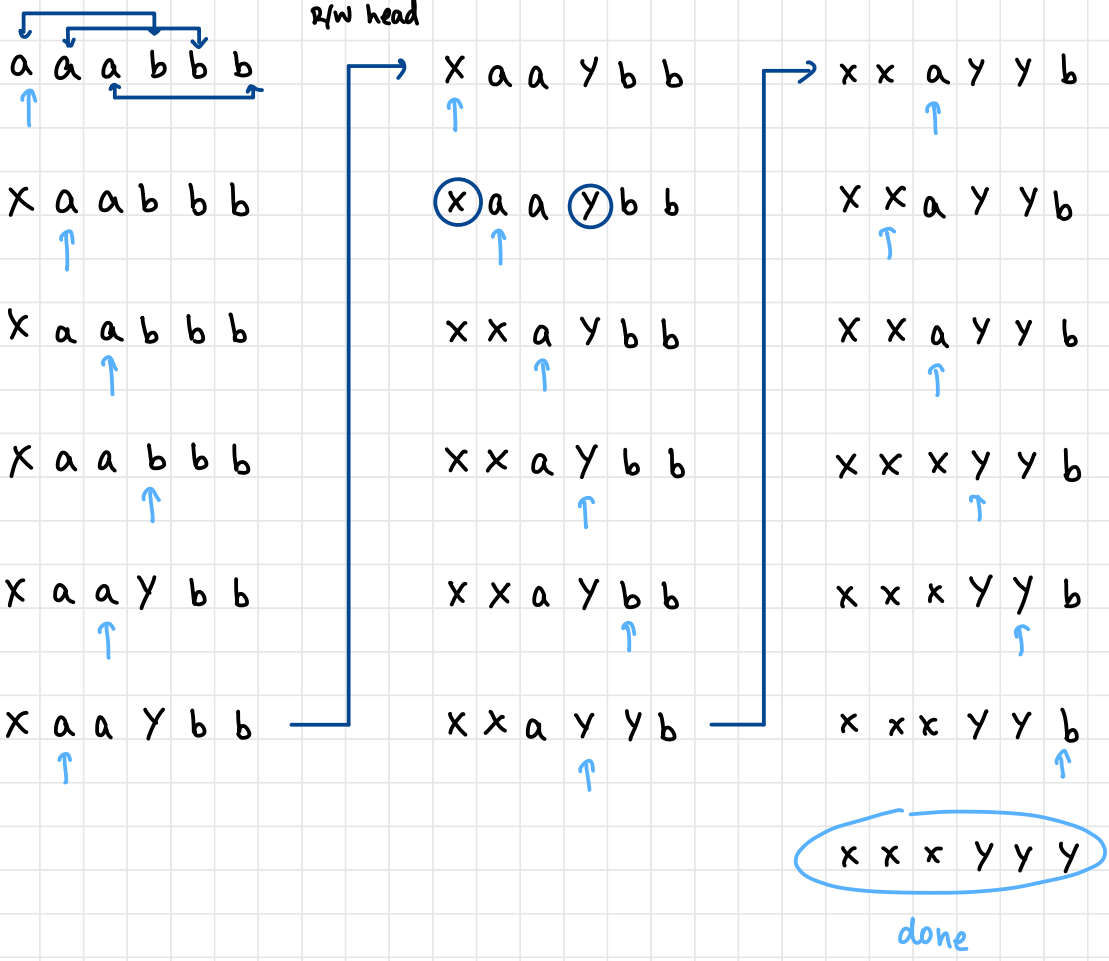
- TM is more powerful than FA and PDA and any string accepted by them is accepted by TM

Question 16

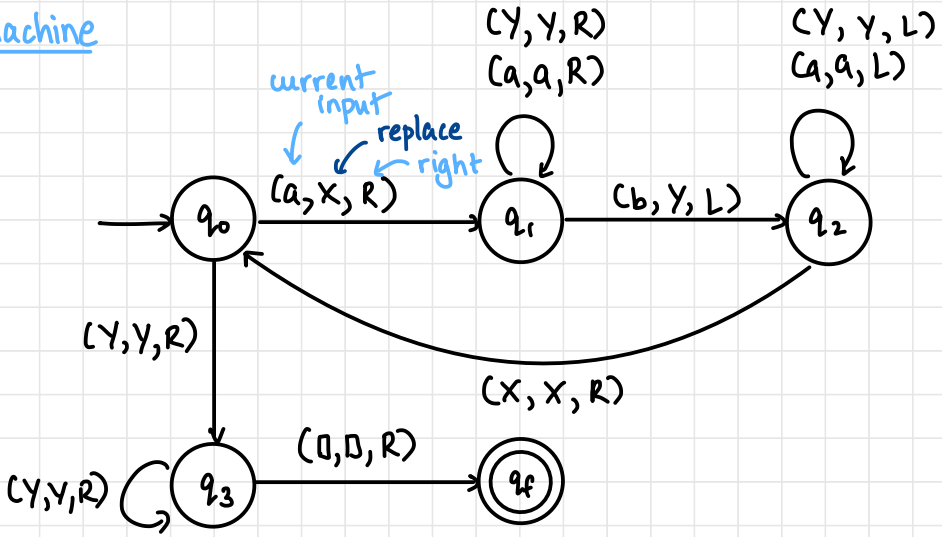
$\mathcal{L} = \{a^n b^n \mid n \geq 1\}$ show acceptance by TM

$w = aaabbb$





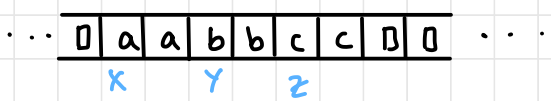
Machine



Question 17

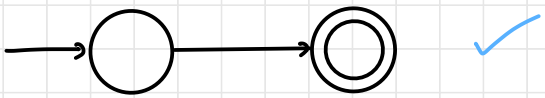
$L = \{a^n b^n c^n \mid n \geq 1\}$

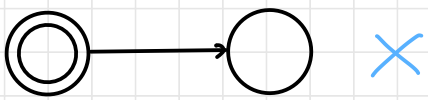
$w = aabbcc$



Acceptance

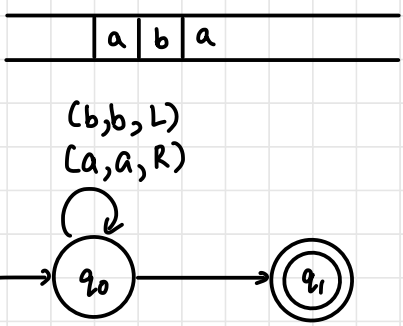
- No outgoing transition from final state





- If machine halts in a non-final state or enters an infinite loop, the string is rejected

Infinite loop



- If machine halts on final state, string is accepted

Turing Machine

Decider

- $\forall w \in \Sigma$
- all strings of a language
- either accepts all or rejects all

decidable or recursive algorithms

- Halt and accept
- Halt and reject

Acceptor

- $w \in L$
- particular string in a language
- only concerned about a single string

- Halt and accept
- Reject
  - halt
  - ∞ unpredictable

- Turing accepted machine
- Recursively enumerable language

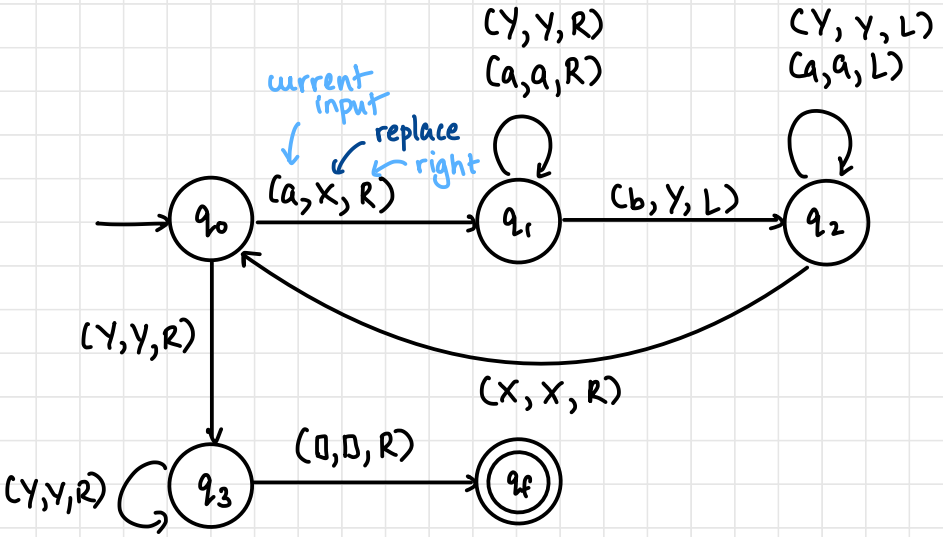
Continuation of Question 16

$$L = \{a^n b^n \mid n \geq 0\}$$

Transition function

$$\delta(q_0, a) = (q_0, X, R)$$
$$\delta(q_0, b) = \text{halt}$$
$$\delta(q_1, a) = (q_1, a, R)$$
$$\delta(q_1, b) = (q_2, Y, L)$$

write for X & Y also (all tape symbols)



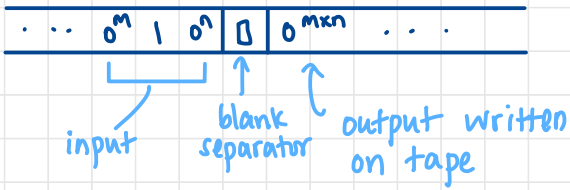
Transition Table

state	a	b	X	Y	□
→q_0	q_1, X, R	halt	halt	q_3, Y, R	halt
q_1	q_1, a, R	q_2, Y, L	halt	q_1, Y, R	halt
q_2	q_2, a, L	halt	q_0, X, R	q_2, Y, L	halt
q_3	halt	halt	halt	q_3, Y, R	$q_f, □, R$
q_f	halt	halt	halt	halt	halt

Question 18

$L = \{0^m 1 0^n \mid n, m \geq 1\}$ Language of Multiplication

i/p → o/p = $0^{m \times n}$



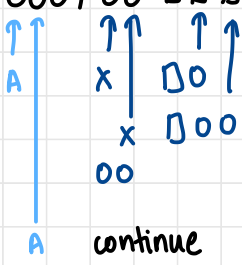
- Perform repeated addition
- w = 000 1 00

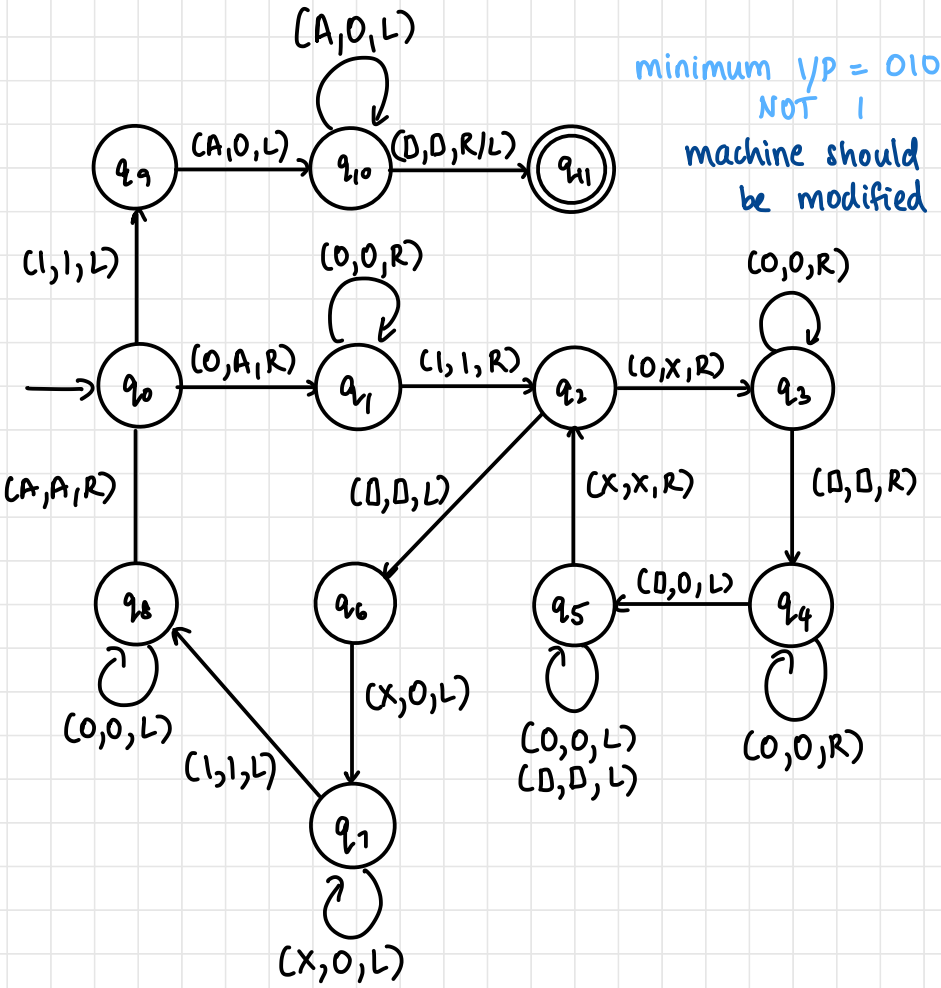
(00) x 3

1st 0 → print 00
2nd 0 → print 00
3rd 0 → print 00

000000 on tape

w = 0001 00





minimum I/P = 010
NOT 1
machine should be modified

w = 0 0 0 1 0 ⊔ ⊔ ⊔ ⊔ ⊔ ⊔

= A 0 0 1 X 0 ⊔ 0 ⊔ ⊔ ⊔ ⊔ ⊔

= A 0 0 1 X X ⊔ 0 0 ⊔ ⊔ ⊔ ⊔

= A 0 0 1 0 0 ⊔ 0 0 ⊔ ⊔ ⊔ ⊔

= A A 0 1 X 0 ⊔ 0 0 0 ⊔ ⊔ ⊔

= A A 0 1 X X ⊔ 0 0 0 0 ⊔ ⊔

= A A 0 1 0 0 ⊔ 0 0 0 0 ⊔ ⊔

= A A A 1 X 0 ⊔ 0 0 0 0 0 ⊔

= A A A 1 X X ⊔ 0 0 0 0 0 0

= A A A 1 0 0 ⊔ 0 0 0 0 0 0

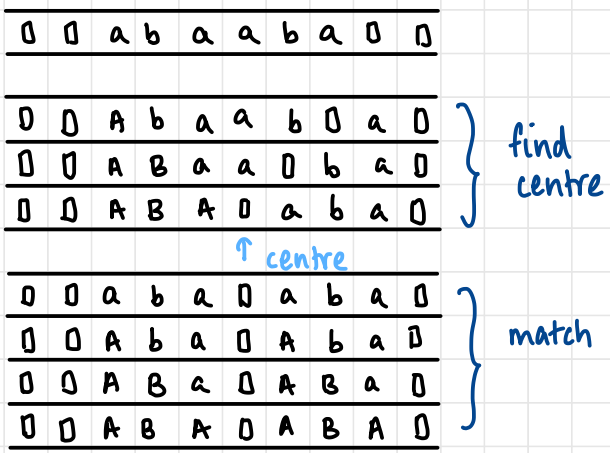
= 0 0 0 1 0 0 ⊔ 0 0 0 0 0 0

Question 19

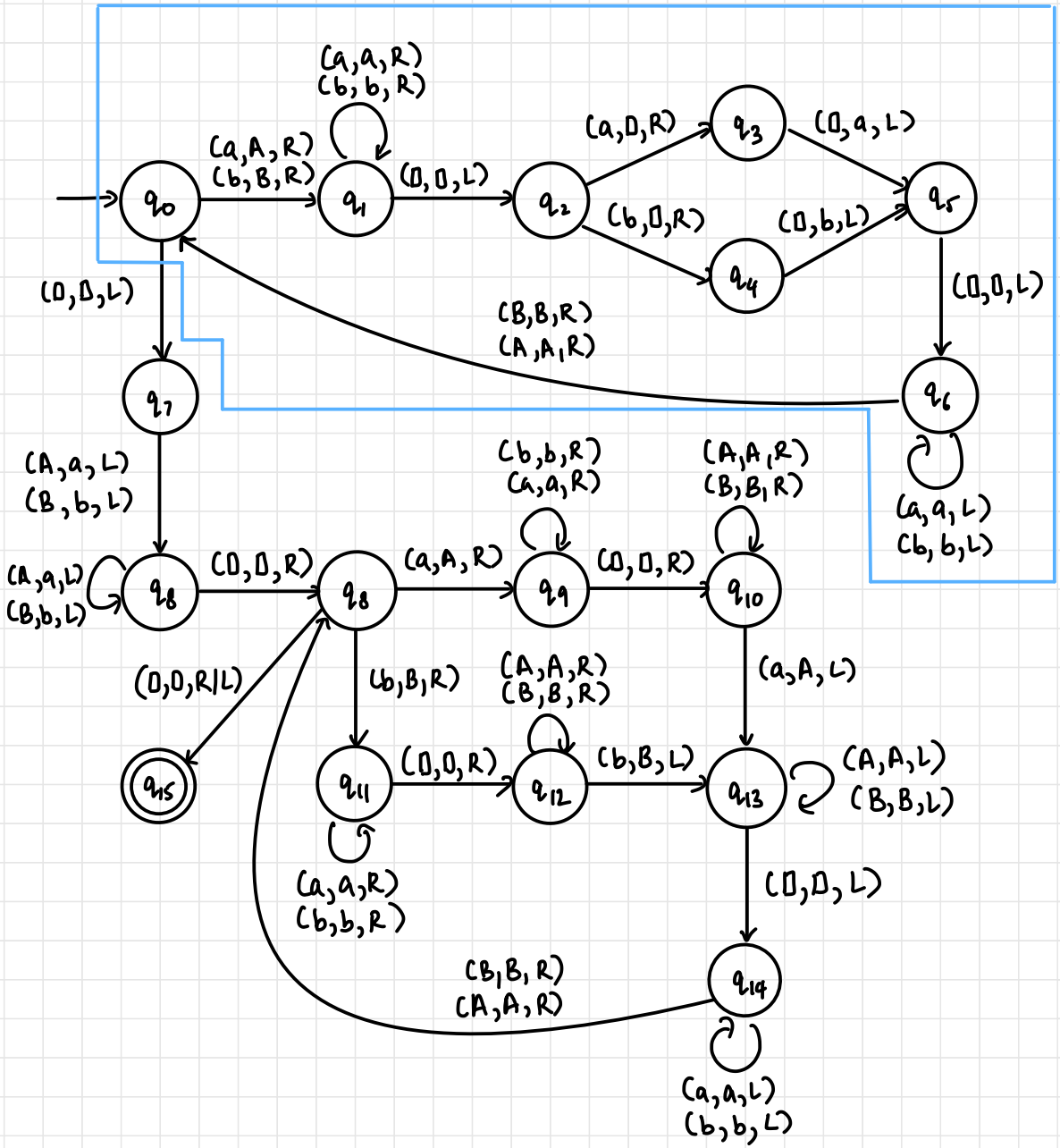
$$L = \{ww \mid w \in \{a,b\}^*\}$$

- write down logic & then construct – mandatory
- must find centre
- divide string of even length into 2 equal parts
- if length not even, halt
- match first half with second half

$w = abaaba$



odd strings will get stuck here



Question 20

Language of subtraction

write c's onto tape
- if a>b : c
- if b>a : -c

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

- #a > #b
- #a < #b
- #a = #b

#a's > #b's

copy # of c's [aabccc → should halt]

⇒ k+n = m or

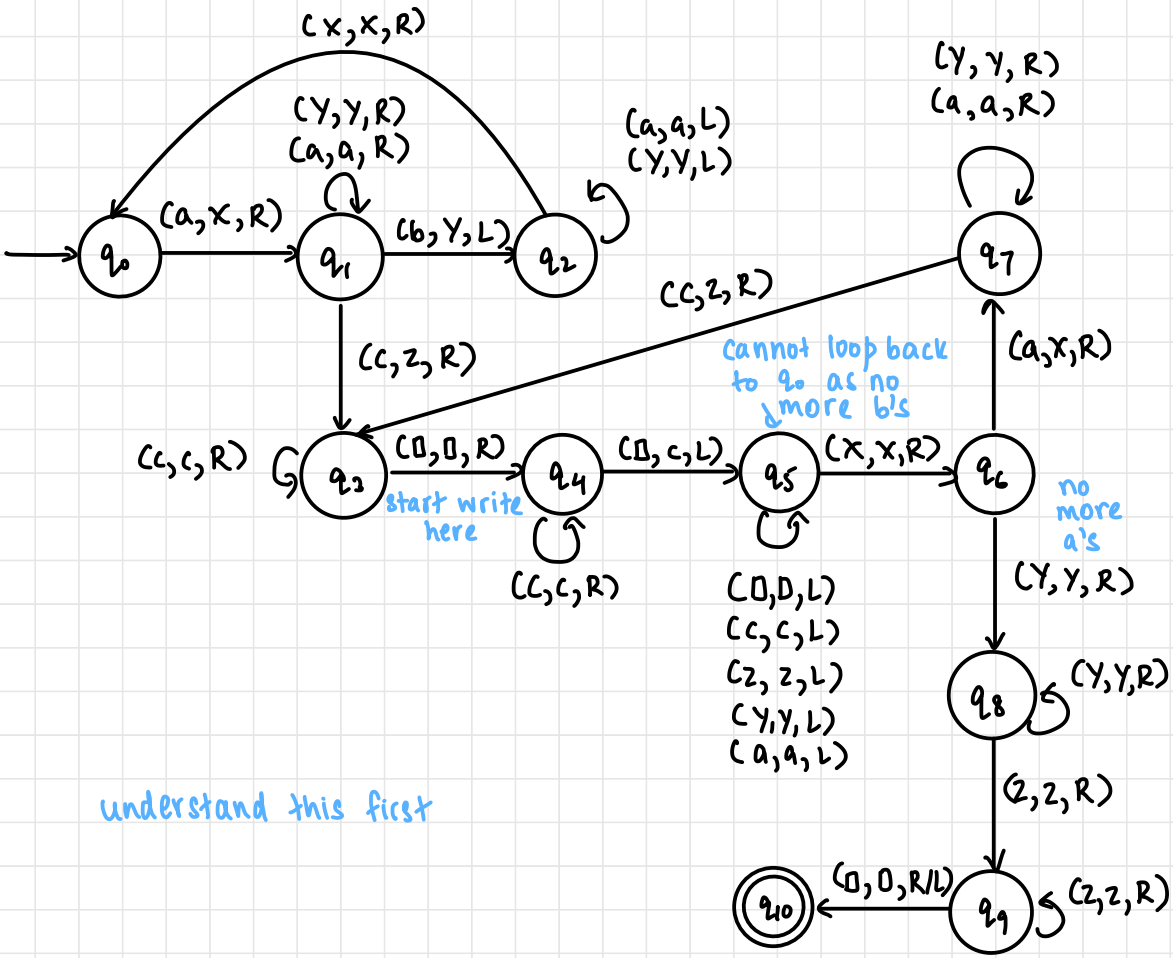
a = #b + #C

								o/p			
	a	a	a	a	b	b	c	c	▯	▯	▯
	x	a	a	a	y	b	c	c	▯	▯	▯
	x	x	a	a	y	y	c	c	▯	▯	▯
	x	x	x	a	y	y	z	c	▯	c	▯
	x	x	x	x	y	y	z	z	▯	c	c

match a's & b's

aaaa bb cc
x aaa Y
x x aa Y Y c
x x x a Y Y z c □ c
x x x x Y Y z z □ c c
↑ $q_6 \rightarrow q_8$

i) Only for #a's = #b's ; NOT COMPLETE YET

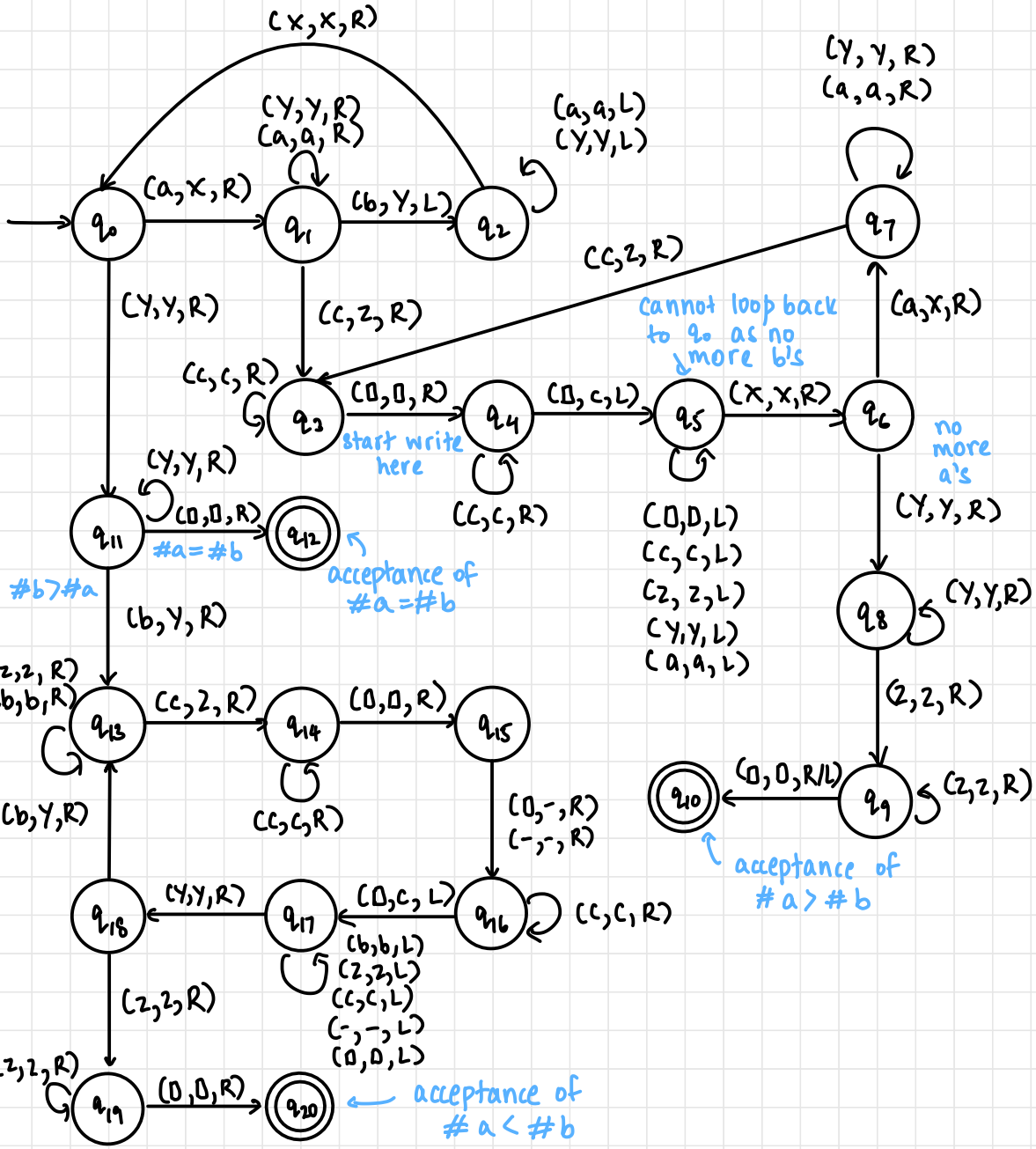


understand this first

2) # a's = # b's & # a's < # b's

aabbbbcc
XaYbbbcc
XXYYbbcc

→ XXYYYbZcB-C
XXYYYYZZB-CC
↑ q_{18} – q_{19}



Question 21

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

$n = 3$

$w = 000111111111$

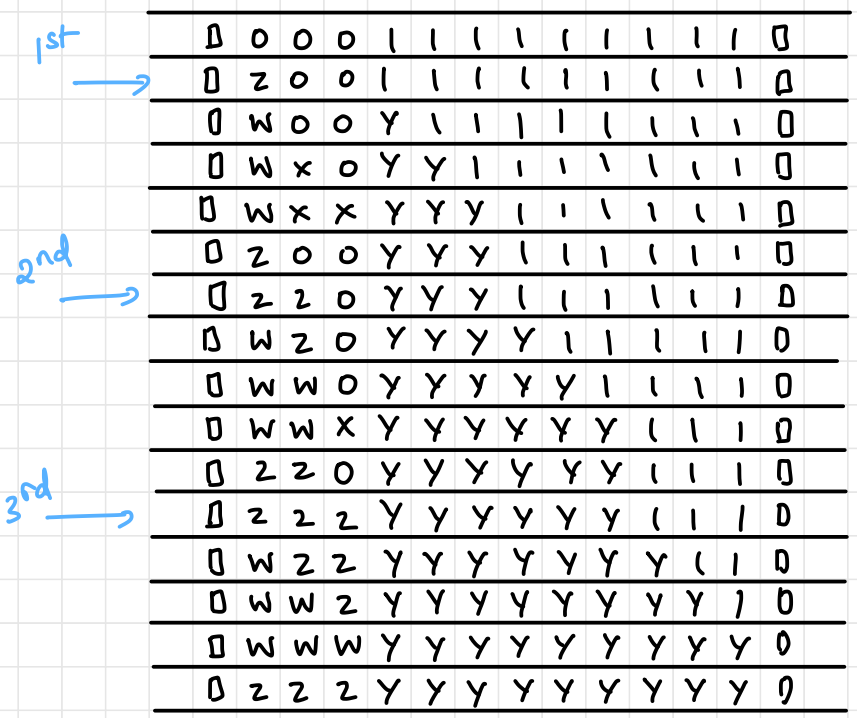
$x00yyy111111$

$xx0yyyyyy111$

$xxxyyyyyyyyy$

roughly

Tape contents



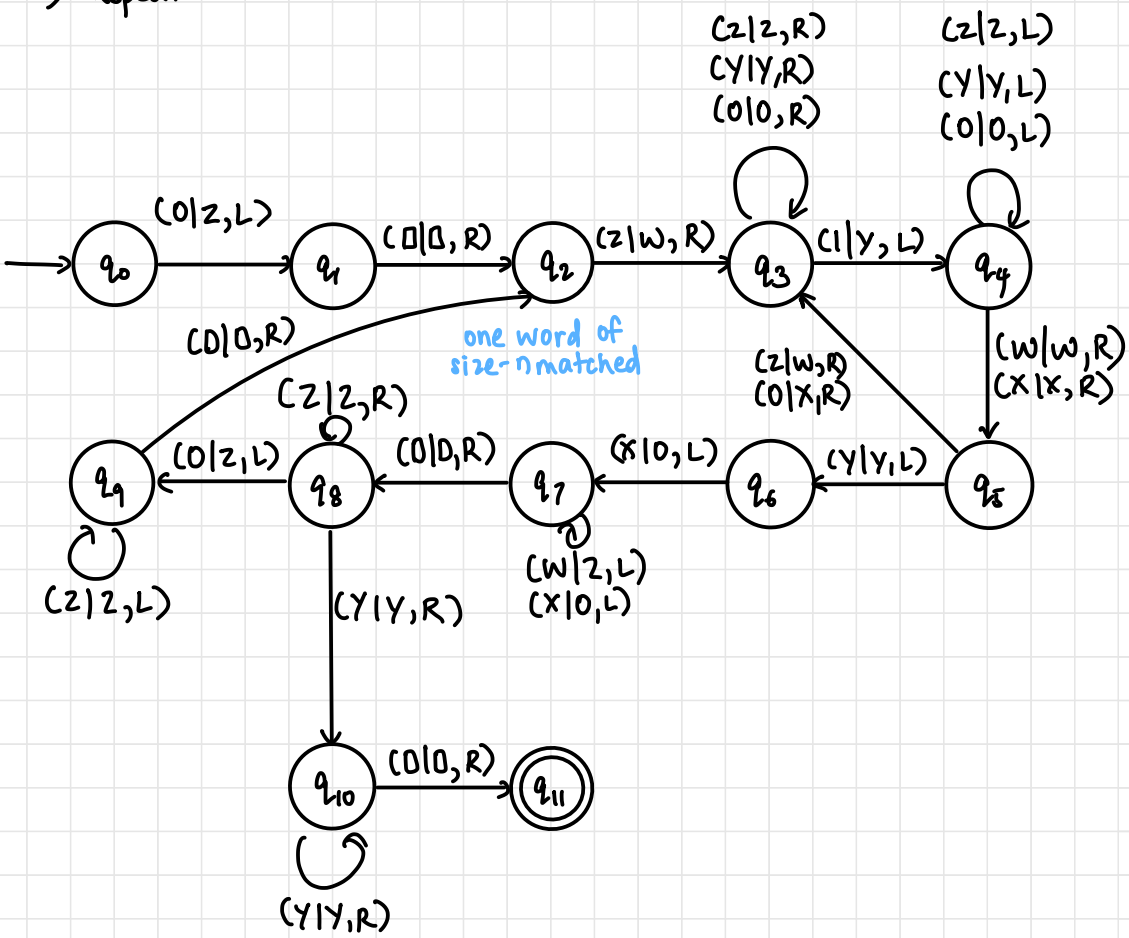
✓ accepted

Logic

1) Check if 0 exists

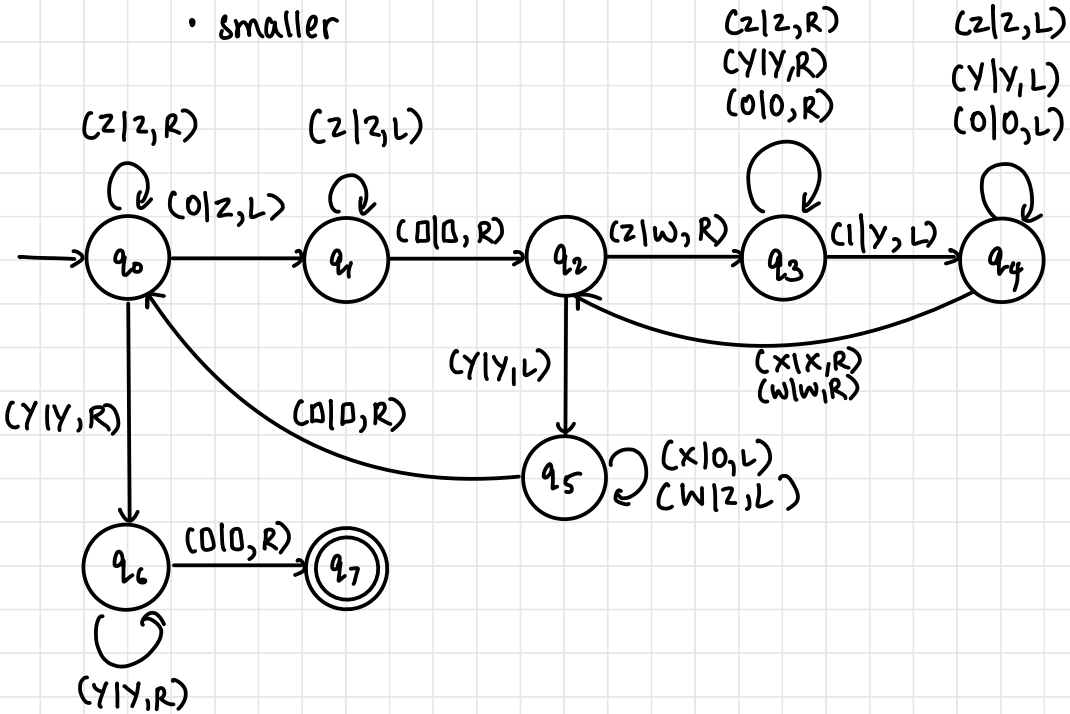
2) Check if n 1's corresponding to that 0 exist

3) Repeat



Alternate Machine

- smaller



Question 22

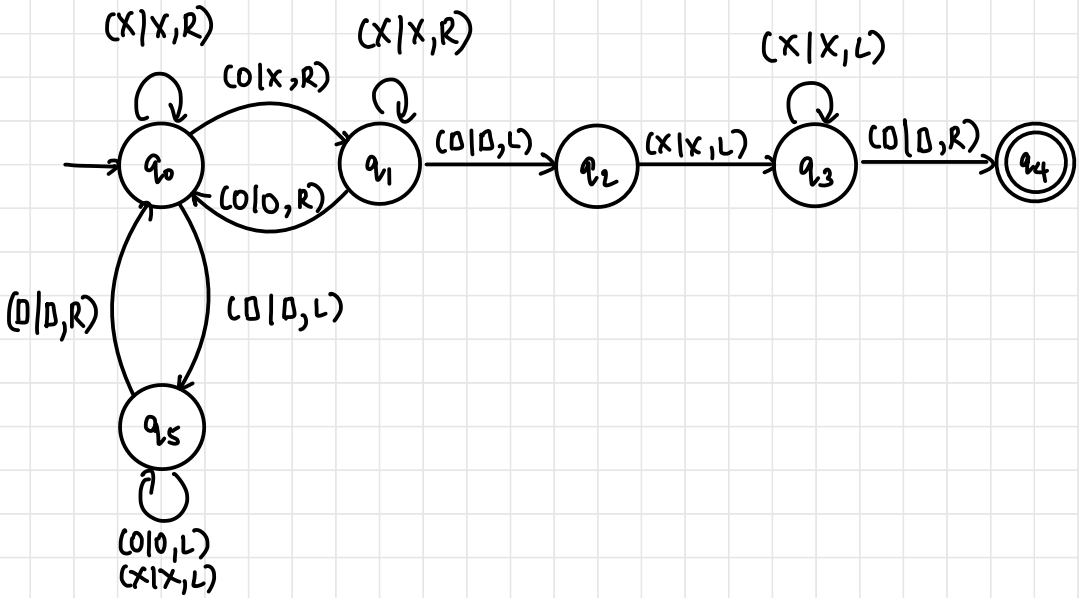
$$L = \{0^{2^n} \mid n \geq 0\}$$

$n = 3$

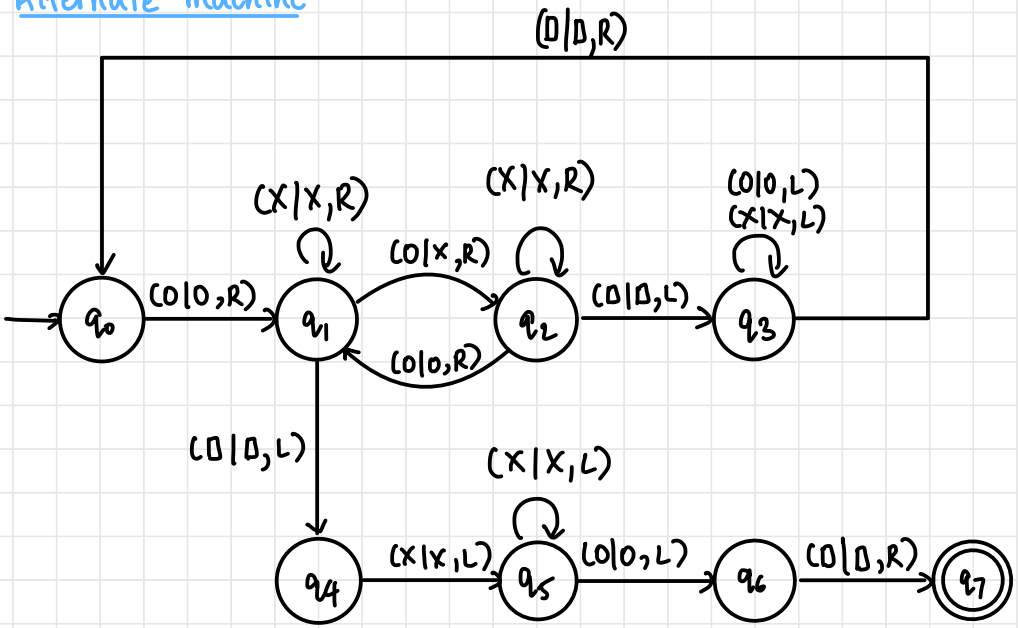
□	0	0	0	0	0	0	0	0	0
□	X	0	X	0	X	0	X	0	□
□	X	X	X	0	X	X	X	0	□
□	X	X	X	X	X	X	X	0	□
□	X	X	X	X	X	X	X	X	□

Logic

- every round, check for (1/2) of the remaining zeroes
- strike out alternate zeroes
- eg: for 8 0's, first round 4 will go, then 2, then 1 then 1



Alternate machine

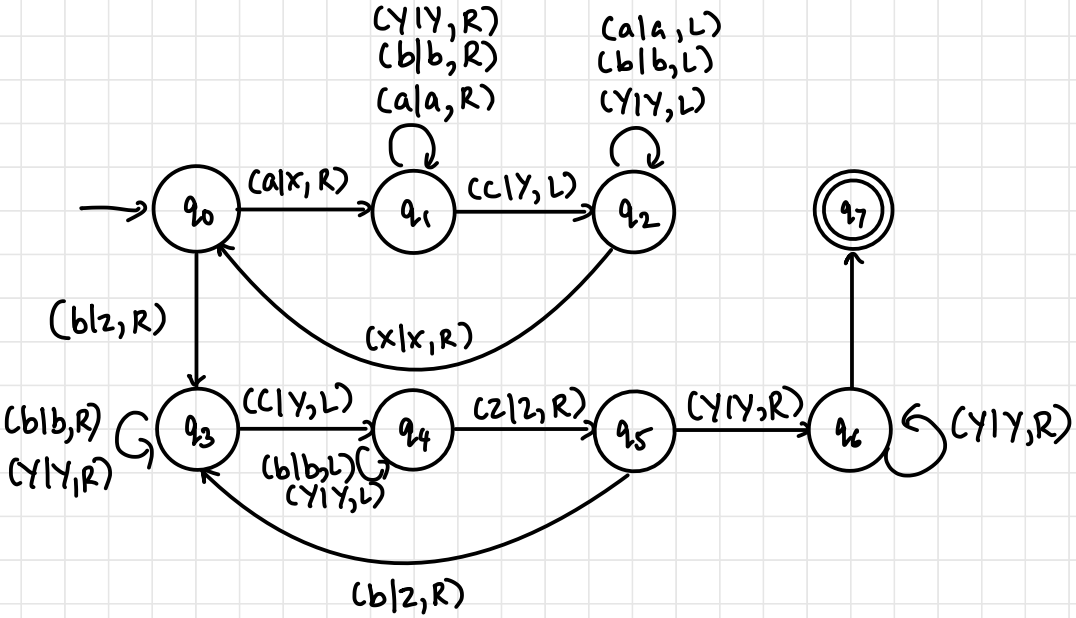


Question 23

$L = \{a^n b^m c^{n+m}\}$ accept

- match #a's with #c's
- match #b's with #c's
- accept

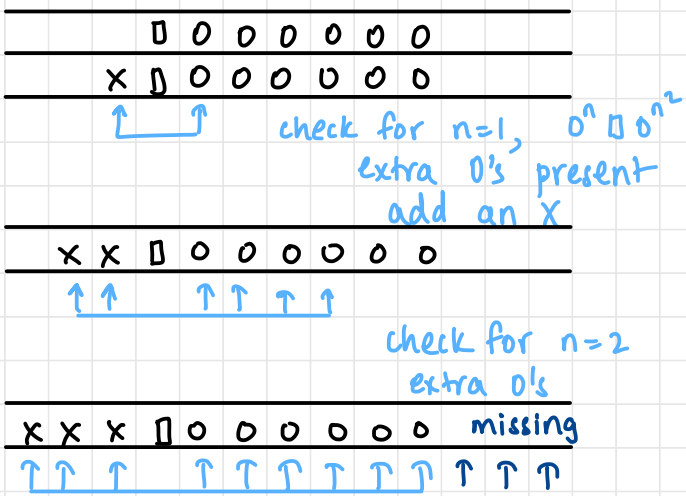
▯	a	a	b	b	b	c	c	c	c	c	▯	▯
▯	X	a	b	b	b	Y	c	c	c	c	▯	▯
▯	X	X	b	b	b	Y	Y	c	c	c	▯	▯
▯	X	X	Z	b	b	Y	Y	Y	c	c	▯	▯
▯	X	X	Z	Z	b	Y	Y	Y	Y	c	▯	▯
▯	X	X	Z	Z	Z	Y	Y	Y	Y	Y	▯	▯



Question 24

$$L = \{0^{n^2}\}$$

- in principle, hit and trial



- possible to construct

Church-Turing thesis (not a theory)

- anything that can be computed can be computed by a turing machine (standard TM)
- variations of the machine add no power (only time/space)
- multi-tape can be converted to single tape by separating the tapes with a symbol (#)

Computable Function

- Function that TM can compute
- Alonzo Church and Alan Turing

- Only functions that are computable by TM's are computable
- Algorithm: TM decider
- Set of all TMs

$\{ M_1, M_2, M_3, \ldots \}$ ← set of all Turing Machines

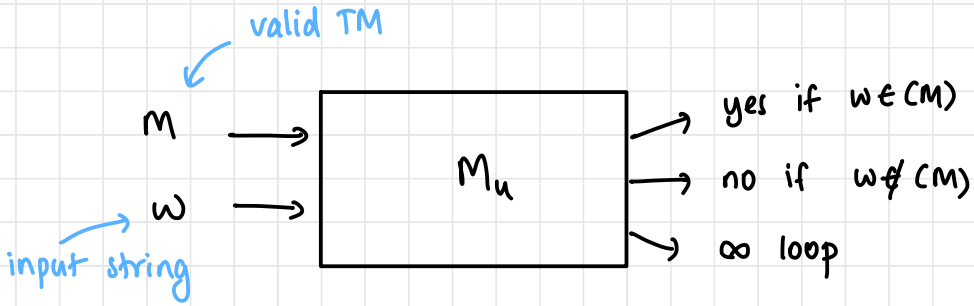
$\{ \mathcal{L}(M_1), \mathcal{L}(M_2), \mathcal{L}(M_3), \ldots \}$

↑ set of all possible computable functions

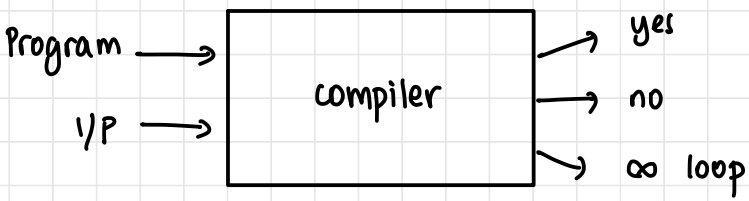
$\{ P_1, P_2, P_3 \ldots \}$ ← valid computer programs one could write

Universal Turing Machine

- Single TM that can compute any computable function
- Single TM that can simulate a TM (TM for a TM)
- Similar to compiler

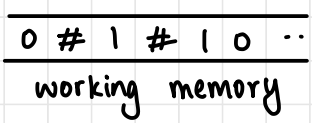


Similar to compiler

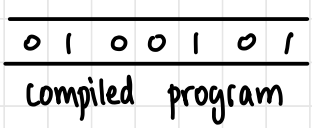


Structure of UTM

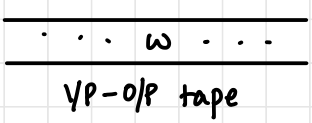
- 3-tape machine



current state no. in binary

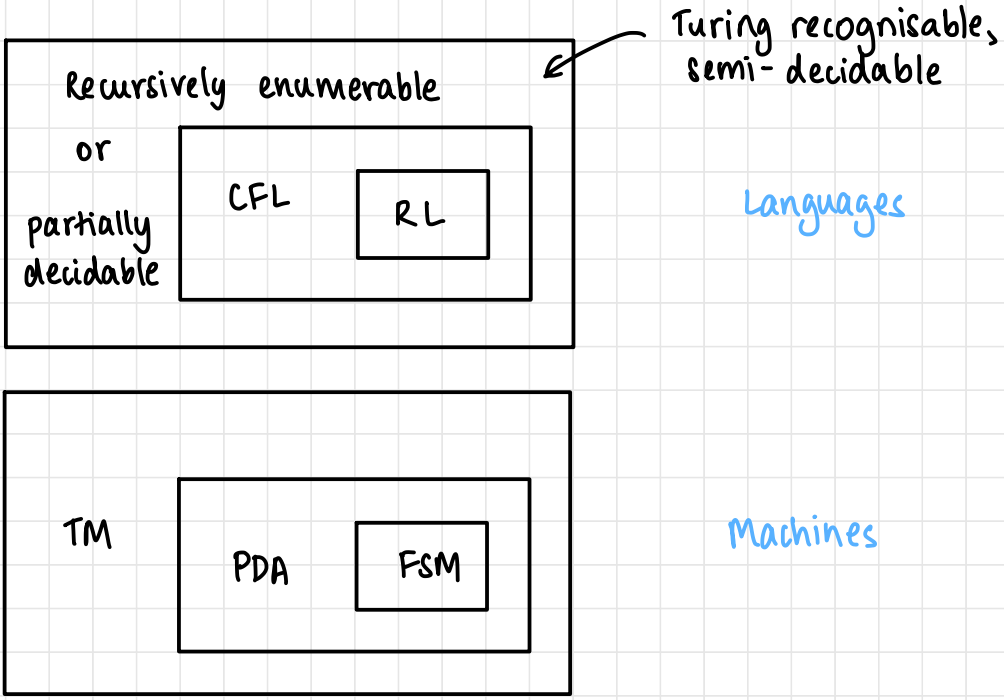


encoding of TM (binary string)
$\delta(q_2, b, L)$ left = 0 etc.



I/P loaded from, O/P written to

Chomsky Hierarchy



Recursively Enumerable Set

These languages are accepted by TM

a) a set is recursively enumerable if there exists an algorithm/ procedure that can enumerate all members of the set

b) makes sense to talk about the i^{th} element of a set

c) can generate an element of the set in finite time

d) if the set can be mapped with the set of natural numbers N in one-to-one correspondence

e) finite no. of elements between two elements in the set

Question 25

Prove that the following are RE languages (write TM to enumerate strings)

1) Set of N
2) Set of even no.s ($\bar{E}$)

] write each element onto tape with 1-1 correspondance to natural no.s

- countable infinity

3) Σ^*
   - set of all strings over the alphabet Σ
   - countably infinite
   - suppose $\Sigma = \{a, b, c\}$
   - generate strings of increasing length

works ↓

λ length 0 ← can print string in finite time

a, b, c length 1

aa, ab, ac, ba, bb, bc, ca, cb, cc length 2

⋮

   - alphabetical does not work

     $\lambda, a, aa, aaa \ldots$ never reaches b

4) Set of all TMs are enumerable
   - every TM can be described as binary string
   - length $|\langle M \rangle|$ is finite — finite binary strings (finite states)
   - set of TM $\subset (0+1)^*$ (Σ^*)

- proper subset (not all encodings are valid)
- enumerate like prev. example
- makes sense to talk about i^{th} TM

5) $Q = \left\{ \frac{m}{n} \mid m,n \in N \right\}$ is a set of +ve rational no.s

Prove countably ∞

	1	2	3	4	5	
1	$\frac{1}{1}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	. . .
2	$\frac{2}{1}$	$\frac{2}{2}$	$\frac{2}{3}$	$\frac{2}{4}$	$\frac{2}{5}$	. . .
3	$\frac{3}{1}$	$\frac{3}{2}$	$\frac{3}{3}$	$\frac{3}{4}$	$\frac{3}{5}$	. . .
4	$\frac{4}{1}$	$\frac{4}{2}$	$\frac{4}{3}$	$\frac{4}{4}$	$\frac{4}{5}$	. . .
5	$\frac{5}{1}$	$\frac{5}{2}$	$\frac{5}{3}$	$\frac{5}{4}$	$\frac{5}{5}$	. . .
	.	.	.	.	.	
	.	.	.	.	.	
	.	.	.	.	.	

check for duplicates in prog.

- count diagonally; not row/column wise

6) Enumerate all strings in RE language

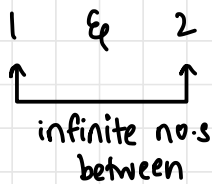
- parallel computation, one step at a time
- no full string processed at once

Non-Recursively Enumerable

- Cantor – pair elements of different sets
- Diagonalisation

Question 26

Set of all real no.s



if precision limited, becomes recursively enumerable

- cannot enumerate all real no.s
- uncountably infinite
- cannot talk about ith element (real no.)

Formal Proof

Diagonalisation – proof by contradiction

Assumption: R is enumerable

	f(n) ⇒ func. that lists all real no.s
1	3 . (1) 4 2 3 4
2	7 . 8 (9) 1 4 0
3	8 . 2 3 (6) 7 1
4	9 . 1 1 7 (9) 2
5	2 . 2 3 0 1 (3)

$r_d = .19693$

complement / add / sub / alter

$\overline{r_d} = .08582$

- $\overline{r_d}$ cannot be #1 ∵ it differs in the first dec.
- cannot be #2 ∵ it differs in the second dec.
- differs from i^{th} row in i^{th} place

- $\overline{r_d}$ is new number that has not been enumerated (contradiction)

- ∴ it is not recursively enumerable

Question 27

Set of all languages

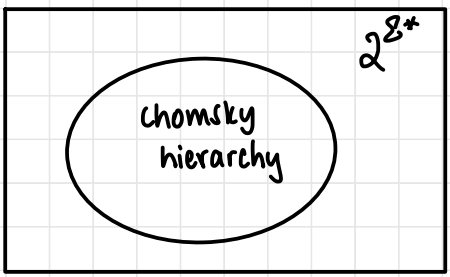
any language $\subseteq \Sigma^*$ ← enumerable

$\{L_1, L_2 \ldots\}$ ↑ all strings

Total no. of subsets that can be formed using the set Σ^*

= Power set of Σ^*

$\mathcal{L} \in P(\Sigma^*)$ 2^{Σ^*} languages



Formal Proof

Assumption: Set of all languages is enumerable

TMs		s_1	s_2	s_3	s_4	s_5	s_6	...	Call strings - enum)
M_1	L_1	1	0	0	1	1	0		$\mathcal{L}_1 = \{s_1, s_4, s_5 \ldots$
M_2	L_2	0	1	0	0	0	1		$\mathcal{L}_2 = \{s_2, s_6 \ldots$
M_3	L_3	1	0	1	1	0	1		$\mathcal{L}_3 = \{s_1, s_3, s_4, s_6 \ldots$
M_4	L_4	1	1	0	0	0	1		$\mathcal{L}_4 = \{s_1, s_2, s_6 \ldots$
M_5	L_5	0	1	1	0	0	1		$\mathcal{L}_5 = \{s_2, s_3, s_6 \ldots$
⋮	⋮	0	1	1	0	1	1		$\mathcal{L}_6 = \{s_2, s_3, s_5, s_6 \ldots$

(1 - string is in language)

$\mathcal{L}_{diag} = \{s_1, s_2, s_3, s_6\}$ ↓ could be enumerated

find $\mathcal{L}$'s complement

$\mathcal{L}_{NON\text{-}RE} = \{s_4, s_5 \ldots\}$ ↓ not same as any enumerated string

- differs from i^{th} row in i^{th} place
- ∴ there are more languages than Turing Machines
- TMs are RE and finite
- There are more problems than solutions (TMs)
- Most $\mathcal{L}$ are non recursively enumerable (do not have direct description)

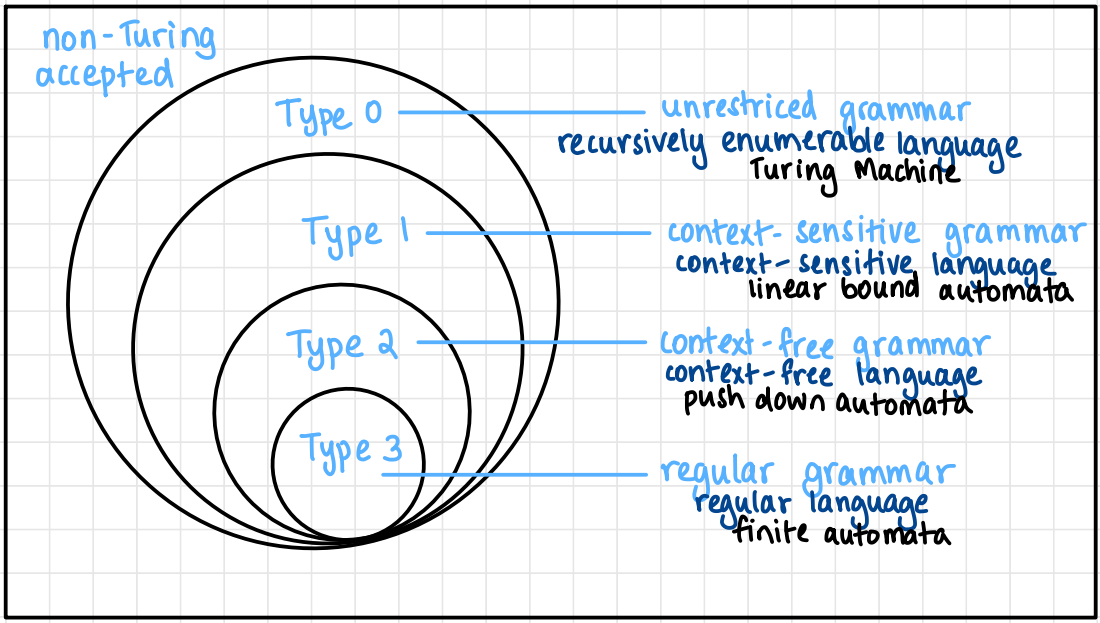
- There exist RE languages that are not decidable as well as non RE languages that are full undecidable

Recursive	Recursively Enumerable
• recursive ⊂ RE	
• enumeration procedure (Turing Machine)	• enumeration procedure (Turing Machine)
• Membership function	• No membership function (can enter ∞ loop)
• TM deciders	• TM acceptors

undecidable → non-RE (R, set of all langs)

undecidable → RE but not decidable (PCP, empty intersection, halting problem, acceptance problem)

Chomsky Hierarchy



Undecidability

- Exploring limits of algorithmic solvability; unsolvability
- Non-RE → no TM
- RE but not decidable → no algorithm for all instances
- Try to simplify / alter problem if unsolvable

POST CORRESPONDANCE PROBLEM

- Undecidable decision problem ⇒ RE but not decidable
- Emil Post in 1946
- Given two sets of strings A & B framed over the same alphabet $\Sigma = \{a, b\}$

	A	B
1	a_1	b_1
2	a_2	b_2
3	a_3	b_3
4	a_4	b_4
⋮	⋮	⋮
n	a_n	b_n

could be infinite; same no. of strings in A & B

- Come up with a sequence such that if strings of A & B are concatenated in that sequence, the resulting strings are equal
- eg: sequence 3, 2, 1

$$a_3 a_2 a_1 = b_3 b_2 b_1$$

should hold true

- Programming approach: if list is finite and repetitions are not allowed, all permutations can be listed and it is solvable
- If infinite or repetitions allowed, unsolvable

Question 28

PCP for the given sets of strings

	A	B
1	10	10 110
2	111	01
3	110110	1010

- cannot start with 2 as it differs in first place
- cannot start with 3 as it differs in second place
- can start with 1

	1	3		
A	10	110	110	] X
B	10	110	1010	

- this set of A&B has no solution

Question 29

	A	B
1	a	baa
2	ab	aa
3	bba	bb

PCP : solve

- cannot start with 1 or 2

	3	2	3	1	
A	bba	ab	bba	a	] bbaabbbaa ← solvable
B	bb	aa	bb	baa	

Question 30

Solve PCP

	A	B
1	bb	bbb
2	baa	aab
3	bbb	bb

1)

	1	3
A	bb	bbb
B	bbb	bb

$(13+31)^*$ any combination

2)

	1	2	3
A	bb	baa	bbb
B	bbb	aab	bb

Question 31

Solve PCP

	A	B
1	10	101
2	011	11
3	101	011

	1	3	1	
A	10	101	10	×
B	101	011	101	

	1	3	3	
A	10	101	101	... never halts
B	101	011	011	...

- If able to prove A: undecidable (PCP), can prove B is undecidable

Idea of Reduction

subroutine

$A \leq_m B$ assume if B can be solved, so can A

(reduce A to B)

know undecidable (PCP)

Question 32

∞ loop

(Problem of empty intersection) Given 2 CFLs $\mathcal{L}_1$ & $\mathcal{L}_2$, can we find out if $\mathcal{L}_1 \cap \mathcal{L}_2$ is ϕ

- Reduce PCP to $\mathcal{L}_1 \cap \mathcal{L}_2 = \phi$; change I/P

$$PCP \leq_m \mathcal{L}_1 \cap \mathcal{L}_2 = \phi ?$$

PCP

	A	B
a	1	111
b	10111	10
c	10	0

convert

2 grammars

List A: $S_A \rightarrow 1S_A a \mid 10111 S_A b \mid 10 S_A c \mid -$

List B: $S_B \rightarrow 111 S_B a \mid 10 S_B b \mid 0 S_B c \mid -$

	b	a	a	c
A	10111	1	1	0
B	10	111	1	0

- try to generate w= 10 111 1110 with grammars

1) $S_A \rightarrow 1S_A a \mid 10111S_A b \mid 10S_A c \mid -$ (baac)

$$S_A \overset{lm}{\Rightarrow} 10111\, S_A\, b$$
$$\Rightarrow 10111\, 1S_A ab$$
$$\Rightarrow 10111\,1\,1S_A aab$$
$$\Rightarrow 10111\,11\,0S_A caab$$
$$\Rightarrow 10111\,11\,10 - caab$$

sequence kept in check

reversed because it is nested

2) $S_B \rightarrow 111S_B a \mid 10S_B b \mid 0S_B c \mid -$

$$S_B \overset{lm}{\Rightarrow} 10\, S_B\, b$$
$$\Rightarrow 10\,111\, S_B\, ab$$
$$\Rightarrow 10\,111\,111\, S_B\, aab$$
$$\Rightarrow 10\,111\,1110\, S_B\, caab$$
$$\Rightarrow 10\,111\,1110 - caab$$

undecidable

- However, PCP is unsolvable ⇒ solution to empty intersection is unsolvable (no algorithm)
- One particular PCP is solvable but not all

Question 33

$A \leq_m B$
$B \leq_m C$
$D \leq_m C$

A is RE but not decidable, C is recursive.

Can this statement be true?

No, it is false ∵ A is UD ⇒ B is UD ⇒ C is UD

Acceptance Problem of TMs & Halting Problem of TMs

- A_{TM} and $HALT_{TM}$
- RE languages but not decidable ⇒ undecidable languages
- $A_{TM} = \{ \langle M, w \rangle \mid$ if M is a TM and M accepts $w \}$
  ↑ pair
- $HALT_{TM} = \{ \langle M, w \rangle \mid$ if M is a TM and M halts on $w \}$
- Diagonalisation → reduced $A_{TM} \leq_m HALT_{TM}$ ← subroutine
  ↑ undecidable

Proving that $L = A_{TM}$ is Undecidable

$$A_{TM} = \{ \langle M, w \rangle \mid \text{if M is a TM and M accepts } w \}$$

- Assume there exists a TM 'H' that can decide any A_{TM} (takes $\langle M, w \rangle$)
- Define new TM 'D' that uses 'H' as a subroutine
- I/P: M, $\langle M \rangle$ ⇒ O/P: accept if M rejects description
  reject if M accepts description

Machine H

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
M_1	A	R	A	R		A
M_2	A	A	A	A	...	R
M_3	R	R	R	R		A
M_4	A	A	R	R		R
⋮			⋮			
D	R	R	A	A		?

can be anything

? must be its contradiction

- ∴ D & H cannot exist
- A_{TM} is undecidable

$HALT_{TM}$

- $HALT_{TM} = \{ \langle M, w \rangle \mid$ if M is a TM and M halts on w $\}$
- Halt by accepting or rejecting
- $A_{TM} \leq_m HALT_{TM}$
- Assume there exists TM 'R' that decides the language $HALT_{TM}$ (subroutine)
- TM that decides A_{TM} = 'H'
- If R decides halting, then definitely H decides A_{TM}; but H cannot exist