COMP003 Revision Notes - Automata.

Set. $A = \{x, y, z\}$ unordered collection.

Tuple: $A = (x, y)$ ordered collection.

Function: $f: D \to R$ map from domain (set) to codomain (range)

Cartesian Product: $A \times B \to \{(a,b) \mid a \in A \wedge b \in B\}$

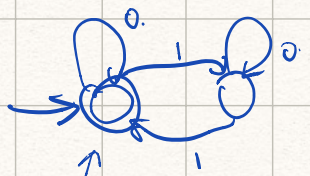
Power set $A = \{x, y, z\}$.

$P(A) = \{\phi, \{x\}, \{y\}, \{z\}, \{x,y\}, \{x,z\}, \{y,z\}, \{x,y,z\}\}$

Graph: nodes & edges (could be directed or undirected)

DFA. deterministic finite automata.

finite state machines



$\Rightarrow$ accepts even number of 1s.

acceptance state.

tracks one piece of information.

Definition DFA $M \to (Q, \Sigma, \delta, q_0, F)$.

- Q: finite set of states
- Σ: finite set of input chars
- δ: transition function $(Q \times \Sigma \to Q)$ (state, input char, next state)
- q_0: start state (1)
- F: accept states set

Example.

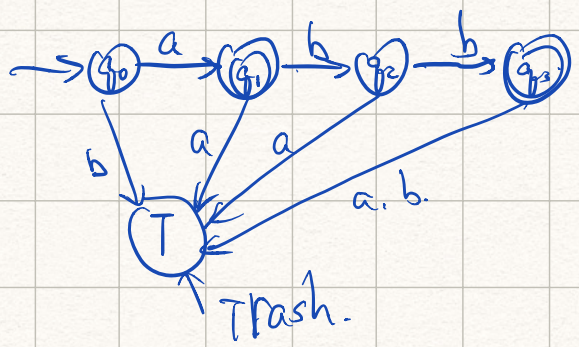
$Q = \{q_0, q_1, q_2, q_3\}$.

$E = \{a, b\}$.

$\delta =$

	a	b
q_0	q_1	
q_1		q_2.
q_2		q_3.
q_3		

⇐ ideally want this be fully defined.



$q_0 = q_0$

$F = \{q_1, q_3\}$.

A DFA accepts a string $w = w_1 w_2 \cdots w_n$

if ∃ seq of states that 1. $r_0 = q_0$ ⇐ in start state

2. $\delta(r_i, w_{i+1}) = r_{i+1}$.

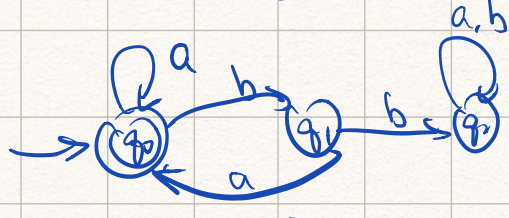
for $i = 0 \cdots n-1$.

Good transitions 3. $r_n \in F$. ↑ ends in valid state.

Language

a set of string that DFA accepts

Exeple



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

$E = \{a, b\}$.

$\delta =$ — — —

$q_0 = q_0$

$F = \{q_0\}$.

$L = \{\phi, a, aa, baaba, aaa, baa, baaa, abaa, aaba, aaaa, babe\}$.

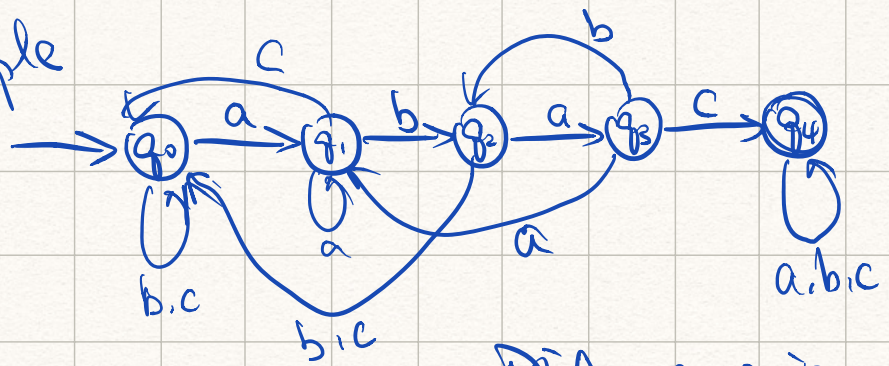
$\{s.t.$ all b must follow by a $a\}$.

Regular Language

a set of Language DFA recognises.

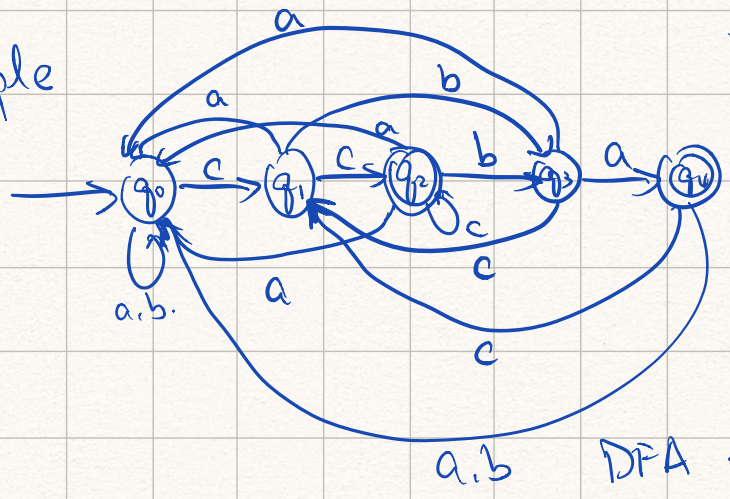
$L \in RL$ iff $\exists$ a DFA $M = \{Q, \Sigma, \delta, q_0, F\}$ that accepts $\forall s \in L$, rejects $\forall s \notin L$

Example



DFA recognise abac.

Example



DFA that recognises strings end in cba or cc

RL Closure.

1. given RL L_1, L_2, $L_1 \cup L_2 = L_3 \Rightarrow RL$
2. $L_3 = \{s_1 s_2 \mid s_1 \in L_1, s_2 \in L_2\}$. Concat.
3. Σ^* / L is also a RL
4. L^* is also a RL

Complement Closure.

DFA M ⇒ make DFA N rejects all M accepts

$F_N = Q / F_M$

Union close.

Given DFA M_1, M_2 → DFA N that recognises all strings that M_1, M_2 does

$Q = Q_1 \times Q_2$. $\quad \delta((q_1, q_2), y) = (\delta(q_1, y), \delta(q_2, y))$

$\Sigma = \Sigma_1 \cup \Sigma_2$.

$q_0 = q_1, q_2$

$F = \{(q_1, q_2) \mid q_1 \in F \cdot \vee \; q_2 \in F\}$

NFA Syntax sugar (is a DFA)

- ε transition.
- Multiple δ with same input. (multiple transitions)
- δ with multiple inputs

Non-deterministic

- multiple actions possible.
- go through simultaneously
- know the right path if accepted

Q, Σ, q_0, F remain same.

$\delta : Q \times \Sigma_\varepsilon \to Q$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

Regular Expressions

1. a for $a \in \Sigma$
2. ε
3. $\emptyset$

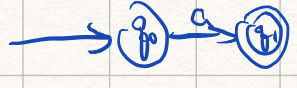
---

4. $(R_1 \cup R_2)$
5. $(R_1 \circ R_2)$
6. R_1^*

Regex same power DFA/NFA.

1. Regex → DFA/NFA

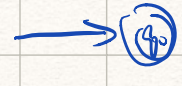
Base → Induction.

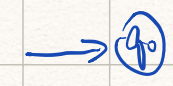
a: 

$R_1 \cup R_2$.

$R_1 \circ R_2$

R_1^*.

ε: 

∅: 

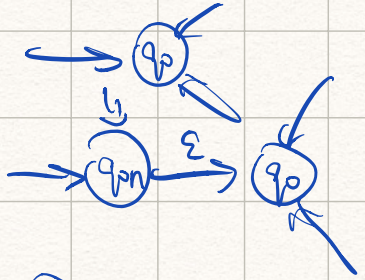
DFA/NFA → Regex.

need GNFA.

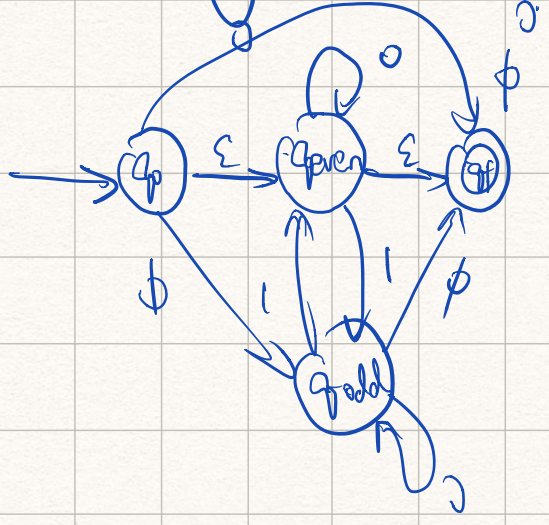
Generalized nondeterministic Automata.

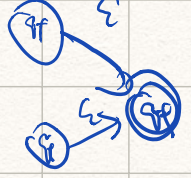
- Like NFA / with regex on transitions
- No incoming arrows on start
- ONE accept state no outgoing arrow.

DFA → GNFA



Example





- All missing arrows have ϕ

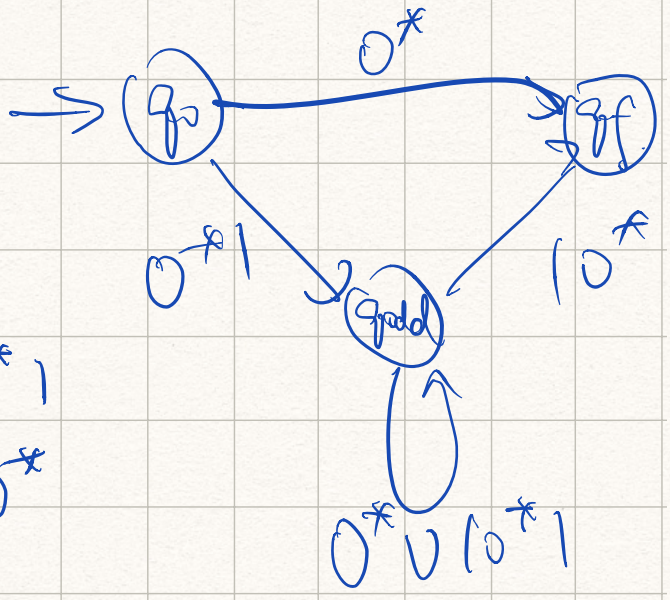
Kill q_{even}

$q_0 \to q_f \quad \phi \cup 0^* = 0^*$

$q_{odd} \to q_{odd} \quad 0^* \cup 10^*1$

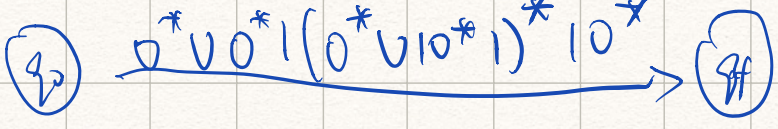
$q_0 \to q_{odd} \quad \phi \cup \varepsilon 0^* 1 = 0^*1$

$q_{odd} \to q_f \quad \phi \cup 10^* \varepsilon = 10^*$



Kill q_{odd}.

$q_0 \to q_f \quad 0^* \cup 0^*1(0^* \cup 10^*1)^* 10^*$



DFA / NFA / Regex Limitations

$L = \{w \mid w$ contains as many 0's as 1's$\}$

Pumping lemma.

$\forall$ RL L_1 $\exists$ $n \in \mathbb{Z}^+$ $\forall s \in L$, $|s| \geq n$,
$\exists u, v, w \in \Sigma^*$ $s = uvw$.

$|uv| \leq n$.

$|v| \geq 1$.

$uv^i w \in L$. for all $i \geq 0$

RL A $\rightarrow$ DFA M. $\rightarrow$ n states

n is puping length.

Any string with n chars starts in one state

(Go to n+1 states.).

at least one state visited twice.

can be infintely traversed

to yield puping

$\Downarrow$

(longer than n)

In RL, any long string can generate

infinitely many / long strings also in RL.

$|v| \geq 1$. DFA consues char during transitions

$|uv| \leq n$. In worst case, the $\Leftrightarrow$ occurs on last state.

$|u|$ should have $n-1$

$|v| = 1.$

$|uv| = n.$

All RL follow pumping lemma.

Example

Prove $A = \{0^n 1^n \mid n \geq 0\}$ is not RL.

Assume pumping length p

$s = 0^p 1^p$

① $s = xyz.$ y only contain 0s. 0s > 1s.

② $s = xyz$ y only contain 1s 1s > 0s

③ $s = xyz$ pump it will become 00110011 ---. not in A.

Example

$A = \{s \mid s$ contains same number of 0s and 1s$\}$

$|xy| \leq p$ $s = xyz$

Assume $s = 0^p 1^p$

y could be a string

all 0.

pumping y would cause s' not in A.

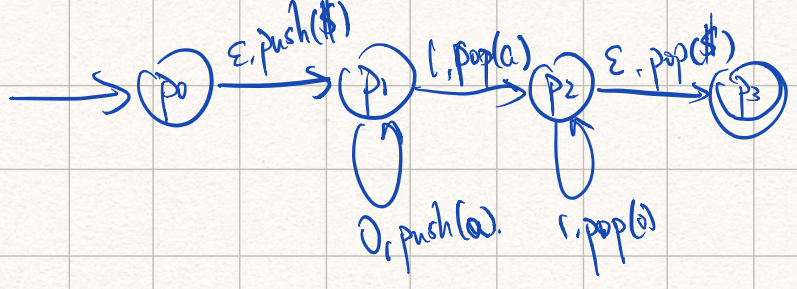
Proof Language not regular (breach of pumping lemma)

New feature Automata.

Pushdown Automata

Like a NFA with a stack. (infinite storage)

Example PDA recognizes $0^n 1^n$.



no valid transition = reject

next char && top of stack (pop) should both match to make a transition.

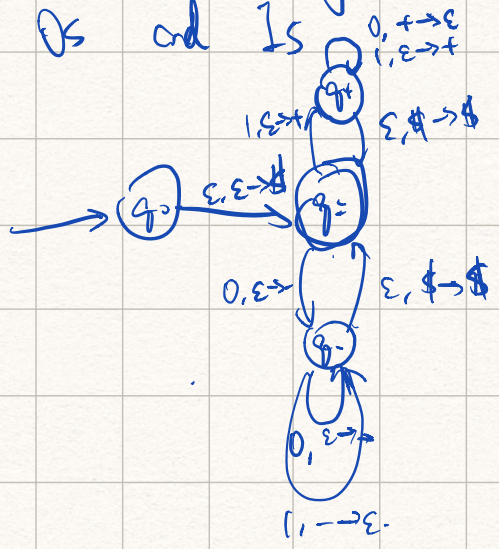
PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$

Γ: set of stack alphabets

$\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to P(Q \times \Gamma_\varepsilon)$

↑ to pop ↑ to push.

Example. PDA that recognises strings with equal number of 0s and 1s



DPDA NPDA recognise CFL.

Language generated by CFG.

Context Free Grammer.

(V, Σ, R, S)

V → finite set of variables (non-terminals)

Σ → terminals

R → rules $A \to w$, $A \in V$, w is a string of terminal and nonterminals

$S \in V$ Start variable

$S \to 0S1 \mid 01S \mid S01 \mid \varepsilon$
$1S0 \mid 10S \mid S10$

Recognise CFL with NPDA/PDA.

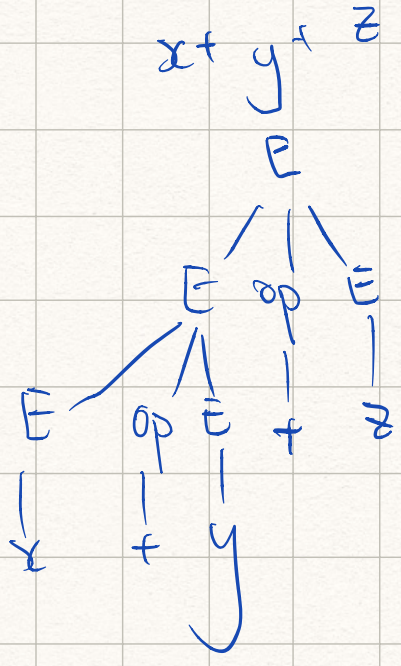
Generate CFL with CFG

Parse Tree.

children of a node → RHS.

leaves = terminals.

interior nodes → variables



Pumping lemma for CFLs

$s \geq$ length p.

$s = uv^i xy^i z \in A$ for $i \geq 0$

second / fourth should be pumped simultaneously.

$|vy| > 0$ non-trivial.

$|vxy| \leq p$ useful for proofs

Let b = maximum number of children a node can have in parse tree / terminal / non-terminal on right hand of a rule.

$n = |V|$ number of non-terminals

$p = b^{n+1}$.

if a tree have b children per node.

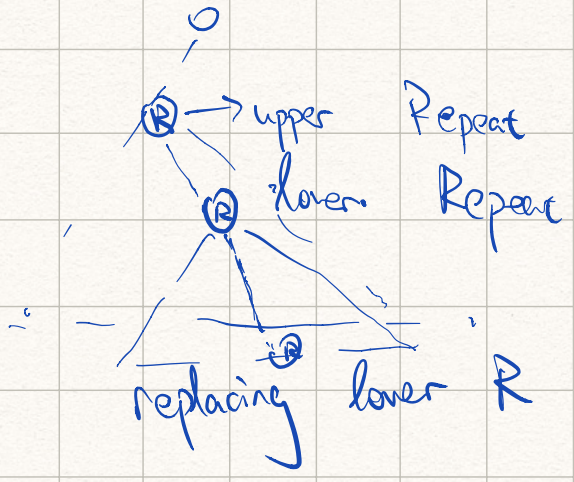
0	level	1
1	level	b.
2	—	b^2
⋮		⋮

if have b^{n+1} terminals only b^n n levels terminals

we need n+1 lvls

if n+1 lvls, from root to terminal we have n+1 non-terminals and a terminal.

a variable/non-terminal is repeated.



replacing lower R with upper R's tree

Example $S = a^p b^p c^p$ not a CFG.

$\forall x\ y\ z$:

$|vxy| \leq p$

vxy cannot contain all 3 chars

if pump v, y, would result in chars not having same occurences

Example

$E \to E\ Op\ E \mid (E) \mid Var$

$Op \to + \mid - \mid * \mid \div$

Var → x | y | z

E → E Op E | (E)

b = 3

n = 3 (E, Op, Var)

$p = b^{n+1} = 81$

Given a language PDA / CFG ?!

given p → construct "evil string" that follows the language and breaks pumping lemma.

Exercise: Prove $L = \{a^i b^j c^k \mid 0 \le i \le j \le k\}$ is not CFG

Given p construct $s = a^p b^p c^p$

should be able to

$uv^i xy^i z$

$|vxy| \le p$ $|vy| > 0$

vxy could not have 3 chars

Exercise

$L = \{ww \mid w \in \{0,1\}^*\}$ is not CFL.

$s = 0^p 1^p 0^p 1^p$

$uvxyz$

$|vxy| \le p$ ① vxy in first half / second half

② vxy shifts the

midway point leftmost right next to let it start with 1

⑤ vxy partly in each half.

$|vxy| \leq p$ part in first half made of 1.

part in second half made of 0s.

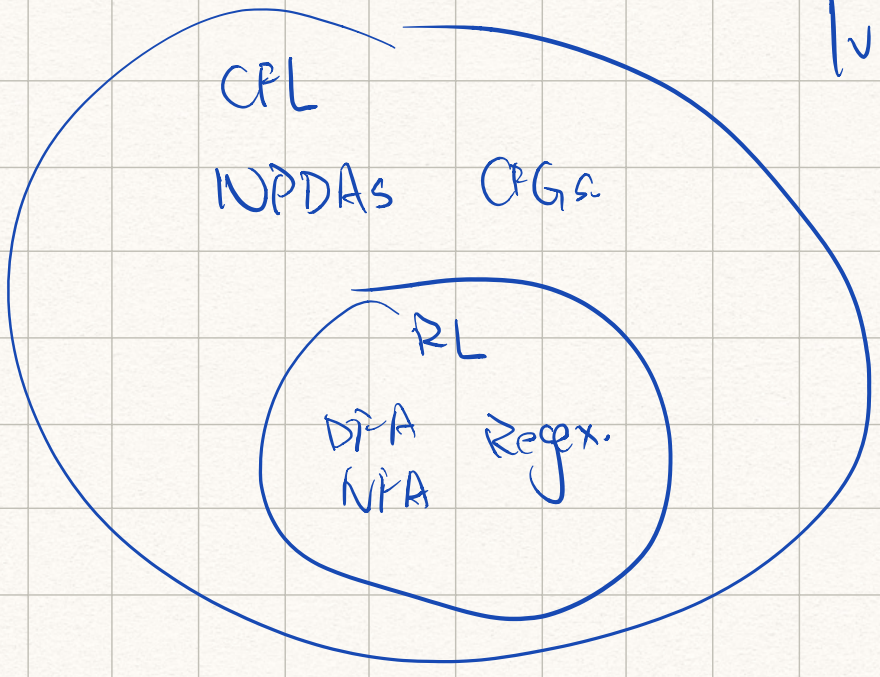
delete them / pump down

cause first half

$\underbrace{0\cdots0}_{p}\ \underbrace{1\cdots1}_{\text{less } p}\ \underbrace{0\cdots0}_{\text{less } p}\ \underbrace{1\cdots1}_{p}$

not L.

$\therefore L \notin CFL$



DFA/NFA/Regex

Limited memory (state).

PDA ↓ Stack / CFG.

infinite memory stack / one access top of stack

What if ?

free access, infinite memory

Turing machine.

NFA with a inf tape and read head.

Every transition.

read / replace. / move read head L/R.

Turing machine have one single accept / reject state.

if in accept / reject immediately

s.t. possible to get stuck in the middle.

Def. TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

Q, Σ, q_0. remain.

Γ (tape alphabet) $= \Sigma \cup \{\sqcup\}$. space.

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

read in / replace / go L/R.

A configuration of TM

(1) current state in TM.

(2) contents of tape

(3) location of read ahead

$C = w_1 q_i w_2$. current state in automata.

string left / current + string right

$C_1 = q_0 w$.

Turing machine accepts a string if:

$C_1 = q_0 w$.

$C_i \Rightarrow C_{i+1}$ for $i = 1 \dots k-1$

if $\delta(q_i, b) = (q_j, c, L)$.

$uaq_ibv \Rightarrow uq_jacv$.

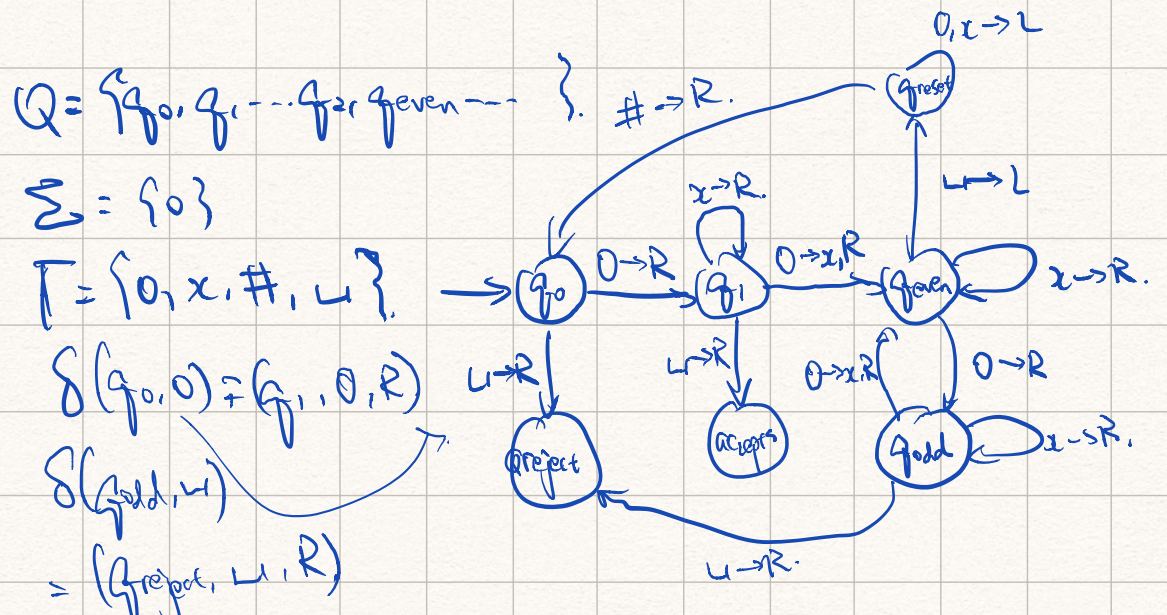
if $\delta(q_i, b) = (q_j, c, R)$

$uaq_ibv \Rightarrow uacq_jv$.

C_k is an accept. config (contains q_{accept})

Assume that all tape begins with #

Exercise Build a TM accepts 0^{2^n}.



reject.

Building TM. → Replace symbols with cross off versions
finding edges of input by looping
and changing states
use cycles to repeat.
go to reject / accept when
conditions met

middle of string.

mark a left → mark a right.

when mark right ← marked left.

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

marker
$w \mid w$

delete one by one.

ACCEPT if all chars are deleted.

Multitape TM

Input one first tape.

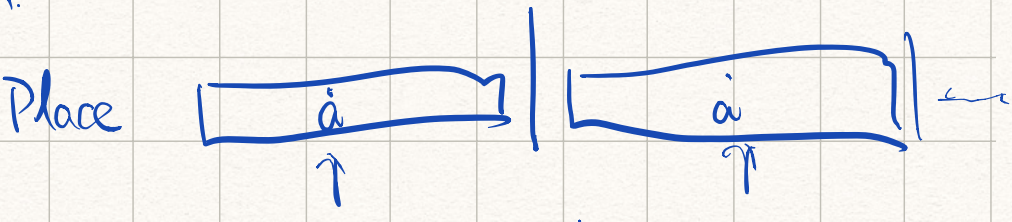
k tapes k heads

Can read k chars

Can change state/ replace chars...

Multitape TM = TM.

vanilla TM.



current read heads.

update k times for each transition on multitape TM.

nondeterministic TM.

$\delta(q_i, a) \rightarrow$ set of $(q_j, b, L/R)$

3 tape multitape TM

Tape 1 original input.

Tape 2 branches made.

Tape 3 current options

Most TM are equivalent. (just additional features)

TM emulates a DFA.

DFA $M = (Q, \Sigma, \delta, q_0, F)$. ← finite stuff.

↓ turn into a string

also appending string for DFA is now possible

Input to TM : $\langle D, w \rangle$

↑ DFA in string.

Check D is a DFA.

:

Language of this TM $\{\langle D, w \rangle \mid D$ is DFA in string$\}$

Church-Turing Thesis.

Alan Turing → Turing machine (automata)

Alonzo Church → notation system called λ-calculus

Turing machine = Turing machine (Turing complete)

Turing machine / λ-calculus can describe anything as an algorithm (math procedure)

with finite steps

Recognition

L TM recognises in a set string it accepts in finite time.

MUST YES → A TM M recognises L if it accepts $\forall w \in L$ in finite time.

Decidability

A TM decides L if it

MUST YES → accepts all $w \in L$ in finite time.

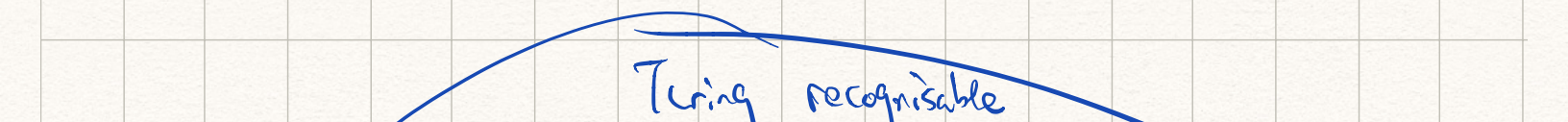
and NO - MUST NO and rejects all $w \notin L$ in finite time

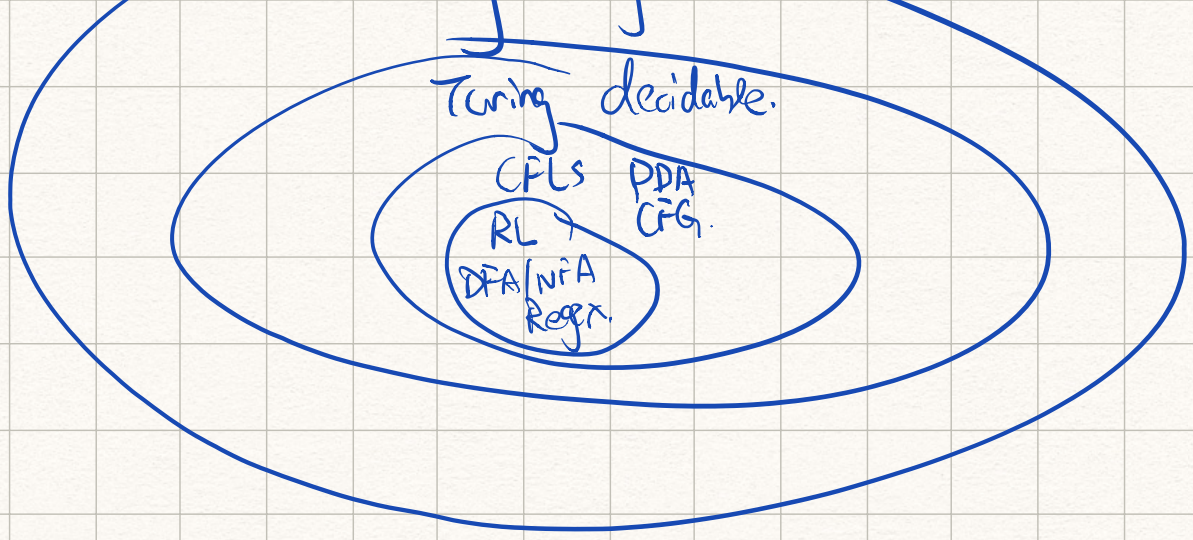
The class of lang recognised by a TM.

Turing recognisable / recursively enumerable

The class of lang decidable by a TM

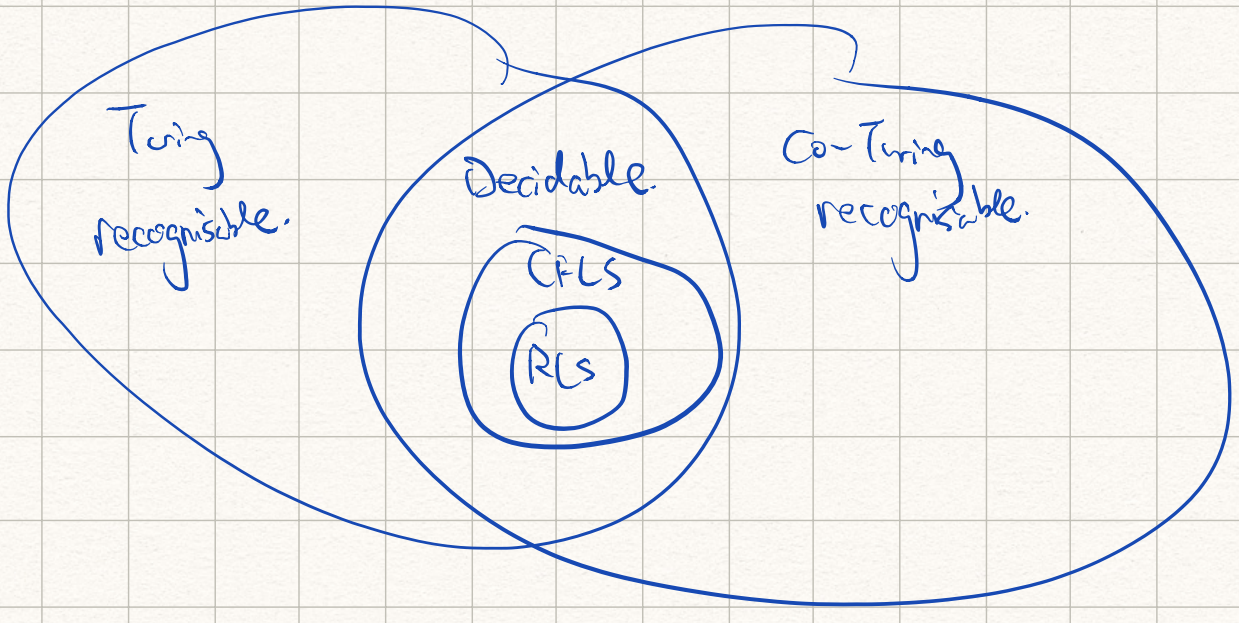
Turing decidable





Co-Turing recognisable

if a L is co-turing recognisable
$\forall w \notin L$, a TM it rejects w in finite time



Example.

$L = \{a^i b^j c^k \mid i*j=k,\ i,j,k \geq 1\}$

Decidable.

$L = \{ww \mid w \in \Sigma^*\}$. Decidable.

$\{\langle D, w\rangle \mid D = \text{DFA accepts string } w\}$

Decidable.

DFA Emptiness Testing.

$L = \{\langle M\rangle \mid M \text{ is a DFA and } L(M) = \emptyset\}$.

Enumerate strings in Σ and emulate DFA M. rejecting if a string is accepted.

↓

Co-turing recognisable. NO finite time

YES infinite time.

analyze the DFA find all reachable states

if one accept state is reachable accept.

if no accept state reject.

Decidable.

Chomsky normal form.

Type of grammer that:

start. S cannot appear on the right of any rule

ε. cannot appear on the right of any rule except S's

All rules except $S \to \varepsilon$ are either in the form.

$A \to a$.

$A \to BC$.

→ Limits the ability to produce ε's

→ Guarantees $|w| = n$ ⇑ string created by n non-terminals because $A \to \alpha$

Convert to chomsky normal form.

1. Add new $S_{new} \to S$ only appears once. on the left.

2. Get rid of $A \to \varepsilon$. by calculating all rules if $A = \varepsilon$.

e.g $B \to A1A$

$A \to \varepsilon$.

$B \to 1A \mid A1 \mid 1$.

3. Remove $A \to B$ (unit rules) by replacing B with Bs rules

4. Fix long non-terminal rules by introducing new non-terminals

Analysis of Greiner → CNF alg.

Finite time for each step

Finite count of work in 1~3

2-4 iterates to remove some work.

finite worked in general.

Example. $L = \{\langle G, w\rangle \mid G$ is a CFG that generates string $w\}$

Read and check $\langle G\rangle$.

Convert G → CNF.

Assume w has n chars

w is derived from Snew

in $(n-1) + n = 2n-1$ steps

(to non-terminals)

Check all derivations in $2n-1$. (finite)

Accept if w;

Guaranteed to stop in finite time

$L = \{\langle M, w\rangle \mid M \text{ is a TM that accepts } w\}$

a TM that simulates a TM.

is called universal TM.

L is not decidable.

Some TM loop infinitely

so U TM loop infinitely

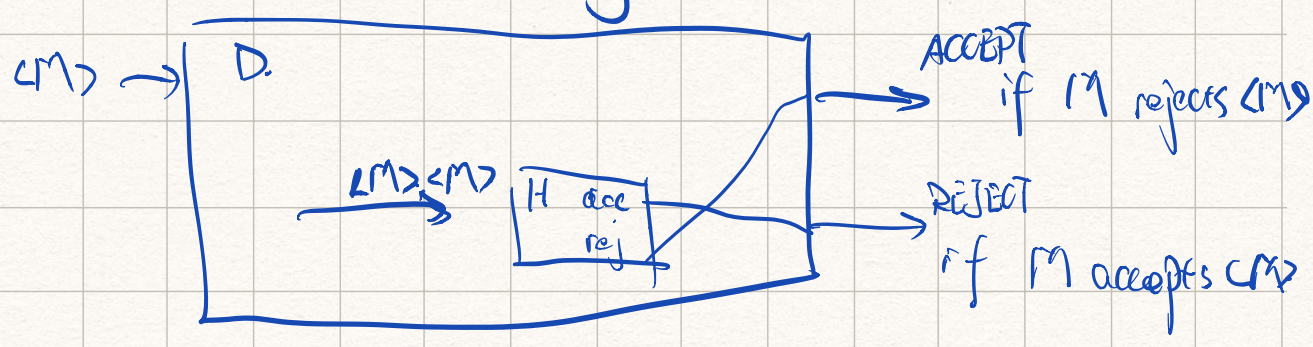
for some L.

Assume $L = \{\langle M, w\rangle \mid M \text{ is a TM accepts } w\}$

Call decider (deciding logic) of L(H)

Create a evil TM D that reads a.

TM in string.



if send <D> in, if H accepts/reject <D>

D should reject/accept <D>

Cause (adiction

L is not decidable.

Recognition

L may contain many strings

$\exists$ TM that accepts $\forall$ wEL in finite time

Unrecognisable.

$\exists L \; \forall TM \; M$ wEL that TM doesn't accept in finite time.

$L = \{\langle M, w\rangle \mid M \text{ is a TM that accepts } w\}$

Proved by contradiction L is undecidable.

But L is recognisable

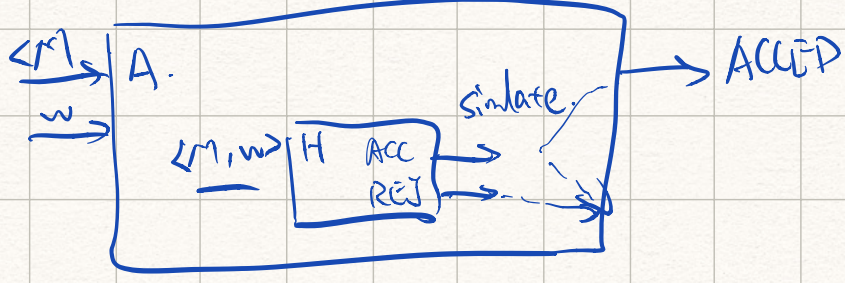
if M accepts w, L would accept in finite time.

Recog $\cap$ CoRecog = Decidable

Unknown: HALT$_{TM}$. Whether TM halts on w.

Known: A_{TM}. whether TM accepts w.

If Halt TM h exists

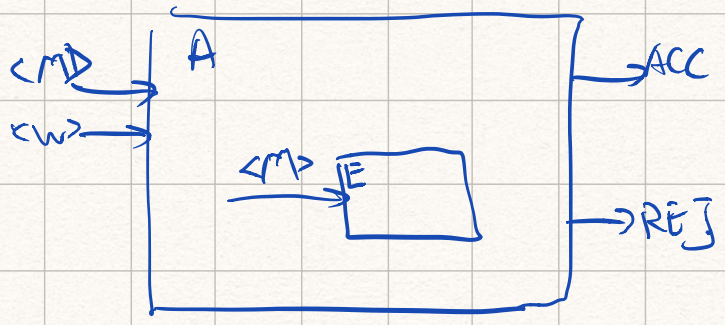


TM emptiness test.

$E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset\}$.

Known undecidable : A_{TM}: whether TM accept string w.

Unknown : E_{TM} : whether TM accepts anything



Alter $M \to M'$. compares input with w, reject if not w.

TM constructs a new TM M' on the spot.

If M' lag is empty, M' must have rejected w.

M rejected w.

If M' lag is not empty, M' must have accepted w

M accepted w.

Totally valid machine.

Contradicts to A_{TM} undecidable

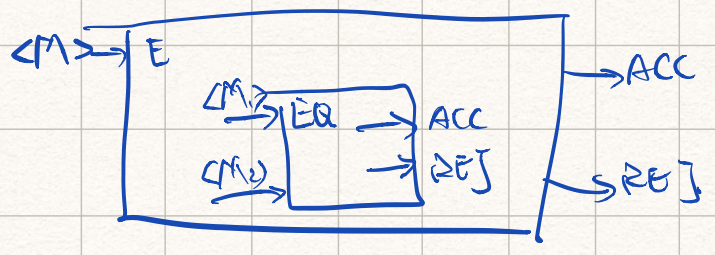
TM equality

$EQ_{TM} = \{\langle M_1, M_2\rangle \mid M_1$ and M_2 are TMs and $L(M_1) = L(M_2)\}$

known undecidable.

E_{TM}: whether TM M accepts anything.

Unknown EQ_{TM} : whether TM $M_1 = M_2$.



$EQ_{TM} \rightarrow E_{TM}$

Have E construct a TM that rejects M_ϕ.

run M and M_ϕ with EQ.

If equal, M is a M_ϕ, ACCEPT

otherwise reject.

Sets of TM with some property: generally undecidable.

DFA — decidable

PDA/CFG. depends

A_{TM}, $\langle (A, w)\rangle$: if A accepts w.

E $\langle A\rangle$ empty. reject all.

ALL. $\langle A\rangle$ sets of A accept Σ^*

EQ. $\langle A_1, A_2\rangle$ $L(A_1) = L(A_2)$.

	DFA/NFA	PDA/CFG	TM.
A	√	√	X
E	√	√	X
ALL	?	X	X
EQ	?	X	X

Prove undecidable.

① contradiction

② reduce (solve a undecidable problem ⇑ (known) with its decider)