Turing Machine

Σ Q q_0 H δ. — transition function

- Σ: symbols.
  - ␣ blank.
  - ▷ end of tape
- Q: states
- q_0: init state
- H: halting states

$$\delta : (Q \setminus H) \times \Sigma \to Q \times (\Sigma \cup \{\rightarrow, \leftarrow\})$$

decidable / recognisable

Formal Language.

give alphabets Σ.
formal language $L = \Sigma^*$

if $\chi_L(x) = \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{otherwise} \end{cases}$

M accepts input $x \in \Sigma^*$ if halts in state Yes N_o.
halting state

M rejects x if it halts in N.

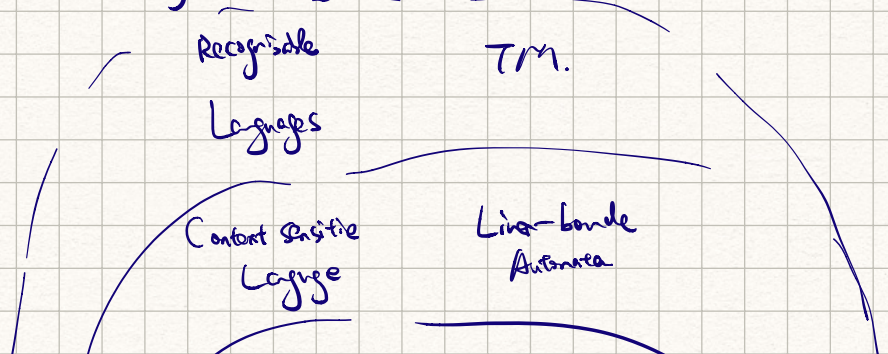
M decides L if When $x \in L$, M acc x.
$x \notin L$ M rej. x.

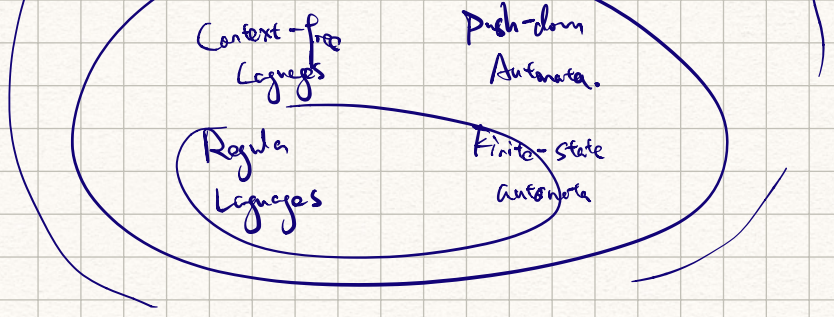
M regnise L if When $x \in L$, M halts
$x \notin L$, M doesn't halt

Decidable = Recursive = Computable.

Recognisable = Recursive Enumerable = Computably Enumerable.

The chomsky Hierarchy.





Church Turing Thesis.

Computable $\Longleftrightarrow$ Computable By a TM.

TM is equivalent in computability to
- "enhanced" TM.
- register machines
- Python, Java, C.
- classical computers
- quantum computers

Turing Machine with multiple tapes = TM.

only difference lies in δ transition.

$$\delta : (Q \setminus H) \times \Sigma^k \to Q \times (\Sigma \cup \{\leftarrow, \rightarrow, \in\})^k$$

Turing machine with two way infinite tapes = TM

Build a TM with two one-way infinite tapes

Non deterministic TM.

$$\delta : (Q \setminus H) \times \Sigma \times Q \times (\Sigma \cup \{\leftarrow, \rightarrow, \in\}).$$

= TM.

Closure properties.

L^- of L is Σ^* / L.

$L_1 L_2 = \{yz \mid y \in L_1 \text{ and } z \in L_2\}$

Decidable Languages are closure

under

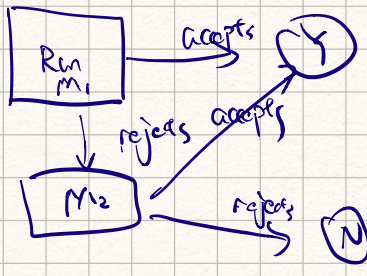
Complementation. 互补性.

M decides L if $x \in L$ accepts x, Y N

$x \notin L$ rejects x reverse.

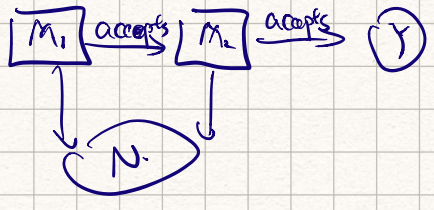
Close under union.

$L_1 \cup L_2$ is also decidable.



Close under intersection.

$L_1 \cap L_2$.



Close under concatenation

$L_1 L_2$

Recognizable L Closure

under concat.
intersection
union

NOT complementation. (could loop forever for L^-)

Register Machines

Turing machine only access sequentially on tapes.

now introduce a computation model — (unlimited register machines URM)

URM
- infinite registers $R_1 - \cdots R_n$
- each stores a natural number r_n.

URM compute functions $N^k \to N$.

if computation halts, output r_1

URM 4 types of instructions.
- Zero(n) set $R_n \to 0$
- Sucessor(n) $+1 \to R_n$.
- Move(n, m) $R_m = r_n$.
- Jump(n, m, p) Program. go to Ip when $r_n = r_m$.

Computing with URM.

$P = I_1 \cdots I_j$.

Eg. with TM.

TM simulates URM.

Tape 1 program counter.
Tape 2 code.
Tape 3 each register, seperated by ␣
Tape 4-6 cache

1. tape 1 finds curr instruction in tape 2.
2. Execute inst.
3. Repeat

Each instruction but Jump modifies tape 3.

we need to use the auxiliary tapes

URM simulates TM.

$\Sigma = \{0, 1, \triangleright, \sqcup\}$.

encode them $code(\sqcup) = 0$
$code(\triangleright) = 1$.
$code(0) = 2$.
$code(1) = 3$

Store the contents as base-4 numbers

$1 \times 4^0 + 0 \times 4^1 + 2 \times 4^2 + 0 \times 4^3 + 3 \times 4^4$
$\triangleright \quad \sqcup \quad 0 \quad \sqcup \quad 1$

Set up registers in terms of M.

Q-IN$_i$	σ-IN$_i$	Q-OUT$_i$	σ-OUT$_i$
q_{in}	σ_{in}	q_{out}	σ_{out}

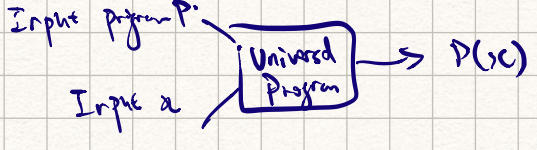
OTHER registers
TAPE HEAP-POS
STM
STATE

A WHILE Lgauge.
high-level language $\equiv$ TM/URM

Syntax
$x := 3$.
while $X \neq Y$ do
PROGRAM 1; PROG 2;
$Z = X + Y$

$=$ TM

Universal Turing Machine



Gödelisation, interpret the program itself
encode

UTM $y(M, x) = M(x)$

Encoding Turing Machine

$Q \Rightarrow q_0 \cdots q_n$.

Σ $\sigma_0 = \sqcup$ $\sigma_1 = \triangleright$ $\sigma_2 = \rightarrow$ $\sigma_3 = \leftarrow$

$\sigma_4, \sigma_5 \cdots$

encode them as numbers of 1s

$t = \langle q_i, \sigma_n, q_j, \sigma_m \rangle$

$code(t) = code(q_i) \, 0 \, code(\sigma_n) \cdots$

$code(\delta) = code(t_1) \, 0 \, code(t_2) \cdots$

It's possible TM same type of job different coding (state difference)

Nonetheless coding 1-1.

two machines different string!

Str $\{0,1\}$ decide if it is some code of TM

decidable!

UTM.

Tape 1. tape of M

Tape 2 $code(M)$. ← input of M

Tape 3 current state

Prep.

1. figure out | code(M) | code(x) |
2. code(M) on tape 2.
3. $\triangleright \sqcup$ code(x) on tape 1.
4. q_0 tape 3

Search M for a tuple where first matches tape 3 state. Second matches. tape 1 symbol.

found! update tape 1. with new value.

and its position.

tape } update state.

if M reaches a halt state halt

UTM can bootstraps!

Limit of TM

cannot solve by a well-defined procedure

Not by a C++/Python program

Decided by a TM ~ Solvable

only recognised by a TM. ~ Unsolvable
semi-decidable

Not even recognised ~ unsolvable

The halting problem

is recognisable.

UTM, M(x).

↑

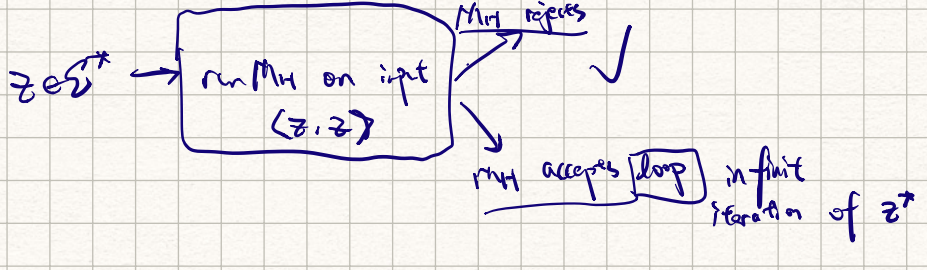
enumerate over M.

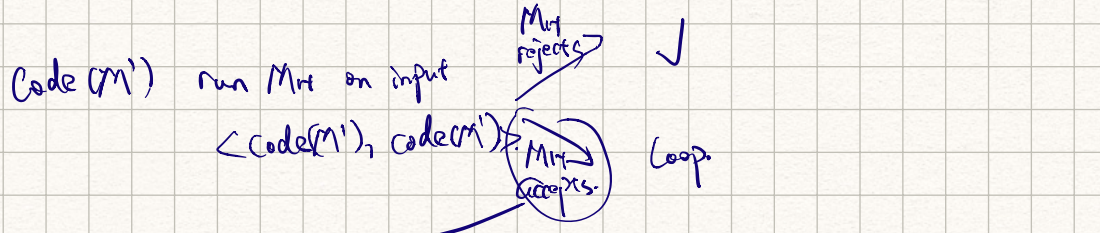
if M(x) halts, then accept

Proof. The halting problem.

Now we are able to define a new TM called M.

⟨code(M,x)⟩ → M_H ↗ ✓ M halts on x
↘ ✗ M doesn't halt





if M_H accepts c(M') c(M') — M' doesn't halt c(M') — M_H should rejects

<Code(M'), code(M')> ∉ HALT.

if M_H rejects c(M') c(M') — M' halts on c(M') — M_H should accept.

<Code(M'), Code(M')> ∈ HALT. problem

M_H doesn't exist

Un recognisable problem

$\overline{HALT}$

Observation 1

if L, $\bar{L}$ recognisable — L is decidable

HALT is recognisable but not decidable.

↑ not closed under complementation

If we recognise $L_\cdot$, then we would decide L'.

As L' is undecidable, the we cannot regonise $L_\cdot$.

i.e. if $L_\cdot$ ∈ $\overline{HALT}$, $L_\cdot$ ∈ HALT

HALT ∈ $L_\cdot$ ∈ $\overline{HALT}$

Empty tape halting problem

if M_{ETH} exists

M_H

On input (y, x).

if $y \neq$ code(M) for all M reject.

if $y =$ code(M), construct $M(m, x)$

it enters a loop on any ξ.

it write x on tape and simulate M on x.

use $M_{\#H}$ on $M_{m,x}$

mapping-reduction. if a TM converts $x \in L' \to f(x) \in L$. $L' \leq L$.

$L' \leq L$ and L is decidable. L' is decidable.

$L' \leq L$ and L' is undec. — L is undecidable

if L is decidable L' is not $L' \not\leq L$

Hierarchy of problems.

... Decidable

HALT UnDecidable / recognisable.

$\overline{\text{HALT}}$ Not recognisable / but complement recognisable.

EQ. Not recognisable complement $\overline{\text{EQ}}$

$$\text{HALT} \leq \overline{\text{EQ}} \geq \overline{\text{HALT}}$$

Decidable.

Other reducibility.

Turing reducability.

An oracle for L is an answer for $x \in L$?

L' is turing reducible to L if we can decide L' given L's oracle

mapping reducibility → Turing reducibility.

TM language property. P is a function from a set of TM → {0, 1}.

Property is non trivial. if $TM(M) \to P(M)=1$.
$TM(M') \to P(M')=0$

If P is a non trivial language property, Does M have P is undecidable.

Language Properties

Structural properties M has 13 states.
typically decidable

Cantor's diagonalisation argument

Countably infinite if $\exists f: \mathbb{N} \to S$
↓
natural number.

Set of all Turing machine / recognisable languages

Countably infinite

Uncountably infinite. problems that are unrecognisable

Definition 1. tractable problem $O(n^k)$
p-time algorithm

Definition 2 decision problem → (0, 1)

Polytime reduction.

Definition 3

A, B problem coded in string with Σ.

M is a deterministic TM.

give an instance w of problem A.

$f_M(w)$ exists.

w is a yes/no instance of A

$f_M(w)$ is a yes/no instance of B

A reduces to B

if M terminates in time $O(n^k)$ for some fixed k.

$n = |w|$. $A \to B$ in p-time.

$A \leq_p B$ A is no harder than B

Properties of p-time reduction

Reflexive $A \leq_p A$

Transitive $A \leq_p B \leq_p C$.

$A \leq_p C$

Symetry X.

$A \leq_p B$

$B \leq_p A$

$A \equiv_p B$

Example Hamilton Circuit Problem.

$HCP \leq_p TSDP$.

✓

G contains a HC.

Make G', if (m,n) is a edge of G. $G'(m,n) = 0$

if (m,n) is not ... $G'(m,n) = 1$.

$(G', 0)$ is an instance of TSDP

Roughly.

an dfp solves TSDP → solve HCP

no fast dfp for HCP → no fast dfp for TSDP

HPP → Path.

no cycle required just a path

HCP $\leq_p$ HPP

HCP $\leq_p$ DHCP

↓ Directed.

DHCP $\leq_p$ HCP.

p-time equivalence.

$A \leq_p B$ and $B \leq_p A$ $A \equiv_p B$.

$A \equiv_p A$ $A \equiv_p B \equiv_p C$ $A \equiv_p B$

$A \equiv_p C$. $B \equiv_p A$.

Complexity class P Problem A is in P if ∃ a DTM M that

M accepts all yes-instances of A.

rejects all no — A

runs in p-time

9. Non-deterministic TM.

a state can have multiple transitions to next.

* M accepts $w \in \Sigma^*$ if some run halts and succeeds
* M rejects w if every run halts and fails
* otherwise f_M is undefined on w.

a deterministic TM is a specific NDTM where choice ≤ 1.

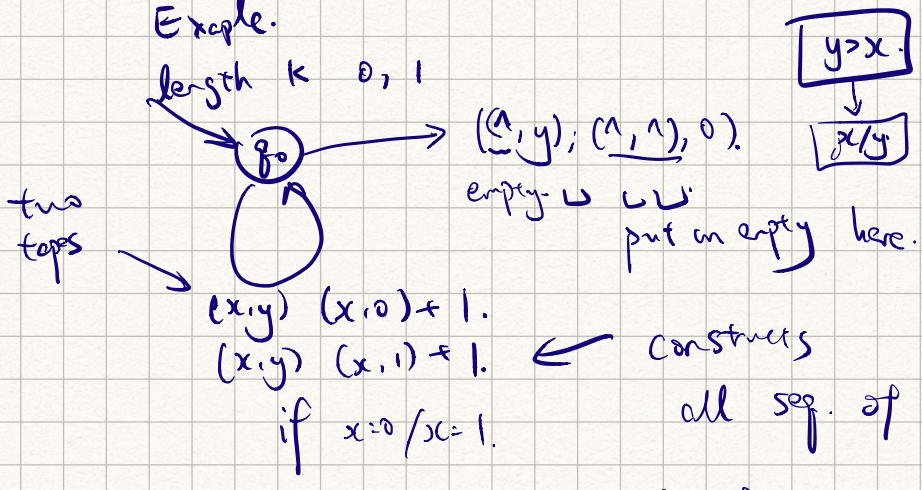
runtime of a non-deterministic TM.

$$t_M(n) = \begin{cases} \max \text{ runtime of } M, \text{ on any input size } \leq n \\ \infty, \text{ no max} \end{cases}$$

The class NP. — non-deterministic TM — in P time $O(n^k)$ for a fixed k.

Example.

length k 0, 1



← constructs k. all seq. of 0, 1.

then test if $Y < X$ input.

arbitrary string

then test division

The longest possible run. is $O(k^2)$ (length of input)

Another Example. HCP belongs to NP.

HCP (V, E)

— pick $v \in V$.

path = $[v]$

while $|path| < |V|$. pick a next.

{ pick $w \in V$ \ path, $(w, head(path)) \in E$

path = $(w : path)$

}

If $(v, head(path)) \in E$, then accept else reject

max runtime $|V|$. → HCP $\in$ NP.

Determinism vs. Non-D. ——.

BFS to iterate though non-deterministic TM.

↑ t steps

then d^t steps for a deterministic TM

P=NP decision problem solved in polytime by a nond TM.
also can be solved in polytime. by a dTM

if not P≠NP.

Complementation.

$\bar{A}$ A.

yes / no instance reverse.

If $A \in P$ then $\bar{A} \in P$

Definition 5. $A \in$ co-k $\Leftrightarrow \bar{A} \in K$. $A \in K \Leftrightarrow \bar{A} \in$ co-K.

↑ complexity class.

Validity problem for prop. logic is co-NP

to test if ϕ FAILS to be valid. is NP

- class P is minimal with ordering $\leq_P$

if $A \in P$ then $A \leq_p B$

- if $B \in NP$ and $A \leq_p B$ then $A \in NP$

let TM T be a $A \to B$.

TM M be a non-d TM for B.

T; M solves A.

instance of $A - n$

$T \sim p(n)$.

$T; M \to q(p(n))$.

total time $p(n) + q(p(n))$ still polynomial

$T; M$ is a non-D TM. Thus $A \in NP$

NP-complete.

- hardest problems in NP

$A \in NP$ and $\forall B \in NP \quad B \leq_p A$.

If a decision problem only satisfies $\forall B \in NP \quad B \leq_p A$, it's NP-hard.

$NPC = NP \cap NP\text{-hard}$.

All NPC problems are p-time equivalent to each other

$A \leq_p B$ ⟸ NPC problem — vice versa.

↑

NP problem.

$A \equiv_p B$.

if P=NP NP problems are P-time equivalent.

then A, B are both $\in P$

$A \to$ p-reduce

use p-time yes/no?

if yes. then write yes B

no — no B

p-time

Ex. 11. Let T be a D.TM that solve $X \in NPC$.

Let $B \in NP$ $X \in NPC$ $B \leq_p X$. DTM

DTM ← $M; T$. solves B

runs in p-time

with DTM

Hence $P = NP \Leftarrow B$ is a NP

Proving NP-completeness

Assume PSAT $\in$ NPC. to prove $A \in$ NPC.

prove $A \in NP$

and $PSAT \leq_p A$.

since $B \leq_p PSAT \leq_p A$

↑ all NP problems.

so $B \leq_p A$ any NP problem.

PSAT. → instance.

prop logic.

operator can be $\vee, \wedge, \rightarrow$.

$\phi ::= prop \mid \neg form \mid form \circ form$

ϕ is yes-instance if it's satisfiable.

no — — — — not.

CNF-SAT.

↑ conjunctive normal form.

n-SAT. a special case of CNF-SAT.

each CNF element is comprised of n literals

$(p \vee \neg p \vee \neg q) \wedge (p \vee \neg q \vee \neg q)$ — — — no-instance.

3-SAT.

To show $PSAT \leq_p CNF\text{-}SAT \leq_p 3\text{-}SAT \leq_p HCP \leq_p TSDP$

$PSAT \leq_p CNF\text{-}SAT$ High level: ϕ has a CNF f

1. eliminate $\rightarrow$
2. replace $\neg\neg\phi \rightarrow \phi$

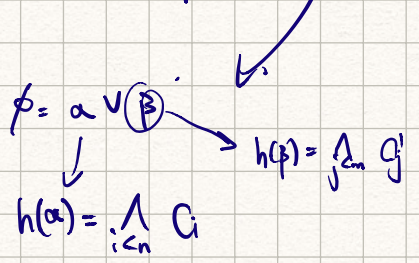
$\neg(\psi \vee \theta) \rightarrow \neg\psi \wedge \neg\theta$. — .

The formula obtained is either literal.

Conjunction

disjunction.

simple


$\phi = \alpha \vee \beta$
$h(\beta) = \bigwedge_{j \le m} g_j$
$h(\alpha) = \bigwedge_{i \le n} c_i$

CNF $\le_p$ 3-SAT