Big-O notation

$f, g: \mathbb{R} \to \mathbb{R}$ (or: $\mathbb{N} \to \mathbb{R}$)

$f = O(g)$: $(\exists c>0)(\exists N)(\forall n \geq N)(|f(n)| \leq c \cdot g(n))$

Example $2n\log n + 7n - 6 = O(n \log n)$.

$3n^2 + 5n + 7 = O(n^2)$

Properties. 1) If $f_1 = O(g_1)$ and $f_2 = O(g_2)$, then $f_1 \cdot f_2 = O(g_1 \cdot g_2)$;

$|f_1(n)| \leq c_1 g_1(n)$, $\forall n \geq N_1$, $|f_2(n)| \leq c_2 (g_2(n))$, $\forall n \geq N_2$.

$\Rightarrow |f_1(n) f_2(n)| \leq c_1 c_2 |g_1(n) g_2(n)|$, $\forall n \geq \max\{N_1, N_2\}$.

2) $f \cdot O(g) = O(f \cdot g)$.

3) If $f_1 = O(g_1)$, $f_2 = O(g_2)$, then $f_1 + f_2 = O(\max(g_1, g_2))$

4) If $f_1 = O(g)$, $f_2 = O(g)$, then $f_1 + f_2 = O(g)$.

$f_1 = O(g)$, $f_2 = O(g)$ 并不意味着 $f_1 = O(f_2)$. 例: $n = O(n^2)$, $\log n = O(n^2)$, 但 $n \neq O(\log n)$.

5) If c is a nonzero constant, and $f = O(g)$, then $c \cdot f = O(g)$.

$O(1)$: constant. $O(\log\log n)$: double logarithmic

$O(\log n)$: logarithmic $O((\log n)^c)$, $c>1$: polylogarithmic. (log 的多项式)

$O(n)$: linear $O(n \log n)$: $n \log n$.

$O(n^c)$, $c>0$: polynomial (认为多项式的).

$O(c^n)$, $c>1$: exponential $O(n!)$: n factorial

Big-Ω: $f = \Omega(g)$: $(\exists c>0)(\exists N)(\forall n \geq N)(|f(n)| \geq c \cdot g(n))$

Big-Θ: $f = \Theta(g)$ if $f = O(g)$ and $f = \Omega(g)$.

$(\exists c_1, c_2 > 0)(\exists N)(\forall n \geq N)(c_1 g(n) \leq |f(n)| \leq c_2 g(n))$

Small-o: $f = o(g)$ if $(\forall \varepsilon > 0)(\exists N)(\forall n \geq N)(|f(n)| \leq \varepsilon \cdot g(n))$

Small-ϑ: $f = \vartheta(g)$ or $f \sim g$ if $(\forall \varepsilon > 0)(\exists N)(\forall n \geq N)(|f(n) - g(n)| \leq \varepsilon f(n))$

Example. 1) $f(n) = 100\ n \log n$, $g(n) = n^2$

$f(n) = O(g(n))$, $f(n) = o(g(n))$, $g(n) = \Omega(f(n))$

2) $f(n) = 6n^2 + 100n + 50$, $g(n) = n^2$

$f(n) = \Theta(g(n))$

3) $f(n) = n^{100}$, $g(n) = 2^n$

$f(n) = O(g(n))$, $f(n) = o(g(n))$

4) $f(n) = n^{O(1)}$: $(\exists c > 0)(f(n) = O(n^c))$. or poly.(n).

5) $f(n) = 2^{\Omega(n)}$: $((\exists c > 0)(\exists N)(\forall n \geq N)(|f(n)| \geq 2^{cn}))$

6) $f(n) = 2^{(\log n)^{O(1)}}$: quasipolynomial

$(\exists c > 0)(\exists N)(\forall n \geq N)(f(n) \leq 2^{(\log n)^c})$

Alphabet and Language

Example. 1) Integer ~~Duplication~~ multiplication. Given $x, y \in \mathbb{N}$, compute $x \cdot y$.

2) Primality testing. Given $p \in \mathbb{N}$, test if p is a prime.

3) Hamiltonian path. Given a graph G, test if G has a Hamiltonian cycle.

An <u>alphabet</u> is a set of symbols.

Roman alphabets $\Sigma = \{a, b, \ldots, z\}$

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

A <u>string</u> over an alphabet is a finite sequence of symbols from the alphabet.

<u>Empty string</u> is a string with no symbols, denoted by $\emptyset$, ε

The set of ~~strin~~ all strings over Σ is denoted by Σ^*

$\Sigma = \{0, 1\}$, $\Sigma^* = \{\emptyset, 0, 1, 00, 01, 10, 11, 000, \ldots\}$, $\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$, where Σ^n denotes the set of strings of length n.

The length of a string is length as a sequence, denoted by $|w|$.

For example, $|\emptyset| = 0$, $|01011| = 4$.

Two strings x, y can be combined to form a third by concatenation, denoted by xy.

String v is a substring of w iff $\exists x, y \in \bar{z}^*$, s.t. $xvy = w$.

String v is a suffix of x if $\exists s \in \bar{z}^*$, s.t. $sv = x$

String v is a prefix of x if $\exists s \in \bar{z}^*$, s.t. $vs = x$.

For string $w \in \bar{z}^*$, $i \in \mathbb{N}$, $w^0 = \emptyset$, $w^{i+1} = w^i w$ for $i \geq 0$

The reversal of a string w, denoted by w^R, is the string "spelled backward". Formal definition by induction on length:

1) If $w = \emptyset$, $w^R = w = \emptyset$.

2) If w has length $n+1$, when $w = ua$, $a \in \bar{z}$, then $w^R = au^R$.

A language is a set of strings over an alphabet. Language $L \subseteq \bar{z}^*$

Eg. $\bar{z} = \{0, 1\}$, $L_{even} = \{0, 10, 100, 110, 1000, \ldots\}$

Let L be a language. The complement of L, denoted by $\bar{L}$, is $\bar{z}^* - L$.

So $\bar{\bar{L}} = L$.

If L_1, L_2 are languages, the concatenation of L_1 and L_2 is $L_1 L_2 = \{w \in \bar{z}^* \mid w = xy$ for some $x \in L_1, y \in L_2\}$

The star of L, denoted by L^*, is the set of strings by concatenating zero or more strings from L, i.e.

$$L^* = \{ w \in \Sigma^* \mid w = w_1 w_2 \cdots w_k \text{ for } k \geq 0 \text{ and } w_1, w_2, \ldots, w_k \in L \}$$

Write L^+ for the language LL^*, i.e.,

$$L^+ = \{ w \in \Sigma^* \mid w = w_1 w_2 \cdots w_k \text{ for } k \geq 1 \text{ and } w_1, w_2, \ldots, w_k \in L \}$$

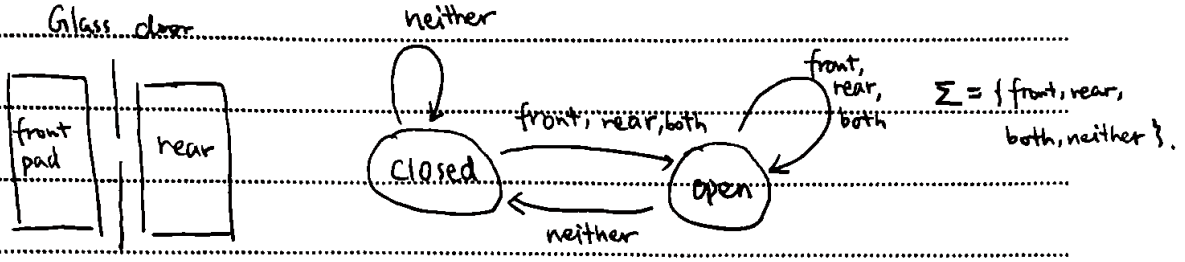
Through encoding, any decision problem becomes a language; any computation problem becomes a function from Σ^* to Σ^*

讨论复杂度时，说的是某个语言的复杂度

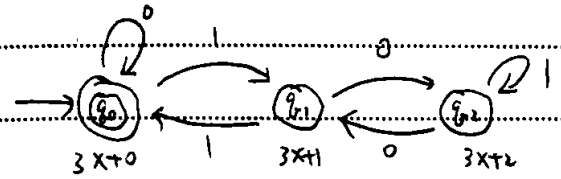
$\Sigma = \{0,1\}$, $\Sigma^* = \{\varepsilon, 0, 1, 00, \ldots\}$, $|\Sigma^*| = \aleph_0$, # languages $= 2^{\aleph_0} = \aleph_1$

Finite automaton

Ex Glass door



$\Sigma = \{$front, rear, both, neither$\}$.



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

$\rightarrow$O : start state. ◎ : accept state

0 0: q_0 q_0 accept

1 1: q_0 q_1 reject

2 10: q_0 q_1 q_2 reject

3 11: q_0 q_1 q_0 accept ~~reject~~

4 100: q_0 q_1 q_2 q_1 reject

5 101: q_0 q_1 q_2 q_2 reject

6 110: q_0 q_1 q_0 q_0 accept 检查3的倍数.

$L = \{a \mid a = a_1 a_2 \cdots a_n,\ a_1 \cdot 2^{n-1} + a_2 \cdot 2^{n-2} + \cdots + a_n \equiv 0 \pmod 3\}$.

Lemma. On input $a_1 a_2 \cdots a_i$, the DFA ends on state q_r, where $r \equiv (a_1 a_2 \cdots a_i)_2 \pmod 3$

Proof. Induction on i. Base step i=0. true.

Induction step. assume it's true for i, prove for i+1.

Let $(a_1 a_2 \cdots a_i)_2 \equiv r \pmod 3$. Then $(a_1 a_2 \cdots a_i a_{i+1})_2 \equiv (2r + a_{i+1}) \pmod 3$.

So. if $r=0$, $a_{i+1}=0$, then $(2r + a_{i+1}) \equiv 0 \pmod 3$. $q_0 \rightarrow q_{r0}$

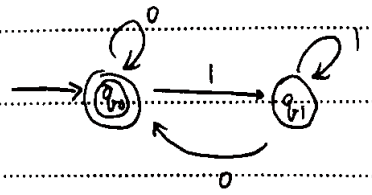
$r=0$, $a_{i+1}=1$, then $(2r + a_{i+1}) \equiv 1 \pmod 3$. $q_0 \rightarrow q_1$

$r=1$, $a_{i+1}=0$, then $(2r + a_{i+1}) \equiv 2 \pmod 3$ $q_1 \rightarrow q_2$

$r=1,\ a_{i+1}=1\ ,\ (2r+a_{i+1}) \equiv 3 \equiv 0 \pmod 3$ $\quad q_{r1} \to q_{r0}$

$r=2,\ a_{i+1}=0,\ (2r+a_{i+1}) \equiv 1 \pmod 3$ $\quad q_{r2} \to q_{r1}$

$r=2,\ a_{i+1}=1,\ (2r+a_{i+1}) \equiv 5 \equiv 2 \pmod 3$ $\quad q_{r2} \to q_{r2}$ □



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

$L = \{w \in \Sigma^* \mid w = w_1 w_2 \cdots w_n,\ w_i \in \{\Sigma \text{ for each } i,\ w_n = 0\}$

~~A formal~~

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

1. Q is a finite set called states
2. Σ is the alphabet
3. $\delta: Q \times \Sigma \to Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the accept states

Formalization: $Q = \{q_{r0}, q_{r1}\}$, $\Sigma = \{0,1\}$

δ:

δ	0	1
q_{r0}	q_{r0}	q_{r1}
q_{r1}	q_{r0}	q_{r1}

, & $F = \{q_{r0}\}$.

Language $A \subseteq \Sigma^*$, let M be a DFA. If A is the set of all strings that M accepts, we say that M recognizes A or M accepts A, denoted by $L(M) = A$.

If M accepts no strings, it recognizes the empty language

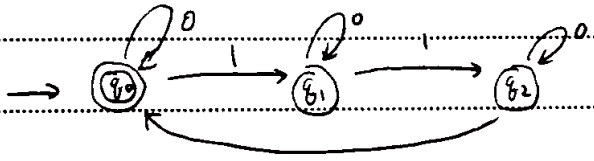
$\{\varepsilon\}$ is NOT the empty language $\emptyset$.

A language is called a regular language if some finite automaton recognizes it.

let $\Sigma = \{0, 1\}$. consider

$L_1 = \{w \in \Sigma^* \mid w = w_1 w_2 \cdots w_n,\ n \geq 2,\ w_1 \neq w_n\}$ is recognized by

$L_2 = \{w \in \Sigma^* \mid$ the number of 1's in w is a multiple of 3 $\}$. is recognized by



Lemma 1.25 If A_1 and A_2 are regular languages, so is $A_1 \cup A_2$

Proof: Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$. Construct $M = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = Q_1 \times Q_2 = \{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$
2. Σ is the same.
3. $\delta: Q \times \Sigma \to Q$, $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
4. $q_0 = (q_1, q_2)$
5. $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$. □

↓ if $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\}$, then M recognizes $A_1 \cap A_2$

Theorem 1.26 If A_1 and A_2 are regular languages, then so is $A_1 \circ A_2$

Concatenation $A_1 \circ A_2 = \{w \in \Sigma^* \mid \exists u, v \in \Sigma^* \text{ s.t. } w = uv \text{ and } u \in A_1, v \in A_2\}$.

$w = w_1 w_2 \cdots w_n$

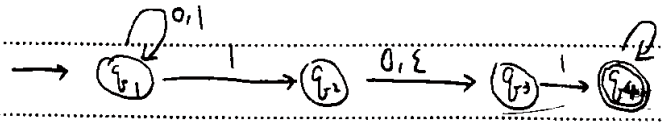
u v

↳ 不知道应该断开的位置.

Nondeterministic Finite Automaton (NFA)

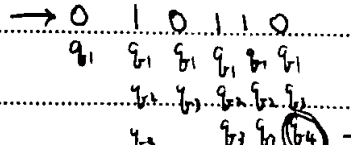
DFA: next state is determined

NFA: several choices may exist



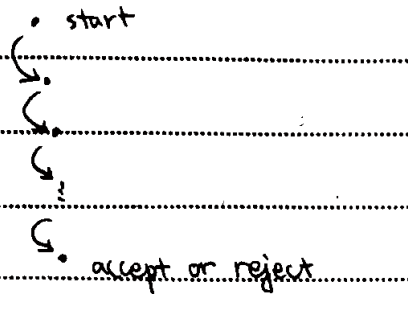
每次转移都能「猜对」以进到接受状态.

On input 010110:

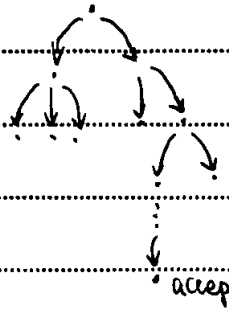


→ accepts if at least one computation branch ends in an accept state

Deterministic



Nondeterministic

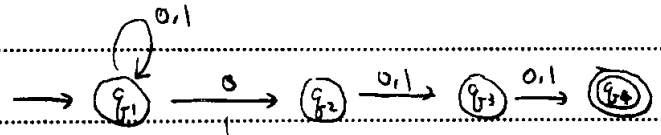


If any of these copies is in an accept state (at the end of the input), the NFA accepts the string.

By definition, a ~~NFA~~ DFA is also an NFA.

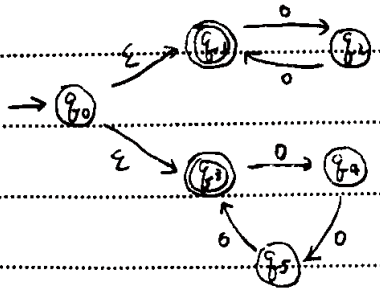
Example: 1) Design an NFA that recognizes

$L = \{w \in \{0,1\}^* \mid w = w_1 w_2 \cdots w_n \text{ and } w_{n-2} = 0\}$.



不确定性：猜是不是倒数第三个.

2) $\Sigma = \{0\}$, $L = \{0^k \mid k$ is a multiple of 2 or 3$\}$.



Formal definition of NFA

For any set Q, write $P(Q)$ to be the collection of all subsets of Q.

called power set

An NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where all but δ are the same as in DFA, and

$$\delta: Q \times \Sigma_\varepsilon \to P(Q) \qquad \Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$$

Say N accepts w if we can write $w = y_1 y_2 \cdots y_m$, where each $y_i \in \Sigma_\varepsilon$ and $\exists r_0, r_1, \ldots, r_m \in Q$, s.t. 1) $r_0 = q_0$

2) $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0, 1, \ldots, m-1$

Equivalence of DFA and NFA.

Clearly, any language recognized by a DFA can be recognized by an NFA

lemma 1.39 Every NFA has an equivalent DFA

Proof. Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing A. We construct a DFA $M = (Q', \Sigma, \delta', q_0', F')$ recognizing A.

For $R \subseteq Q$, let $E(R) = \{q \in Q \mid q$ can be reached from R by traveling along 0 or multiple ε's$\}$, 本课 $0 \xrightarrow{\varepsilon} 0$.

Define M as follows:

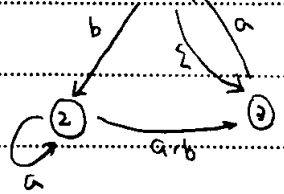
1) $Q' = P(Q)$

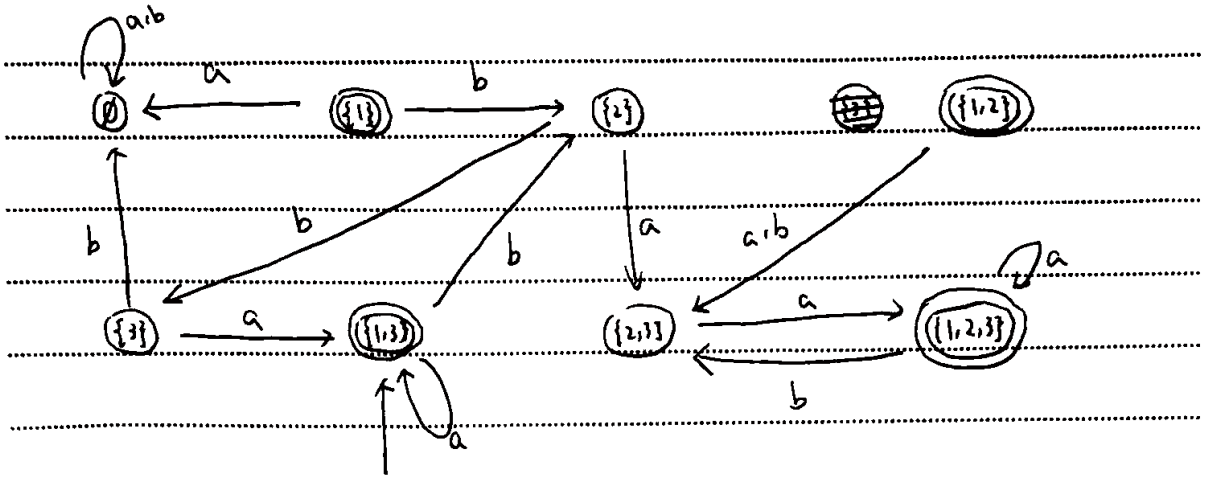
2) For $R \in Q'$, $a \in \Sigma$, $\delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a))$

3) $q_0' = E(\{q_0\})$

4) $F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$. □

5 → ① 等价的 DFA:

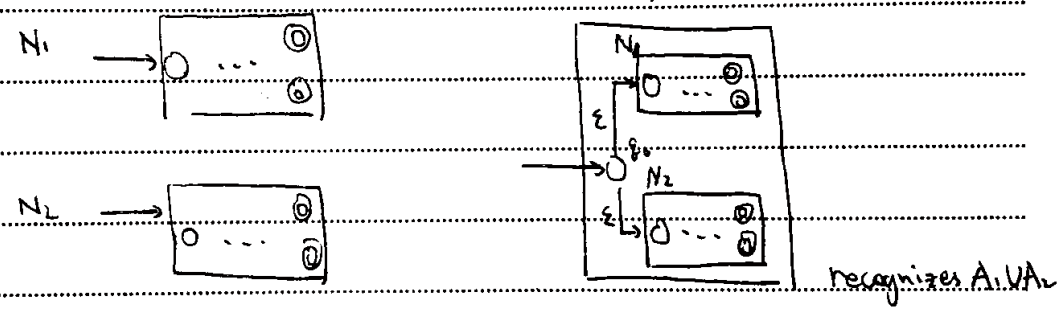




Properties of regular languages

Lemma 1.45: If A_1 and A_2 are regular languages, so is $A_1 \overset{U}{\circ} A_2$ (proof by NFA):

Proof



Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ ~~reg~~ recog. A_1

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recog. A_2.

Construct $N = (Q, \Sigma, \delta, q_0, F)$:

1) $Q = Q_1 \cup Q_2 \cup \{q_0\}$.

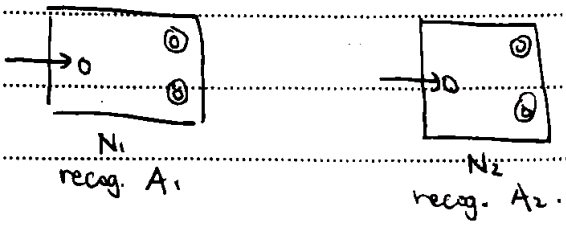
2) q_0 is the start

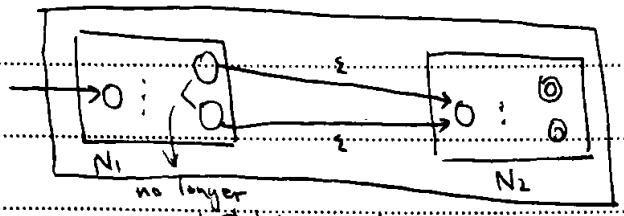
3) $F = F_1 \cup F_2$

4) For $q \in Q$ and $a \in \Sigma_\varepsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a), & q \in Q_1, \\ \delta_2(q,a), & q \in Q_2, \\ \{q_1, q_2\}, & q = q_0 \text{ and } a = \varepsilon, \\ \emptyset, & q = q_0 \text{ and } a \neq \varepsilon. \end{cases}$$

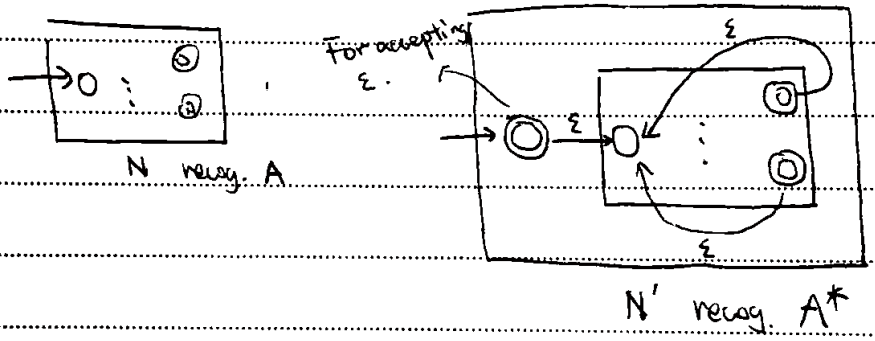
Theorem 1.47. If A_1 and A_2 are regular languages then so is $A_1 A_2$.





□

Theorem 1.49 If A is a regular language, then so is A^*



□

Regular languages are closed under the following operations

1. union 2. intersection 3. complement 4. concatenation.

5. star.

Regular expression

$(0 \cup 1)0^* = \{0, 1, 00, 10, 000, 100\}$

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

$(0\Sigma^*) \cup (\Sigma^* 1)$: strings that begin with a 0 or end with a 1.

Def 1.52 R is a regular expression if R is

1) a for some $a \in \Sigma$
2) ε
3) $\emptyset$
4) $(R_1 \cup R_2)$, where R_1, R_2 are regular expressions
5) $(R_1 R_2)$, where R_1, R_2 are regular expressions
6) (R_1^*), where R_1 is a regular expression.

Parentheses may be omitted when omission causes no confusion.

regular language : a language recognizable by a DFA / NFA

regular expression : a pattern that matches a specific set of strings

to the set of all regular languages is countable

Example. regular expressions

1. $0^*1\,0^*$ 2. $\Sigma^*1\Sigma^*$ 3. $\Sigma^*001\Sigma^*$ 4. $1^*(01^+)^* = \{$ $w \in \{0,1\}^*$ | every 0 is followed by at least one 1. $\}$

5. $(\Sigma\Sigma)^* = \{w \mid |w| \text{ is even}\}$ 6. $(\Sigma\Sigma\Sigma)^* = \{w \mid |w| \text{ is a multiple of 3}\}$

7. $01 \cup 10 = \{01, 10\}$ 8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w$ starts and ends with the same symbol$\}$

9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$ 10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{01, 0, 1, \varepsilon\}$

11. $1^*\emptyset = 1^*$ 12. Let $\emptyset^* = \{\varepsilon\}$ 13. $\emptyset^+ = \emptyset$

Exercise

1. all strings that contain 110 as a substring

$\Sigma^* 110\, \Sigma^*$

2. all strings that do not contain 00 as a substring

~~$\Sigma^*00\Sigma^*$~~ $(1 \cup 01)^*(0 \cup \varepsilon)$

3. the number of 1's is a multiple of 3

$(0^*1\,0^*1\,0^*1\,0^*)^* \cup 0^*$

4 contain at least two 1's and at least one 0.

$\Sigma^*\left((0\Sigma^*1\Sigma^*1) \cup (1\Sigma^*0\Sigma^*1) \cup (1\Sigma^*1\Sigma^*0)\right)\Sigma^*$

Theorem 1.54 A language is regular if and only if some regular

expression describes it.

Given two regular expressions R_1, R_2, check if $L(R_1) = L(R_2)$

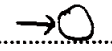
1. Find the DFAs M_1, M_2 recognizing $L(R_1), L(R_2)$ respectively
2. ~~Check~~ Construct a DFA M recognizing
$$(L(R_1) - L(R_2)) \cup (L(R_2) - L(R_1))$$
3. Check if $L(M) = \emptyset$, ie. if the accept states can be visited. This is a graph theoretic problem.

Lemma 1.55 If a language is described by a regular expression, then it is a regular language.

regex R describing L $\rightarrow$ NFA recog. L

If: 1. $R = a$, $a \in \Sigma$. $L(R) = \{a\}$.

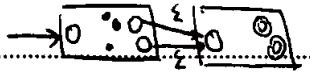
2. $R = \varepsilon$.



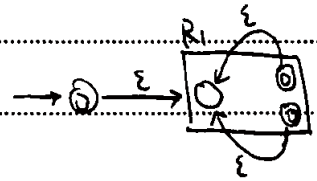
3. $R = \emptyset$

4. $R = R_1 \cup R_2$

5. $R_\circ = R_1 \circ R_2$

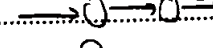


6. $R = R_1^*$

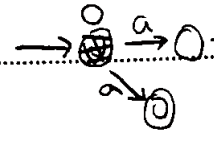


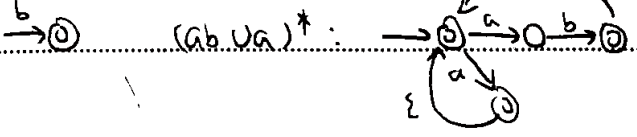
□

Example. $(ab \cup a)^*$

ab: 

a: 

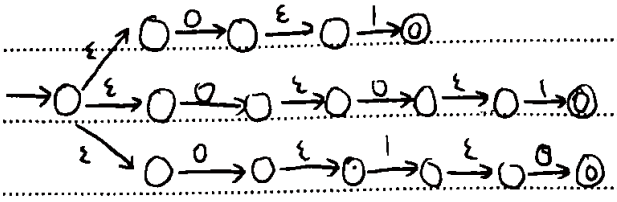
$ab \cup a$: 

$(ab \cup a)^*$: 

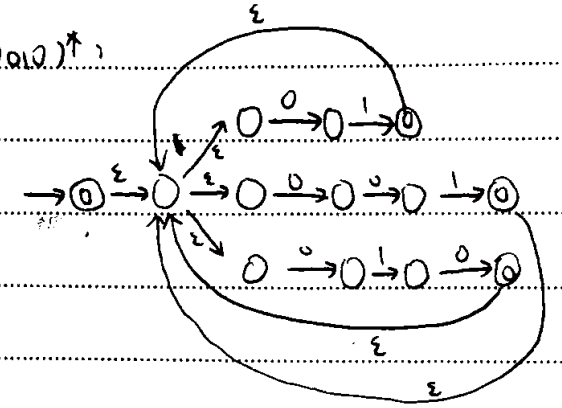
Exercise: 1. $(01 \cup 001 \cup 010)^*$ 2. $((00)^*1) \cup 10)^*$

1. 0:  1:

$01 \cup 001 \cup 010$:



$(01 \cup 001 \cup 010)^*$:



2.

Lemma 1.60 If a language is regular, then it is described by a regular expression.

DFA recog. $L \rightarrow$ regex describing L.

DFA → GNFA (generalized NFA) → regex

Def (GNFA): A generalized NFA is a 5-tuple $(Q, \Sigma, \delta, q_{start}, q_{accept})$.

$\delta: (Q - \{q_{accept}\}) \times R$ (all regex on Σ) $\rightarrow (Q - \{q_{start}\})$.

↳ Accept state has no out-edges.

Start state has no in-edges

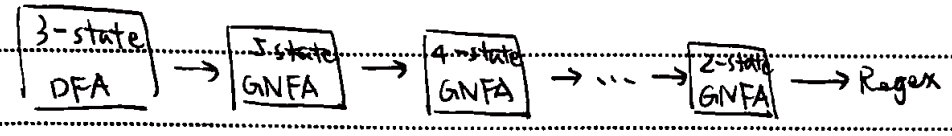
Label on transition arrows can be any regular expression

Require: 1. Start state has no incoming edges

2. only one accept state, and it has no outgoing edges.

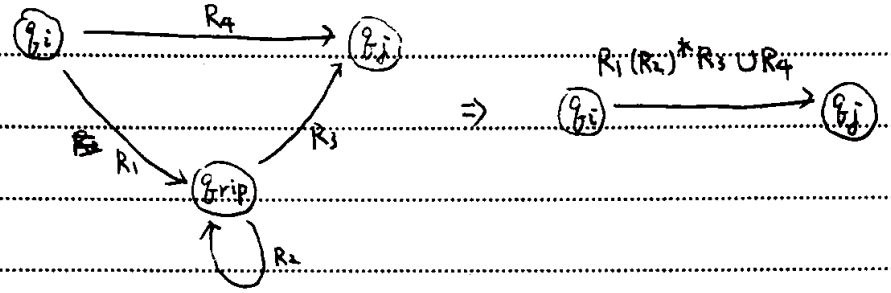
3. Except the starte state and accept state, there is an edge

between any two states, and also from a state to itself. (This can be done by adding $\emptyset$ arrows)



Let M be a DFA for language L. Convert M to a GNFA G_1.

Convert(G) (takes a GNFA as input, returns a regular expression)



Convert(G)

1. Let k be the number of states
2. If k=2, done.
3. If k>2, select $q_{rip} \in Q - \{q_{start}, q_{accept}\}$

   Let G' be the GNFA $(Q', \Sigma, \delta', q_{start}, q_{accept})$ where

   a) $Q' = Q - \{q_{rip}\}$

   b) For each $q_i \in Q' - \{q_{accept}\}$, $q_j \in Q' - \{q_{start}\}$, let

   $\delta'(q_i, q_j) = \delta(q_i, q_j) \cup \delta(q_i, q_{rip})\,\delta(q_{rip}, q_{rip})^*\,\delta(q_{rip}, q_j)$.
4. Return Convert(G')

Claim. For any GNFA, $L(G) = L(\text{Convert}(G))$. $G' = \text{Convert}(G)$.

Proof. 1) $L(G') \subseteq L(G)$

Suppose G' accepts w. Then $q_{start}, q_1, q_2, \ldots, q_{accept}$ $\Rightarrow$ ...

2) $L(G) \subseteq L(G')$

Non-regular languages

Given a computation model, we are interested in its computability. That is what is computable.

For FA, only regular languages are computable

By counting argument, most languages are uncomputable

How to prove a specific language is not regular?

Example. 1. $\{0^p \mid p \text{ is prime}\}$. 2. $\{w \mid w = w^R\}$. 3. $\{w \mid \#1's = \#0's\}$

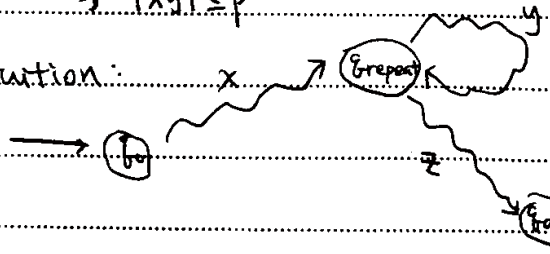
Pumping lemma. Let L be a regular language $\exists p \in \mathbb{N}$, s.t. for any string $s \in L$ of length $\geq p$, $\exists x, y, z \in \Sigma^*$, s.t. $s = xyz$, and

1) $xy^i z \in L$, $\forall i \geq 0$

2) $|y| > 0$

3) $|xy| \leq p$

Intuition:



Pigeon hole principle.

M has p states. String of length p goes through p+1 states in M during its computation.

$\Rightarrow xy^i z \in L$

Example. 1. $\{0^n 1^n \mid n \geq 0\}$ is not regular.

Let $p \in \mathbb{N}$ be the constant ~~that~~ in the Pumping lemma. Consider $w = 0^p 1^p$. By pumping lemma $w = xyz$ where $|xy| \leq p$. so, $y \in 0^+$.

So, $xy^2z \notin L$. L is not regular.

2. $\{w \mid w$ has ~~an~~ equal number of 0's and 1's $\}$

E.g. 1. Euclidean algorithm:

gcd(210, 45) = gcd(45, 30) = gcd(30, 15) = gcd(15, 0) = 15

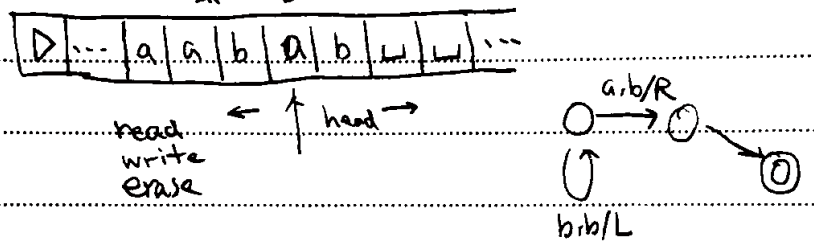
2. ____ sort

3. Dijkstra Alg for shortest path

...

algorithm: a mechanical process to be followed in calculations or other problem-solving operations.

Intuition:



Def (Turing machine) A Turing machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where 1. Q is the set of states

2. Σ is the input alphabet

3. Γ is the tape alphabet ($\Gamma = \Sigma \cup \{\sqcup, \triangleright\}$)

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

5. $q_0 \in Q$ is the start state

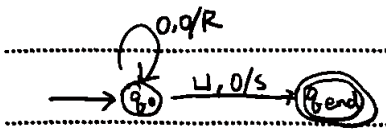
6. $q_{accept} \in Q$ is the accept state

7. $q_{reject} \in Q$ is the reject state

for computing (instead of deciding) problems.

Change to a single $q_{end} \in Q$.

Eg ① Add 1 (unary): input 0^n, output 0^{n+1}. 000␣␣... → 0000␣...



(2) Add 1 (binary) $i \to i+1$.

input: binary number n (least significant bit first)

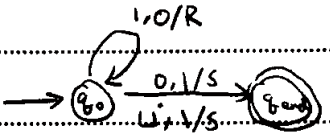
output: n+1.

$101\text{␣␣}\cdots \to 011\text{␣␣}\cdots$ $\quad \Sigma = \{0,1\}$, $\Gamma = \{\triangleright, ␣, 0, 1\}$

$011\text{␣␣}\cdots \to 111\text{␣␣}\cdots$

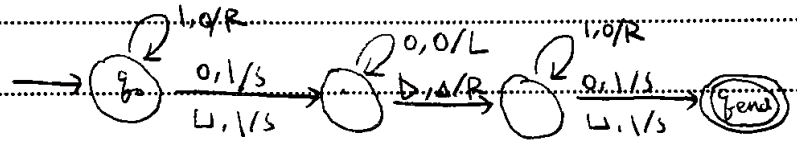
$111\text{␣␣}\cdots \to 000\text{␣␣}\cdots$

$110\text{␣}\cdots \to 001\text{␣}\cdots$



(3) Add 2 (binary)

→ Add 1 → Go back → Add 1



Or: Shift left by one bit and then add one.

(4) Palindrome $L = \{w \in \{0,1\}^* \mid w = w^R\}$

0 1 1 1 0 ␣ ⋯ idea: read the ~~left-most~~ leftmost symbol

1 1 1 0 ␣ ⋯ memorize and erase it;

1 1 1 # ␣ ⋯ go to the rightmost symbol;

reject if not match at any step.

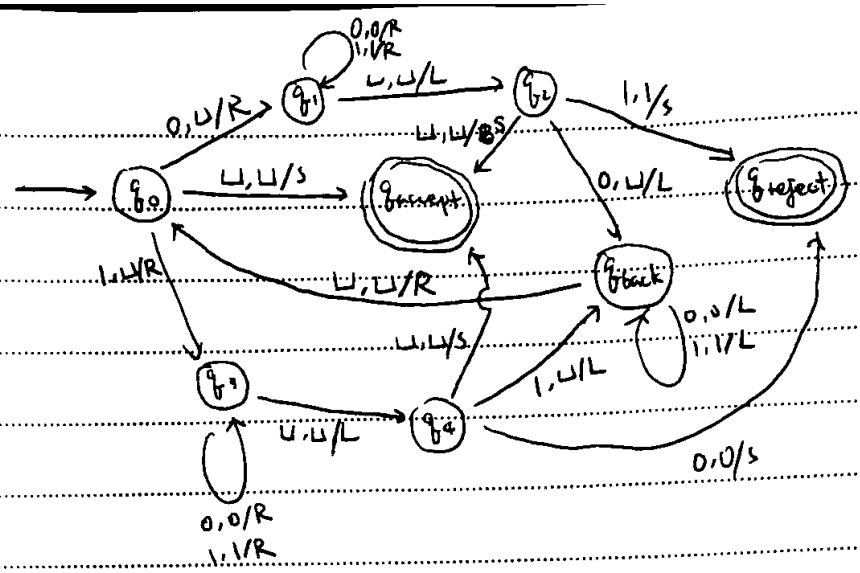
1 1 # ␣ ⋯ check if it matches the leftmost and erase it.

⋯⋯ repeat the above process

1 # # ␣ ⋯

(or # # # # ␣ ⋯ if length is even)

then accept.



(5) Accept language $L = \{0^{2^n} \mid n \geq 0\}$

idea: 1. Sweep left to right, crossing off every other 0

2. If the tape contains a single 0, accept

3. If the tape contains more than one 0's, and the number is odd, reject

4. go to step 1.

00 00 00 ␣ → 00 00 00 00 ␣

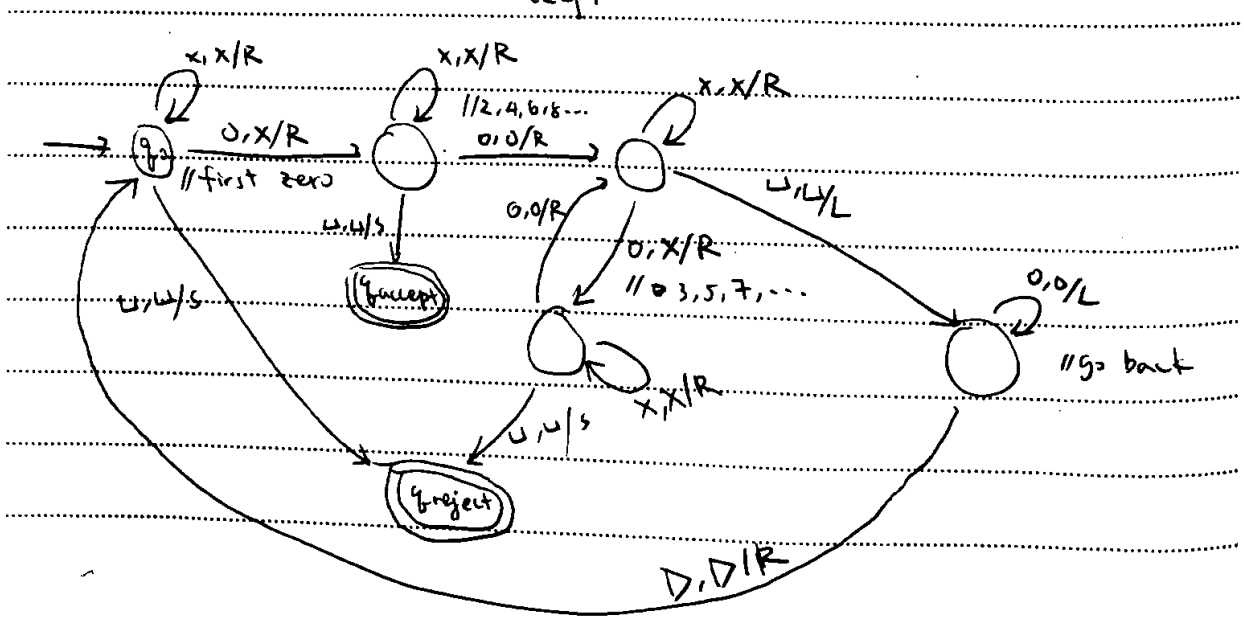
X0 X0 X0 ␣ → X 0 X0 X0 X0 ␣

reject. → X X X0 XX X0 ␣

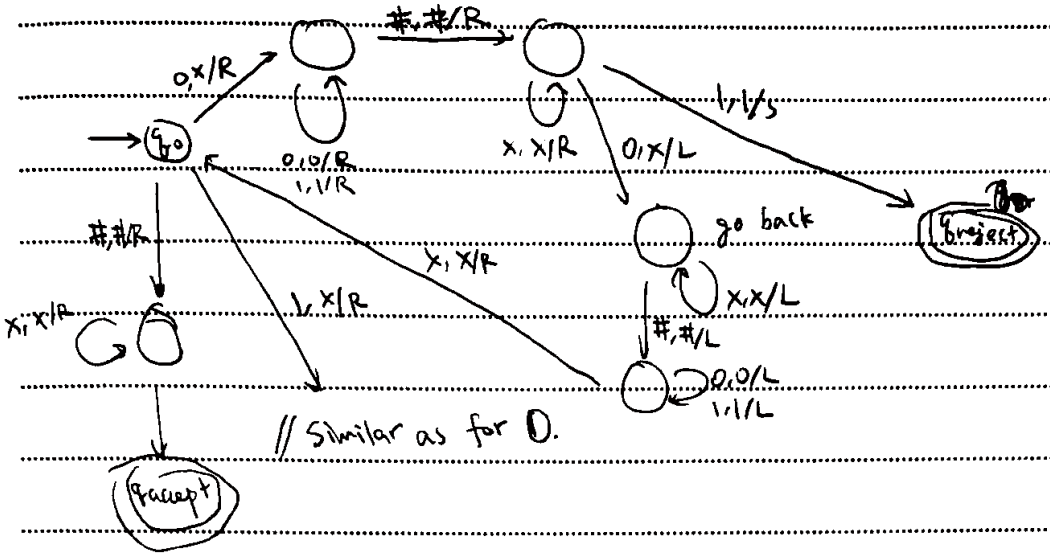
→ X X X X XX X0 ␣

$\Sigma = \{0\}$

$\Gamma = \{0, X, ␣, \triangleright\}$



(c) $L = \{w\#w \mid w \in \{0,1\}^*\}$



Exercise (1) Binary comparison

input: x, y : binary, significant bits first

output: $x \geq y$.

1) Compare length first $|x| > |y|$ accept

$\Gamma = \{0, 1, \hat{0}, \hat{1}\}$

2) Compare bit by bit

if $<$ reject, if $=$ continue, if $>$ accept

Def. Let $L \subseteq \{0,1\}^*$ be a language. Let M be a TM. We say that M decides L in time $T(n)$ if for every $x \in \{0,1\}^*$

1) M always halts in $T(|x|)$ steps

2) If $x \in L$, then M accepts x

3) If $x \notin L$, then M rejects x

Call a language (Turing) decidable if there is a TM that decides it.

Turing recognizable: Change 3) to "M rejects x or loops".

The set of strings that M accepts is the language recognized by M, denoted by $L(M)$.

$L(M) = \{x \in \Sigma^* \mid M \text{ accepts } x\}$.

Call a language Turing recognizable if some TM recognizes it

Obviously, every decidable language is recognizable. The converse is false

Let $f: \{0,1\}^* \to \{0,1\}^* \cup \{\text{undefined}\}$

Def Say TM M computes f, if for every $x \in \{0,1\}^*$ with $f(x) \neq$ undefined, M halts in $T(n)$ steps with $f(x)$ on its tape.

What is an algorithm? An algorithm = a TM.

Despite the model's simplicity, it is capable of implementing any algo

Variants of TMs

The original model and its reasonable variants all have the same power, ie. they decides / recognize the same set of languages

Change alphabet

Lemma If language L is decidable in time $T(n)$ by a TM on tape alphabet Γ, then it is decidable on alphabet $\{0,1\}$ in time $O(T(n) \log|\Gamma|)$.

Proof - Encode any symbol in Γ using $k = \lceil \log|\Gamma| \rceil$ bits. To simulate one step of M, the new TM M will:

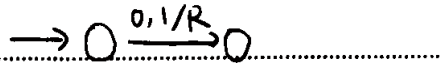
1) Use k steps to read a symbol a

2) According to δ, get the new symbol b

3) Overwrite a by b.

→ 当右下重复了.

Lemma. If $L \subseteq \{0,1\}^*$ is decidable in time $T(n)$ by a TM on alphabet Γ, then it is decidable in time $O(\log|\Gamma| \cdot T(n))$ by a TM on alphabet $\{0,1,\triangleright,\sqcup\}$



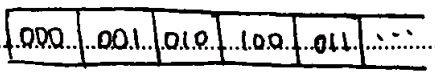
0 000

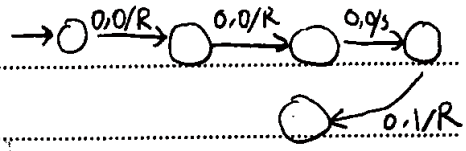
b 011

1 001

100

a 010





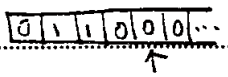
Proof. Encode any symbol in Γ using $k = \lceil \log_2 |\Gamma| \rceil$ bits. To simulate one step of M, TM M' will

1) Use k steps to read a symbol
2) Transit to the next step
3) Use k steps to overwrite the symbol if needed
4) Move the head properly (k steps to the next symbol in M.)

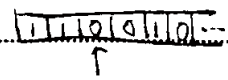
$\leq k+1+k+k = O(k)$

In total, the simulation takes at most $O(kT(n))$ step. □

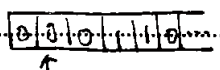
Multitape TM. Def. A k-tape TM is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

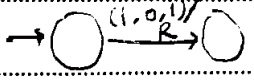


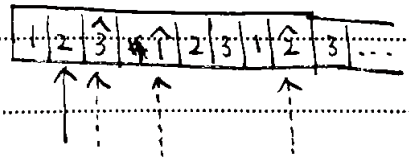
$\delta: Q \times \Sigma^k \to Q \times \Sigma^k \times \{L, R, S\}^k$



Lemma Let $L \subseteq \Sigma^*$. If L is decidable by a k-tape TM in time $T(n)$, then L is decidable in time $O(kT(n)^2)$ by a single-tape TM.







Proof. Use locations $i-1, k+i-1, 2k+i-1, \ldots$ to simulate the ~~ith~~ i^{th} tape where $i=1,2,\ldots$. For every symbol $a \in \Gamma$, ~~in t~~ introduce a new symbol $\hat{a} \in \Gamma'$ to denote where the head is.

To simulate one step we need

1) sweep the tape to read the symbols with $\wedge$
2) transit state
3) Sweep the tape to update the symbols, and move $\wedge$ if needed.

Time # steps $T(n) = \max_{x \in \{0,1\}^n} \{\#\text{execution steps on input } x\}$

Space # cells wed $S(n) = \max_{x \in \{0,1\}^n} \{\#\text{ cells wed on input } x\}$

$S(n) \leq kT(n)$ on k-tape TM (move one cell for one step)

$\Rightarrow$ 1): $O(kT(n))$

2): 1 for ~~one~~ simulation of one step.

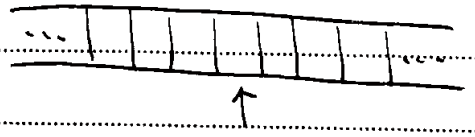
3): $O(kT(n))$

$\Rightarrow$ In total, the running time is $O(kT(n)^2)$

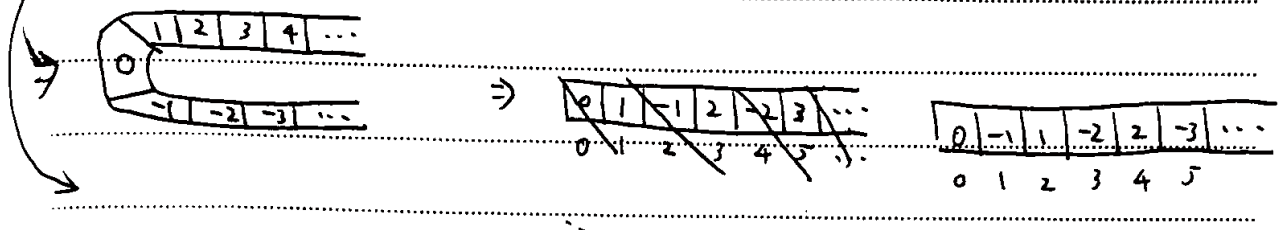
$T(n) \to T(n)^{O(1)}$

Bidirectional TM

Bidirectional TM is a TM where where tape is infinite in both directions



Lemma. $L \subset \Sigma^*$. If L is decidable by a bidirectional TM in time $T(n)$, then L is decidable ~~by $O(T(n))$~~ by a TM in time $O(T(n))$.



$i \geq 0$

Proof. Map position i to $2i$, position $-i$ to $2i-1$, $i \geq 1$

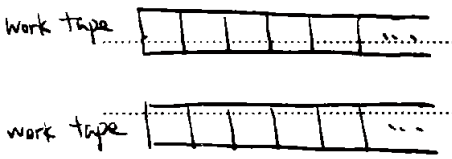
For each step of M, M'

1) reads the symbol
2) transits to the next state
3) Overwrite the symbol if needed
4) Move left or right in two steps

} ≤ 2 steps

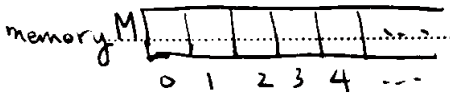
Random access memory TM

~~RAM Turing machine is~~ 1. M has an infinite memory tape ~~and~~ indexed by $\mathbb{N}$

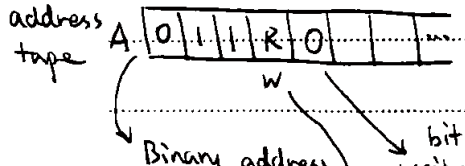


2. M has an address tape A

3. Γ contains two special symbols R and W



4. Q has special states denoted by $Q_{access} \subset Q$.



Whenever M gets into a state $q \in Q_{access}$

1) If the address tape contains $[i]R$ ~~then~~ the value $M[A[i]]$ is written to the cell next to R

2) If the address tape contains $[i]W\sigma$, then $M[A[i]] \leftarrow \sigma$

δ $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}, Q_{access} \subset Q)$

$\delta: Q \times \Gamma^{k+1} \to Q \times \Gamma^{k+1} \times \{L, R, S\}^{k+1}$

work and address tapes

Lemma. If $L \subset \Sigma^*$ is decidable by a ~~TM~~ RAM TM in $T(n)$, then it is decidable by a TM in time $O(T(n)^3)$

Proof. Use one extra tape M for memory

It contains pairs like $(i, M[i])$ that have been read from or written to.

(访问过的地址自然不超过 $T(n)$ 个) ← 假如 RAM TM 不停在 A上读 1. 则为 2^n 地址，非法机的位置访问会变成 $O(2^n)$，不满多项式的。

To simulate one step of M, M' will

1. scans through M to find the address that matches i in the address tape
2. read or write M[i] accordingly
3. If such i does not exist, create a new entry $(i, M[i])$

On ~~#~~ tape M: # pairs = $O(T(n))$, and for each pair $(i, M[i])$, its length is ~~$O(n)$~~ $O(T(n))$ (The RAM TM is $T(n)$).

⇒ 1, 2, 3 takes $O(T(n)^2)$ time

⇒ M' run in $O(T(n)^3)$ time in total

RAM TM → multitape TM → single-tape TM

$O(T(n))$ → $O(T(n)^3)$ → $O(T(n)^6)$

↑

Assembly program ← C++

$O(T(n))$ $O(T(n))$

ASM: a constant number of registers (64 bits)

a program counter

addition, subtraction, multiplication

ADD R0, R1, R2 // R0 ← R1 + R2
SUB R0, R1, R2 // R0 ← R1 − R2
MUL R0, R1, R2 // R0 ← R1 * R2
} can be simulated in O(1) time. One tape for one register.

Compare (write result to the status register)

CMP R0, R1

Move

MOV R0, R1 // R0 ← R1

Bitwise OR, NOR, AND, ...

ORR R0, R1, R2 // R0 ← R1 | R2

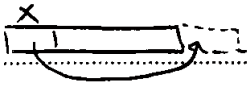
Test equal

TST R0, #8 // R0 = 8?

Logical shift

LSL R0, R1, #3 // R0 ← R1 << 3

R2 R2.

Rotate 

Branch

Loop: transition between states,

// some instr

B Loop

Load and Store: O(1) in RAM TM.

LDR R0, [R1] // R0 ← M[R1]

STR →

⇒ All asm instr can be simulated by a RAM TM in O(1).

O(T(n)) → O(T(n))

When referring to TMs in texts:

algorithm: RAM TM

complexity: multitape TM

Ignoring polynomial factors, all variants are equivalent.

Church-Turing Thesis:

"Every function that can be physically computed can be computed by a Turing machine"

Strong C-T Thesis (suspected):

"... with polynomial overhead"

Def. Let $T: \mathbb{N} \to \mathbb{N}$ be some function. Language $L \subseteq \Sigma^*$ is in DTIME($T(n)$) iff. there exists a Turing machine M that decides L in time $O(T(n))$.

Def. $P = DTIME(n) \cup DTIME(n^2) \cup \cdots$

Encoding of a TM.

1) Every string $\alpha \in \{0,1\}^*$ represents some TM, denoted by M_α (on invalid encoding α, M_α always reject)

2) Every TM is represented by infinitely many strings in $\{0,1\}^*$

E.g. $M=(Q,\Sigma,\Gamma,\delta,q_a,$ ~~q_{reject}~~ $q_{accept}, q_{reject})$

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

~~the~~ 可表示为有限个五元组 (q_i, a, q_j, b, D).

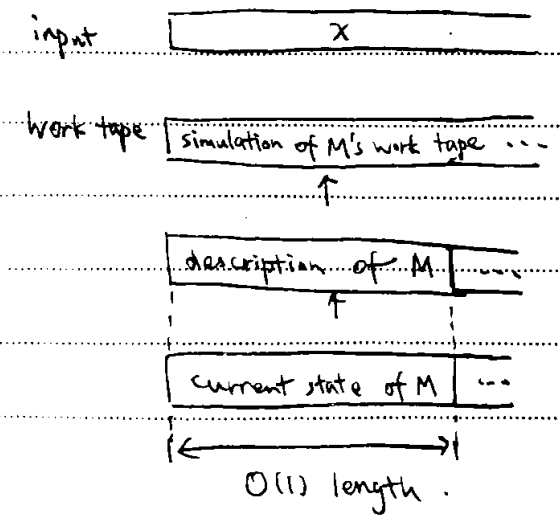
最后编码为二进制串

一个 TM 的编码无穷多：规定可在编码后加任意多空格 $\sqcup$

Theorem (universal TM) There exists a multitape TM U s.t. for all $x,\alpha\in\{0,1\}^*$, $U(x,\alpha)=M_\alpha(x)$. Moreover, if M_α halts on input x within T steps, then $U(x,\alpha)$ halts in $O_M(T\log T)$ steps.

Proof of a ~~stee~~ weaker version. $O(T^2)$

We have shown that every multitape TM with running time $O(T(n))$ can be transformed to a single TM with running time $O(T(n)^2)$



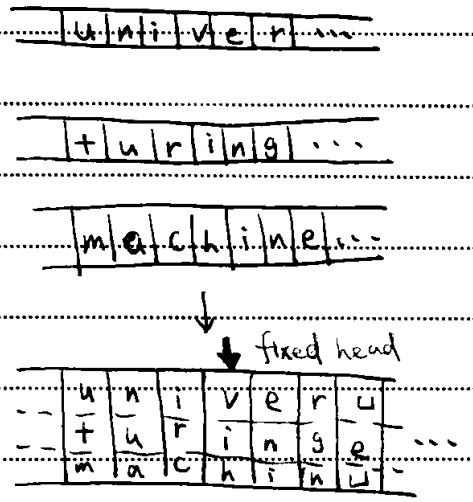
To simulate one step of M, the UTM U

1. reads the symbol on the simulation work tape
2. sweeps the description tape of M to get the transition rule $(q_{current}, a, q_{new}, b, D)$ (O(1))
3. overwrite ~~current~~ a with b if $a\neq b$
4. move the head on the simulation tape if needed
5. transit to the new state q_{new}.

The above can be done in $O(1)$ steps. So the total running time is $O(T(n)^2)$.

Proof. Let k be the number of work tapes, and Γ its alphabet. Assume k was alphabet Γ^k

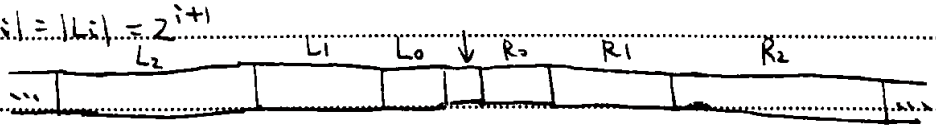
U moves the tape, instead of moving the heads. However, the simulation takes $O(T(n)^2)$ (i, r, b)



解决方法：引入空字符作间隔 ☒.

如用同 c++ 的 vector<T>. 使得字符移动更快.

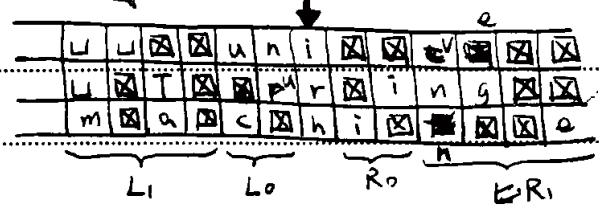
Split U's parallel tape into zone, denoted by $R_0, L_0, R_1, L_1, \ldots, R_{\lceil \log T \rceil}, L_{\lceil \log T \rceil}$ (The central cell is not in any zone), where $|R_i| = |L_i| = 2^{i+1}$



Maintain the following invariants:

1) Each zone is empty, full or half full. # That is, # ☒ in L_i (or R_i) is $0, 2^i$ or 2^{i+1}

2) # ☒ cells in $L_i \cup R_i$ is 2^{i+1}.



Perform a shift (left shift for example)

1) Find the smallest i_0 s.t. R_{i_0} is not empty

2) Put the leftmost non-$\boxtimes$ symbol of R_{i_0} in the center, and shifts the remaining $2^{i_0}-1$ non-$\boxtimes$ from R_{i_0} to $R_0, R_1, \ldots, R_{i_0-1}$, filling exactly half.

$$|R_0| + |R_1| + \cdots + |R_{i_0-1}| = 2^1 + 2^2 + \cdots + 2^{i_0-1} = 2(2^{i_0}-1)$$

3) U performs the symmetric operation to the left

R_{i_0} is not empty $\Rightarrow$ L_{i_0} is not full

Move the central cell and exactly half of non-$\boxtimes$ cells in $L_0, L_1, \ldots, L_{i_0-1}$ to L_{i_0}.

before		after	
$R_0, \ldots, R_{i_0-1}$ empty		$R_0, \ldots, R_{i_0-1}$ half	
$L_0, \ldots, L_{i_0-1}$ full		$L_0, \ldots, L_{i_0-1}$ half	
L_{i_0} half, R_{i_0} half	or	L_{i_0} empty, R_{i_0} full	
L_{i_0} full, R_{i_0} empty	or	L_{i_0} half, R_{i_0} full	

Once we perform a shift with index i, the next 2^i-1 shift will have index less than i. This implies that at most $1/2^i$ fraction of shifts has index i.

The highest possible index is i_{max}

$$1 + \sum_{i=0}^{i_{max}} |R_i \cup L_i| \geq 2T(n)$$

$$1 + \sum_{i=0}^{i_{max}} 2^{i+2} = 1 + 4(2^{i_{max}+1} - 1) \geq 2T(n)$$

$$2^{i_{max}+1} \geq \frac{2T(n)+3}{4}$$

$$i_{max} \geq \log(2T(n)+3) - 3$$

$$i_{max} = \lceil \log T(n) \rceil + 1.$$

$$\sum_{i=0}^{i_{max}} \frac{T}{2^i} O(2^i) = \sum_{i=0}^{i_{max}} O(T) = O(T \log T)$$

Lemma. Almost all languages are undecidable.

Proof. #TMs ≤ # strings over $\{0,1\} = \aleph_0$

languages $= 2^{\aleph_0} = \aleph_1$

Let $L_{flip} = \{\alpha \in \{0,1\}^* \mid M_\alpha$ does not accept α, i.e. M_α rejects α or loops forever$\}$

diagonalization

Lemma. L_{flip} is undecidable.

Proof. Assume for contradiction that $\exists$ TM M_β that decides L_{flip}.

Case 1: $\beta \in L_{flip}$. So, M_β does not accept β by the definition of L_{flip}. However, M_β decides L_{flip}, which implies that M_β accepts β. Contradiction.

Case 2: $\beta \notin L_{flip}$. By the definition of L_{flip}, M_β accepts β. Since M_β decides L_{flip}, and $\beta \in L_{flip}$, M_β rejects β. Contradiction.

Turing halting problem:

$L_{halt} = \{(\alpha, x) \mid M_\alpha \text{ halts on input } x\}$

Theorem L_{halt} is undecidable

Proof. Assume for contradiction that L_{halt} is decidable, i.e. $\exists$ TM M_{halt} that decides L_{halt}

Construct a TM M on input α as follows

1) Use M_{halt} to decide if M_α halts on α. If not, ~~reject~~ accept.

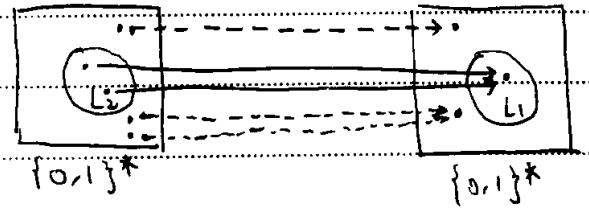
2) If halts, simulate M_α on α, flip the output.

We claim M decides L_{flip}. Contradiction. □

Turing reducible

Def. Let $L_1, L_2 \subseteq \{0,1\}^*$. Say L_2 is Turing reducible to L_1, denoted $L_2 \leq_C L_1$, if $\exists$ Turing computable function $\varphi: \{0,1\}^* \to \{0,1\}^*$ s.t.

1) $\forall x \in L_2$, $\varphi(x) \in L_1$, 2) $\forall x \notin L_2$, $\varphi(x) \notin L_1$.



φ不需要为单射.

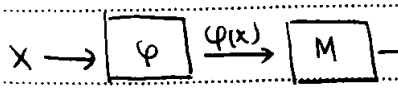
$L_2 \leq L_1$.

Lemma: If L_1 is decidable, then L_2 is decidable.

Proof. Suppose TM M_1 decides L_1. Construct L_2 as follows

1) on input x, compute $\varphi(x)$.

2) run M_1 $\varphi(x)$. return the result.



Corollary. Let $L_2 \leq L_1$. If L_2 is undecidable, then L_1 is undecidable.

Theorem. Let $L_{accept} = \{(M, x) \mid$ TM M accepts input $x\}$. L_{accept} is undecidable.

Proof. Assume for contradiction that L_{accept} is decidable, ie. $\exists$ TM M_{accept} that decides L_{accept}.

Construct TM M to decide L_{halt} as follows: on input (α, x)

1) Construct a new TM $M_{\alpha'}$, which simulates M_α and flips the output. ($M_{\alpha'}$ 跟 M_α 結果反轉 accept/reject. M_α 不停就 loop 則 $M_{\alpha'}$ 也不停就 loop).

2) Use M_{accept} to decide if $(M_\alpha, x) \in L_{accept}$ or $(M_{\alpha'}, x) \in L_{accept}$.

3) If one of them is true, then accept (α, x); otherwise, reject.

We claim M decides L_{halt}. Contradiction. □

Theorem. Let $L_{empty} = \{\langle M \rangle \mid M$ is a TM s.t. $L(M) = \emptyset\}$. L_{empty} is undecidable.

Proof. Assume for contradiction that L_{empty} is decidable, i.e. $\exists$ TM M_{empty} that decides L_{empty}.

Construct TM M_{halt} on input (α, x) as follows

1) Construct a new TM M_β where $\beta = \beta(\alpha, x)$, which, on input y

a) simulates M_α on x

b) if it halts, always accept input y.

2) Use M_{empty} on input β to decide if $L(M_\beta) = \emptyset$.

We claim that M_{halt} decides L_{halt}. Contradiction. ⊟

Case 1: M_α halts on x. So $L(M_\beta) = \{1, 0\}^*$

Case 2: M_α does not halt on x. So $L(M_\beta) = \emptyset$. □

Theorem $L_{regular} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$

$L_{regular}$ is undecidable.

Proof: Assume for contradiction that $L_{regular}$ is decidable, i.e. $\exists$ TM $M_{regular}$ that decides $L_{regular}$

Obs: $\{0^n 1^n \mid n \geq 1\}$ is not regular. On input $\langle \alpha, x \rangle$, construct a TM M_{accept} as follows

1) Construct TM M_β, where $\beta = \beta(\alpha, x)$, as follows:

   a) on input y, accept y if $y = 0^n 1^n$ for some $n \geq 1$

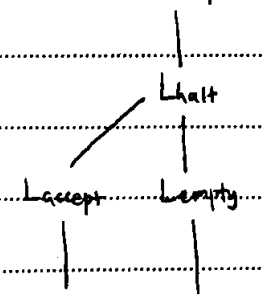
   b) if y does not have this form, run M_α on x, and accept y if M_α accepts x

2) Run $M_{regular}$ on input $\langle \beta \rangle$

3) If $M_{regular}$ accepts β, accept $\langle \alpha, x \rangle$. Otherwise, reject.

Verify: Case 1: $\langle \alpha, x \rangle \in L_{accept}$. So, $L(M_\beta) = \{0,1\}^*$ is regular. So $M_{regular}$ accepts β. Thus M_{accept} accepts $\langle \alpha, x \rangle$.

Case 2: $\langle \alpha, x \rangle \notin L_{accept}$. So $L(M_\beta) = \{0^n 1^n \mid n \geq 1\}$, is not regular. So, $M_{regular}$ rejects β, and M_{accept} rejects $\langle \alpha, x \rangle$. □



Theorem Let $L_{equal} = \{\langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$

L_{equal} is undecidable.

Proof: Assume for contradiction that L_{equal} is decidable, i.e. $\exists$ TM M_{equal} that decides L_{equal}. We will prove L_{empty} is decidable:

On input $\langle M \rangle$, construct a TM M_{empty} to decide L_{empty}:

1) Run M_{equal} on input $\langle M, M_1 \rangle$ where M_1 rejects all input.

2) If M_{equal} accepts, accept. Otherwise, reject. □

「任何关于图灵机所识别的语言的性质之非平凡者均不可判定」

Property $\mathcal{P}$ is about the language recognized by TMs if whenever $L(M) = L(N)$ then $\mathcal{P}$ contains $\langle M \rangle$ iff. $\mathcal{P}$ contains $\langle N \rangle$.
↳ 对不接受的它输入，M,N功能不一定一样（拒绝/循环）

Rice's Theorem (not empty or complete) Any nontrivial property about the language recognized by Turing machines is undecidable.

nontrivial: $\exists$ TM M s.t. $\langle M \rangle \in \mathcal{P}$ and $\exists$ TM N s.t. $\langle N \rangle \notin \mathcal{P}$

Proof. WLOG (without out of generality), assume not the empty (always reject) TM is in $\mathcal{P}$ (Otherwise, take the complement of $\mathcal{P}$). Assume $\langle M_\beta \rangle \in \mathcal{P}$

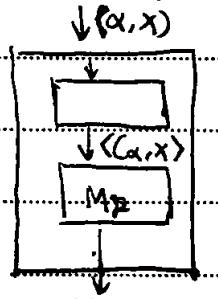
Assume for contradiction that $\mathcal{P}$ is decidable by a TM $M_{\mathcal{P}}$

1. On input (α, x), create a TM $C_{\alpha,x}$ as follows

a) On input y, let M_α run on x until it accepts. Goal: design a
(If it does not accept, $C_{\alpha,x}$ will run forever) TM to decide

b) Run M_β on y, accept if M_β accepts y. L_{accept} using $M_{\mathcal{P}}$

2. Feed $\langle C_{\alpha,x} \rangle$ to $M_{\mathcal{P}}$. Accept if $M_{\mathcal{P}}$ accepts.
Otherwise, reject.



We claim that the above TM decides L_{accept}

1. If M_α accept x, then $L(C_{\alpha,x}) = L(M_\beta)$. Since $\langle M_\beta \rangle \in \mathcal{P}$, so is $\langle C_{\alpha,x} \rangle$

2. If M_α does not accept x, then $L(C_{\alpha,x}) = \emptyset$ (无循环). Since $\langle M_\emptyset \rangle \notin \mathcal{P}$, so is $\langle C_{\alpha,x} \rangle$. Thus $M_{\mathcal{P}}$ will reject $\langle C_{\alpha,x} \rangle$ □

Post Correspondence Problem (PCP)

Domino: $\left[\frac{a}{ab}\right]$, ...

A collection of dominos:

$$\left\{\left[\frac{b}{ca}\right], \left[\frac{a}{ab}\right], \left[\frac{ca}{a}\right], \left[\frac{abc}{c}\right]\right\}$$

E.g. $\left[\frac{a}{ab}\right]\left[\frac{b}{ca}\right]\left[\frac{ca}{a}\right]\left[\frac{a}{ab}\right]\left[\frac{abc}{c}\right]$ (每张牌可用无数次)

a	b	c	a	a	a	b	c
a	b	c	a	a	b/a	b	c

The PCP P is to determine whether a collection of dominos has a match.

Formally, an instance of the PCP is a collection of dominos

$$P = \left\{\left[\frac{t_1}{b_1}\right], \left[\frac{t_2}{b_2}\right], \ldots, \left[\frac{t_k}{b_k}\right]\right\}$$

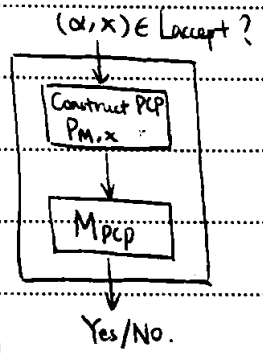
A match is a sequence $i_1, i_2, \ldots, i_l$ where $t_{i_1} t_{i_2} \cdots t_{i_l} = b_{i_1} b_{i_2} \cdots b_{i_l}$

Let $PCP = \{\langle P\rangle \mid P \text{ is an instance of PCP that has a match}\}$

Theorem 5.15 PCP is undecidable.

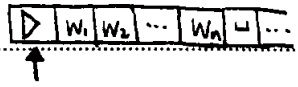
Proof idea: Reduce L_{accept} to PCP. algorithmically

Given any TM M and input x, we construct a PCP instance $P_{M,x}$ such that M accepts x iff $P_{M,x} \in PCP$.



Handle 3 technical points:

1) M on w never attempts to move its head off the left-hand end.



2) If $w = \varepsilon$, use the string ␣.

3) A match must start with $\left[\frac{t_1}{b_1}\right]$ (此假设后续可消除)

Call it modified PCP (MPCP).

Proof. Let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ Construct an instance of

MPCP P' s.t. P' has a match iff. M accepts w

Part 1. Put $\left[\frac{\#}{\#q_0w_1w_2\cdots w_n\#}\right]$ as the first domino　#用于分隔两个TM在运行过程中的两个Configuration「截图」

Part 2. If $\delta(q,a)=(r,b,R)$, put $\left[\frac{qa}{br}\right]$ into P'　描述右移

Part 3. If $\delta(q,a)=(r,b,L)$, put $\left[\frac{cqa}{rcb}\right]$ into P'　描述向左移，c是「a左侧的之字符」

for every $c\in\Gamma$.

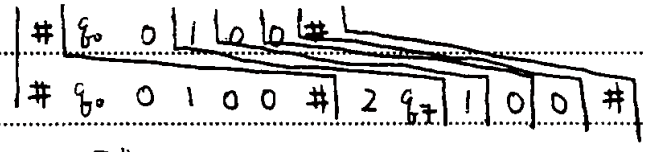
Part 4. For every $a\in\Gamma$, put $\left[\frac{a}{a}\right]$ into P'.　对齐

Let $\Gamma=\{0,1,2,\sqcup\}$, $w=0100$, start state is q_0, $\delta(q_0,0)=(q_7,2,R)$

Part 1 : $\left[\frac{\#}{\#q_0 0100\#}\right]$

Part 2 : $\left[\frac{q_0 0}{2 q_7}\right]$

Part 4 : $\left[\frac{0}{0}\right],\left[\frac{1}{1}\right],\left[\frac{2}{2}\right],\left[\frac{\sqcup}{\sqcup}\right]$

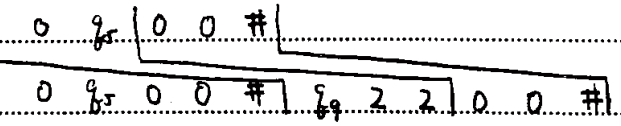


Part 5: Put $\left[\frac{\#}{\sqcup\#}\right]$ and $\left[\frac{\#}{\#}\right]$ into P'

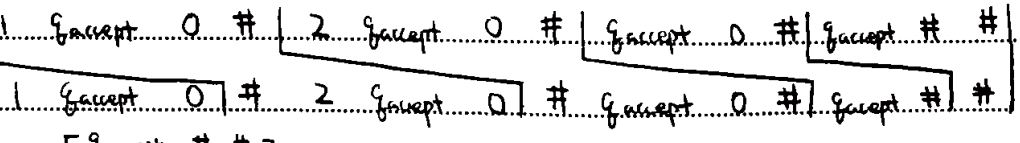
↳ simulate infinitely many blanks.

Suppose $\delta(q_5,0)=(q_9,2,R)$

$\left[\frac{0\,q_5\,0}{q_9\,0\,2}\right],\left[\frac{1\,q_5\,0}{q_9\,1\,2}\right],\left[\frac{2\,q_5\,0}{q_9\,2\,2}\right],\left[\frac{\sqcup\,q_5\,0}{q_9\,\sqcup\,2}\right]$



Part 6. For every $a\in\Gamma$, put $\left[\frac{a\,q_{accept}}{q_{accept}}\right]$ and $\left[\frac{q_{accept}\,a}{q_{accept}}\right]$ into P'



Part 7. Add $\left[\frac{q_{accept}\,\#\,\#}{\#}\right]$

Finally, we convert MPCP instance P' to a PCP instance P. Let

$u = u_1 u_2 \cdots u_n$.

Define $\bigstar u = *u_1 * u_2 * \cdots * u_n$

$u \bigstar = u_1 * u_2 * \cdots * u_n *$

$\bigstar u \bigstar = * u_1 * u_2 * \cdots * u_n *$

If P' were $\{[\frac{t_1}{b_1}], [\frac{t_2}{b_2}], \ldots, [\frac{t_n}{b_n}]\}$, Let P be

$\{[\frac{\bigstar t_1}{b_1 \bigstar b_2 \bigstar}], [\frac{\bigstar t_1}{b_1 \bigstar}], [\frac{\bigstar t_2}{b_2 \bigstar}], \ldots, [\frac{\bigstar t_k}{b_k \bigstar}], [\frac{\bigstar \diamond}{\diamond}]\}$

↓ 使 P 不得不以此为开头 □

Peano arithmetic

1) Constant 0

2) Successor operator S

$S(0) = 1$, $S(S(0)) = 2$, ...

3) addition, multiplication

4) logical conjunction $\wedge$, $\vee$, $\neg$

5) quantifier $\forall$, $\exists$

6) binary relation $<$, $=$

7) parentheses

8) variables $x, x^*, x^{**}, \ldots$

DIVIDE (x,y) : $\Leftrightarrow (\exists k)(y = kx)$ $(x|y)$

PRIME (y): $y \neq 1 \wedge (x = 1 \vee x = y \vee \neg \text{DIVIDE}(x,y))$

Goldbach ~~conjecture~~: $(\forall x \geq 2)(\exists y)(\exists z)(2x = y + z \wedge \text{PRIME}(\frac{y}{\bigstar}) \wedge \text{PRIME}(z))$

Godel incompleteness theorem: ←

Theorem. Peano arithmetic includes undecidable propositions.

All consistent axiomatic formulation of number theory include Peano arithmetic has undecidable propositions

~~Proof idea For every $\alpha, x \in \{0,1\}^*$, construct a formula $\varphi_{\alpha,x}$ s.t. $\varphi_{\alpha,x}$ is true iff. M_α halts on x.~~

Examples

1) $(\forall q)(\exists p)(\forall x,y)(p>q \wedge (xy>1 \Rightarrow xy \neq p))$

There are infinitely many primes

2) $(\forall a,b,c,n)(a,b,c>0 \wedge n>2 \Rightarrow a^n+b^n \neq c^n)$

Fermat's Last Theorem

3) $(\forall q)(\exists p)(\forall x,y)(p>q \wedge (x,y>1 \Rightarrow (xy \neq p \wedge xy \neq p+2)))$

Twin Prime Conjecture

Proof idea For every $\alpha, x \in \{0,1\}^*$, construct a formula $\varphi_{\alpha,x}$ s.t. $\varphi_{\alpha,x}$ is true iff. M_α halts on x.

If F is complete, we use a TM to enumerate all proofs up to length k, and verify if it is a correct proof of $\varphi_{\alpha,x}$ or $\neg\varphi_{\alpha,x}$. In this way, L_{halt} is decidable. Contradiction. □

Hilbert 10th problem:

Given a Diophatine equation with any number of unknown quantities and integral coefficients, design an algorithm that decides whether the equation is sovable in integers.

E.g. $3x^2-2xy-y^2z-7=0$, $x=1, y=2, z=-2$.

$x^2+y^2+1=0$ has no integral solution

Hilbert 10th problem is undecidable

Def. Let $T: \mathbb{N} \to \mathbb{N}$ Language $L \subseteq \{0,1\}^*$ is in $DTIME(T(n))$ iff. there exists a TM M that runs in time $O(T(n))$ and decides L

$$P = \bigcup_{c \geq 1} DTIME(n^c) = DTIME(n^{O(1)})$$

$$EXP = \bigcup_{c \geq 1} DTIME(2^{n^c}) = DTIME(2^{n^{O(1)}})$$

Remark. 1) The computation model does not matter, as they are polynomially equivalent.

2) P roughly corresponds to the class of problems ~~solvable~~ that are realistically solvable.

3) Although ignoring the actual exponent is absurd, the exponent for practical algorithms is often small

4) In contrast, brute-force search often takes exponential time

5) Polynomials satisfy closure property: if $f(n), g(n)$ are polynomials, then so is $f(n)+g(n)$, $f(n) \cdot g(n)$, $f(g(n))$

In TCS literature, efficient = polynomial time

~~Theorem 7.14~~

PRIME = {x | x is a prime number in binary.} PRIME $\in$ EXP.

PRIME(x) input length: $l = \lceil \log_2 x \rceil$

$y \leftarrow 2$

while $y \leq \sqrt{x}$ # iterations $x-2 \leq 2^l$ $(\sqrt{x} \leq 2^{l/2})$ → 也是指数的

if $x \bmod y = 0$ time in each iteration $O(l^2)$ (除法是 l^2 的，类似竖式除法).

reject

$y \leftarrow y+1$ in total. $\leq 2^l \times O(l^2) = 2^{O(l)}$

accept

So. PRIME $\in$ EXP [AKS04]: PRIME $\in$ P.

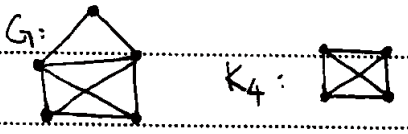
The class NP. (Nondeterministic Polynomial)

NP is the set of languages that can be verified in polynomial time.

Def. Language $L \in \{0,1\}^*$ is in NP if there exist a polynomial $p: \mathbb{N} \to \mathbb{N}$ and a ~~pf~~ polynomial TM M (called the verifier) s.t. for every $x \in \{0,1\}^*$, $x \in L$ iff $\exists\, w \in \{0,1\}^{p(|x|)}$ s.t. $M(x,w)=1$.

Such w is called a certificate for x (with respect to L and M)

1) k-CLIQUE = $\{\langle G, k\rangle \mid$ G contains a K_k subgraph$\}$ $\in$ NP



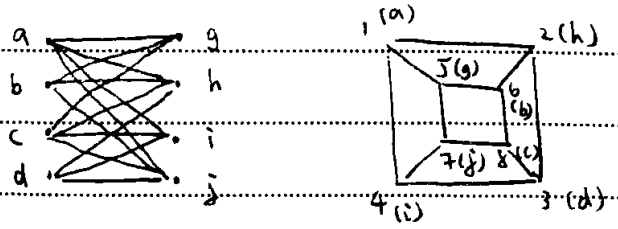
Certificate: a subset of V(G) of size k

verifier M:

(1) Check the certificate is a list of k vertices

(2) Check any two vertices in the list has an edge

2) GRAPHISO = $\{\langle G, H\rangle \mid$ undirected graphs G and H are isomorphic$\} \in$ NP



Certificate : $f: V(G) \to V(H)$.

Verifier M: check if f is an isomorphism.

Def (NP). Language $L \subseteq \{0,1\}^*$ is in NP if $\exists$ polynomial $p: \mathbb{N} \to \mathbb{N}$ and a poly-time TM M s.t.

1) $(\forall x \in L)(\exists w \in \{0,1\}^{p(|x|)})(M(x,w)=1)$

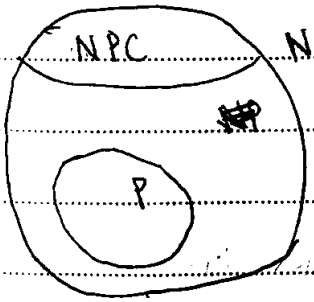
2) $(\forall x \notin L)(\forall w \in \{0,1\}^{p(|x|)})(M(x,w) \neq 1)$

w: certificate, M: verifier

Traveling salesman: Given a set of n nodes, $\binom{n}{2}$ number d_{ij} decides the distances, and a number k. Decide if $\exists$ a closed tour that visits every node exactly once and that total length $\leq k$.

0/1 integer programming: Given m linear inequalities with integral certificates over n variables $u_1, u_2, \ldots, u_n$, decide if there is an assignment of zeros and ones to $u_1, \ldots, u_n$ satisfying all certificates.
certificate: $u_1, \ldots, u_n \in \{0,1\}$. verifier: if it satisfies all inequalities

Factoring: Given 3 numbers n, l, u, decide if n has a prime factor $p \in [l, u]$.
certificate: p, verifier: 1) $p \in [l,u]$ 2) p is a prime 3) $p \mid n$



Traveling Salesman, k-Clique, 0/1 integer programming, Subset sum: NPC

Composite number, Prime: P.

GI, Factoring: (Suspected to be) not P, not NPC

$EXP = \bigcup_{c \geq 1} DTIME(2^{n^c})$

Lemma. ~~$P \subseteq EXP$~~ $P \subseteq NP \subseteq EXP$

Proof. ($P \subseteq NP$) Suppose $L \in P$. Then $\exists$ poly-time TM M that decides L. Let $p(n) = 0$ and let $w = \varepsilon$. Let the verifier be M.

($NP \subseteq EXP$) Suppose $L \in NP$. Let $p: \mathbb{N} \to \mathbb{N}$ and TM M (verifier) be as in the definition. We decide L by enumerating all possible certificates $w \in \{0,1\}^{p(|x|)}$ and use M to check if w is a valid certificate.

$$2^{p(n)} n^{O(1)} \leq 2^{n^c} n^{O(1)} \leq 2^{n^{c+1}} = 2^{n^{O(1)}}, \quad \square$$

Nondeterministic Turing machine

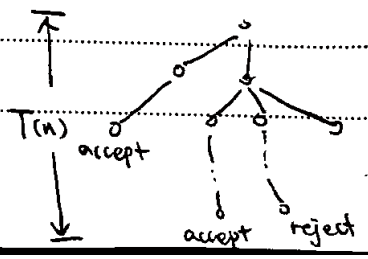
At any point in a computation, the machine proceeds according to several possibilities. The transition function

$$\delta : Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R, S\})$$

Formally, an NTM $M = (\Sigma, \Gamma, Q, \delta, q_0, q_{accept}, q_{reject})$. M accepts x if at least one of the possible computation paths leads to an accept state.

Def. Say M runs in time $T(n)$ if for every input $x \in \{0,1\}^*$ and <u>every</u> sequence of nondeterministic choice, M reaches an end step in $T(n)$ steps

Every NTM has an equivalent deterministic-TM. (simulation by BFS).



Def (Binary-choice NTM)

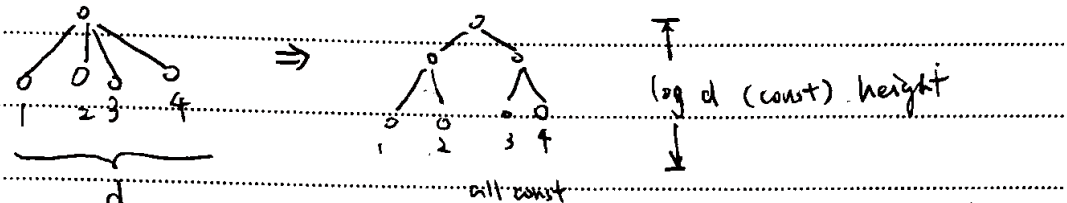
$M = (Q, \Sigma, \Gamma, \delta_1, \delta_2, q_0, q_{accept}, q_{reject})$

$\delta_1, \delta_2 : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$

At each step M applies δ_1 or δ_2

Lemma. Let $L \subseteq \{0,1\}^*$. If L can be decided by an NTM in time $T(n)$, then L can be decided by a binary-choice NTM in $O_M(T(n))$

Proof idea: 多叉树 → 二叉树



$d \le 2^{|Q| \times |\Gamma| \times 3}$. $\lceil \log d \rceil \le 3 \cdot |Q| \cdot |\Gamma|$ One step (of NTM) can be simulated in $3 \cdot |Q| \cdot |\Gamma| = O_M(1)$ steps (by binary-choice NTM)

Def (NTIME) Language $L \subseteq \{0,1\}^*$ is in $NTIME(T(n))$ if there exists an NTM M that runs in time $O(T(n))$ and decides L.

Theorem. $NP = \bigcup_{c \ge 0} NTIME(n^c)$

Proof. 1) Let $L \in \bigcup_{c \ge 0} NTIME(n^c)$ so ∃ NTM ∃ binary-choice NTM N that decides L in time $d \cdot n^c$ for some constant c. Let $p(n) = O(n^c)$, and let $w \in \{0,1\}^{p(n)}$ indicate which of the transition functions to apply. The verifier simulates N according to N and w, and checks if N accepts x.

For factorization of n:

$m = \lceil \log n \rceil$

NTM

1) guess $m \in [1, m]$

2) guess $p_1, p_2, \dots, p_m$

3) check if $p_1, \dots, p_m$ are prime

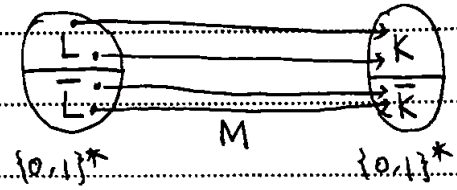
4) check if $n = p_1 \cdots p_m$.

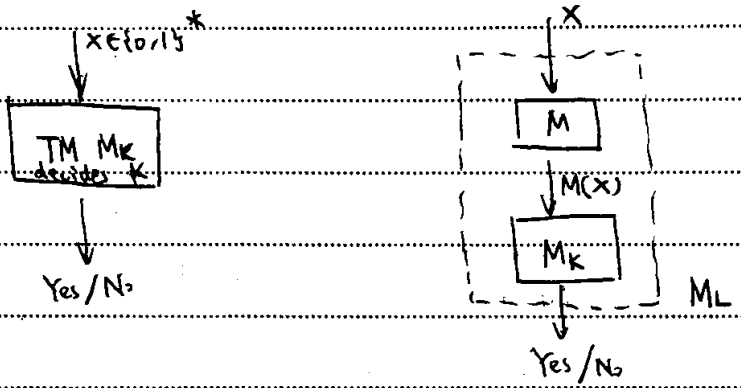
2) Let $L \in NP$, and let $p: \mathbb{N} \to \mathbb{N}$ and TM $V \in M$ (verifier) be as in the definition. Construct an NTM on input x:

1) Guess $w \in \{0,1\}^{p(|x|)}$ 2) Simulate M on input (x, w) 3) Accept if M accepts. □

Reduction in NP

$L, K \subset \{0,1\}^*$. Denote $L \leq_p K$ if $\exists$ poly-time TM M s.t. for all $x \in \{0,1\}^*$, $x \in L$ iff $M(x) \in K$





If M_K runs in poly-time, then M_L runs in poly-time.

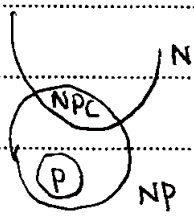
If $K \in P$, then $L \in P$.

Lemma: If $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$, then $L_1 \leq_p L_3$

Def. (NP-hard) Language $L \subseteq \{0,1\}^*$ is NP-hard, if for all language $K \in NP$, $K \leq_p L$. (L may not be in NP).

Def. (NP-complete) $L \subseteq \{0,1\}^*$ is NP-complete if

1) $L \in NP$, $\Rightarrow$ and 2) L is NP-hard.



Lemma. If L is NP-hard, and $L \in P$, then $P = NP$.

Lemma. If L is NP-complete, then $L \in P$ iff. $P = NP$.

$P = NP$ iff. $\exists$ poly-time alg. for ...

$P \neq NP$ iff. there is no poly-time alg for ...

SAT (Cook-Levin Theorem)

|

3SAT

|

CLIQUE

SAT (Boolean Satisfiability Problem) :

Variable : x, y, z can take TRUE or FALSE

literal : a variable or its inverse . E.g. $x, y, \neg x, \neg y$

clause : OR of one or more literals. E.g. $\neg x \vee y$, $\neg y \vee z \vee x$

formula : AND of one or more clauses

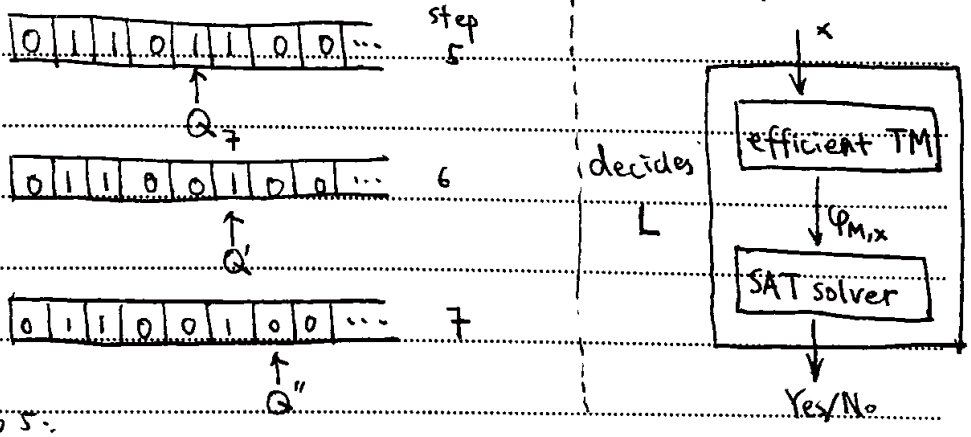
$\varphi = (\neg x \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z \vee y)$. Is φ satisfiable ?

Yes. ~~$y=1, z$~~ y, z = TRUE, x = TRUE or FALSE.

$SAT = \{ \langle \varphi \rangle \mid \varphi$ is satisfiable$\}$. $SAT \in NP$.

Cook - Levin Theorem : SAT is NP - complete

To prove SAT is NP - complete, it suffices to prove for every poly - time NTM M, for every $x \in \{0,1\}^*$, we can algorithmically construct a formula $\varphi_{M,x}$ such that M accepts x iff. $\varphi_{M,x} \in SAT$



Encoding : $H_{5,5}$ = TRUE, $Q_{5,7}$ = TRUE

$T_{i,j,k}$: $i \in [T(n)] = \{1, 2, \ldots, T(n)\}$, $j \in \Gamma$, $k \in [T(n)]$

$T_{i,j,k}$ = Cell i in the tape contains symbol j at step k.

$H_{i,k}$: $i, k \in [T(n)]$. $H_{i,k}$ = head is on cell i at step k.

$H_{q,k}$: $k \in [T(n)]$, $q \in Q$. TM is on state q at step k.

Size

~~Length~~: $T(n)^2|\Gamma| + T(n)^2 + T(n)|Q| = n^{O(1)}$ Boolean variables

Initialization: input $x \in \{0,1\}^*$ of length n.

$T_{i,x_i,0}$ = ~~false~~ true for $i = 1, 2, \ldots, n$

$T_{i,\sqcup,0}$ = true for $i = n+1, n+2, \ldots, T(n)$

$H_{1,0}$ = true

$Q_{q_0,0}$ = true

Restriction:

1) at most one symbol per cell. i.e., $\forall i,k \in [T(n)], \forall j \neq j' \in \Gamma$,

$$\neg T_{i,j,k} \vee \neg T_{i,j',k}$$

2) at least one symbol per cell. i.e., $\forall i,k \in [T(n)]$,

$$\bigvee_{j \in \Gamma} T_{i,j,k}$$

3) at most one state at a time i.e. $\forall k \in [T(n)], \forall q \neq q' \in Q$,

$$\neg Q_{q,k} \vee \neg Q_{q',k}$$

4) at least one state at a time. i.e. $\forall k \in [T(n)]$,

$$\bigvee_{q \in Q} Q_{q,k}$$

5) at most one head position at a time i.e. $\forall k \in [T(n)], \forall i \neq i' \in [T(n)]$,

$$\neg H_{i,k} \vee \neg H_{i',k}$$

6) at least one head position at a time. i.e. $\forall k \in [T(n)]$,

$$\bigvee_{i \in [T(n)]} H_{i,k}$$

Transition rule:

Obs. $A \to B$ is equivalent to $\neg A \vee B$, which is a clause

最后一步不转 转了，故 $T(n)-1$.

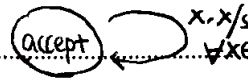
1) Cell remains unchanged unless written: $\forall i \in [T(n)], \forall k \in [T(n)-1], \forall j \neq j' \in \Gamma$,

$$T_{i,j,k} \wedge T_{i,j',k+1} \rightarrow H_{i,k}$$

2) Transition function δ: $\forall i \in [T(n)], \forall k \in [T(n)-1], \forall q \in Q, \forall j \in \Gamma$,

$$H_{i,k} \wedge Q_{q,k} \wedge T_{i,j,k} \rightarrow \bigvee_{(q',j',d) \in \delta(q,j)} H_{i+d,k+1} \wedge Q_{q',k+1} \wedge T_{i,j',k+1}$$

Assume that the TM loops in the accept state after accept, i.e.

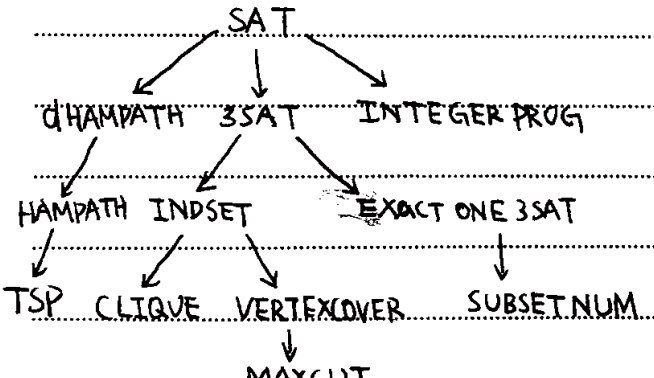


Halt in an accept state:

$$\bigvee_{j \in [T(n)]} Q_{accept,j}$$

Take AND of all the clauses: The size is $n^{O(1)}$. □

Web of reductions



3SAT: each clause has at most 3 literal

Theorem SAT $\leq_p$ 3SAT.

Proof: If some clause has > 3 literals, replace it by an equivalent 3CNF. E.g. $x_1 \vee x_2 \vee \overline{x_3} \vee \overline{x_4} \vee x_5$

$$= (x_1 \vee x_2 \vee z) \wedge (\overline{z} \vee \overline{x_3} \vee x_5)$$ □

引入新变量也只是多项式地变多.

Theorem. SAT $\leq_p$ INTEGER PROG

Proof. $x_1 \vee \bar{x}_2 \vee \bar{x}_3 \Rightarrow x_1 + (1-x_2) + (1-x_3) \geq 1$.

1: TRUE, 0: FALSE.

Let φ be a CNF on n variables with m clauses

For each Boolean variable, introduce a variable in INTEGER PROG denoted by $x_1, x_2, \ldots, x_n \in \{0,1\}$, $0 \leq x_i \leq 1$

For each clause in φ, sum up all literals (by replacing $\bar{x}_i$ by $1-x_i$) and asserts the sum is at least 1.

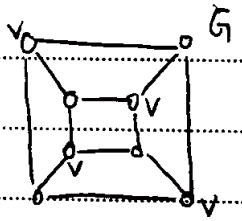
It is clear that φ is satisfiable iff. the integer linear programming is satisfiable. □

INDSET = $\{\langle G,k\rangle \mid$ G has an independent set of size k$\}$

Theorem. 3SAT $\leq_p$ INDSET

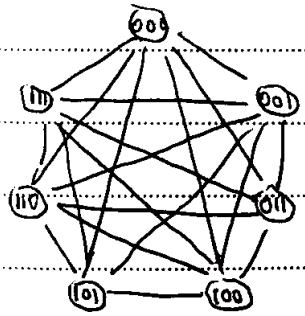
Let the 3CNF have m clauses

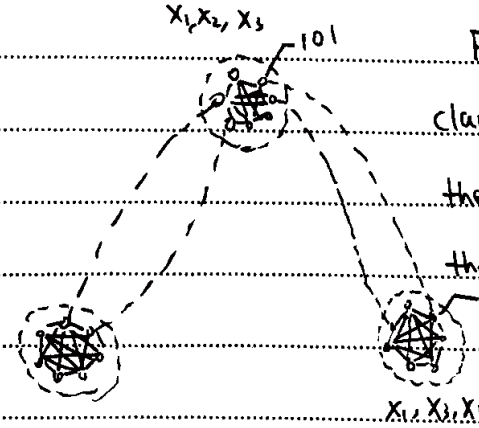
Proof. For each clause, make a cluster (clique) consisting of 7 satisfying assignments



E.g. $C = x_1 \vee \bar{x}_2 \vee \bar{x}_3$: (0,1,0) satisfies. $\langle G,4\rangle \in$ INDSET

$(x_1, x_2, x_3) = \{(0,0,0), (0,0,0), (0,1,1), (1,0,0), (1,1,0), (1,0,1), (1,1,1)\}$. $\langle G,5\rangle \notin$ INDSET



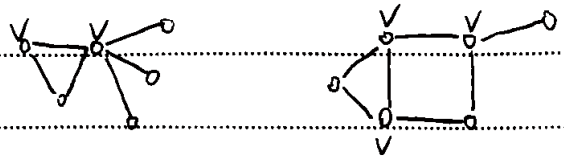


For different clauses, connect the assignments if they are inconsistent.

φ is satisfiable iff. G has an independent set of size m.

VERTEX COVER = $\{\langle G, k\rangle \mid G$ has a vertex cover of size $k\}$

a vertex cover of G is a set of vertices that includes at least one endpoint of every edge of G.



Theorem. INDSET $\leq_p$ VERTEX COVER

Proof. Fix G. If $V' \subset V(G)$ is a vertex cover, then $V(G) - V'$ is an ind set.

If $V \subset V(G)$ is an ind set, then $V(G) - V$ is a vertex cover.

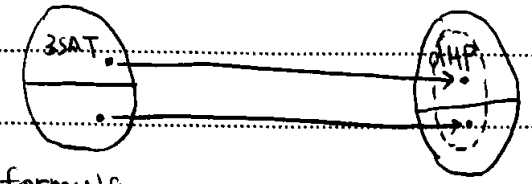
$\Rightarrow$ V' is a vertex cover iff $V(G) - V'$ is an ind set.

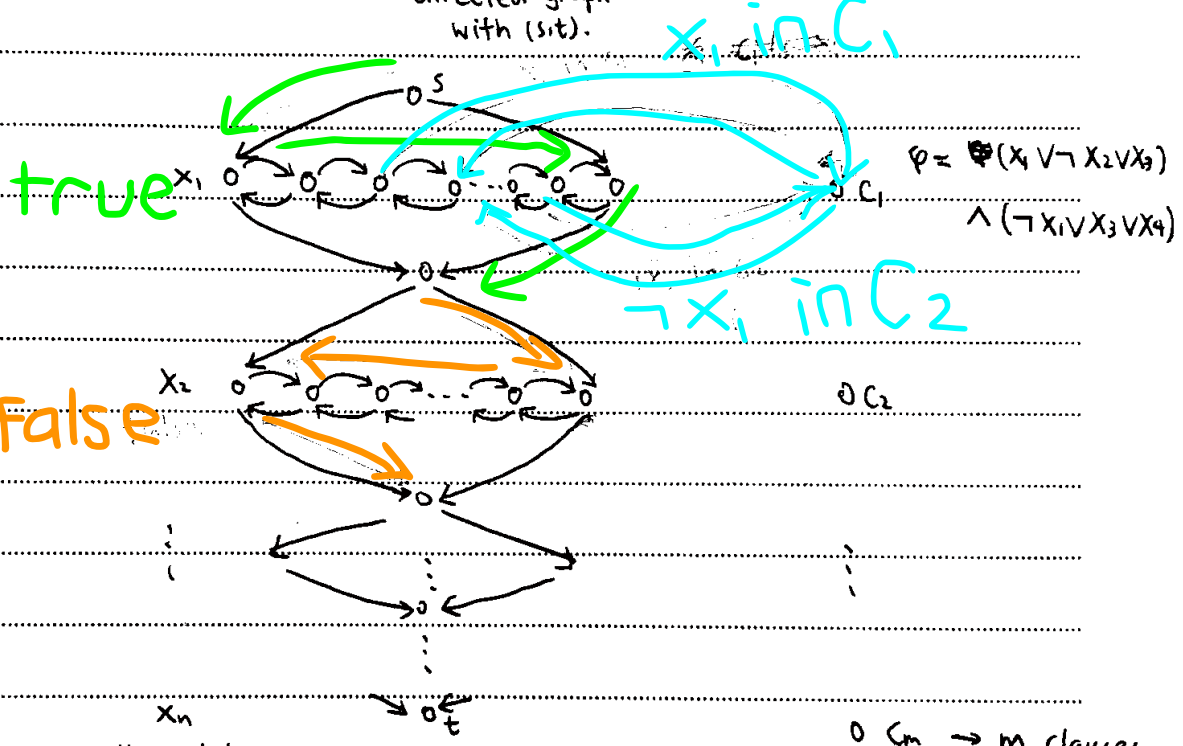
$\langle G, k\rangle \in$ VERTEX COVER iff. $\langle G, |V(G)| - k\rangle \in$ INDSET. $\square$

dHAMPATH = {⟨G, s, t⟩ | directed graph G has a Hamiltonian path from s to t.}

Theorem 7.16. 3SAT $\leq_p$ dHAMPATH

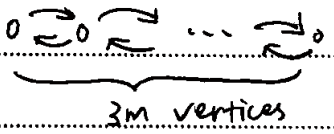
构造构件(gadget)，每个构件完成一定功能。(特殊形式的图)



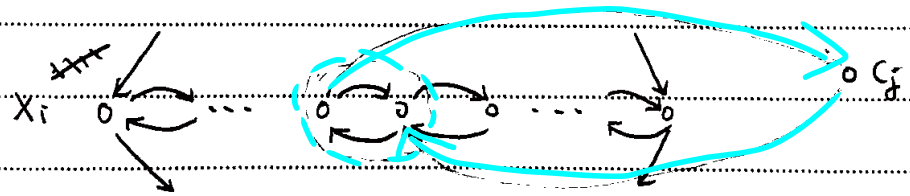


n = # variables

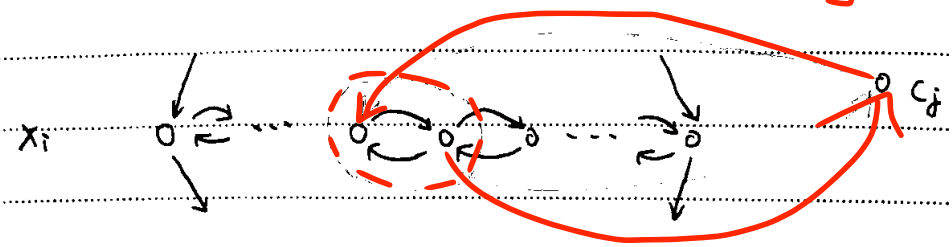
m = # clauses



If variable x_i appears in clause c_j, add the following edges



If $\neg x_i$ appears in c_j, add the following edges

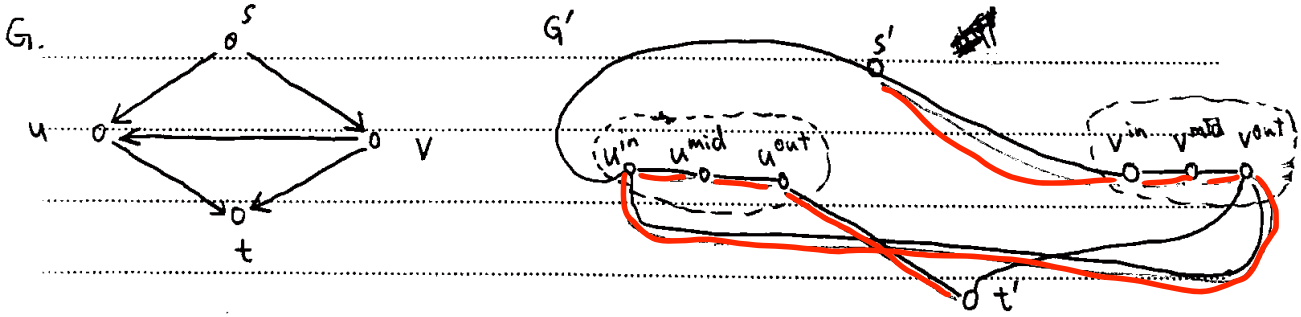


$\Rightarrow$ If φ is satisfiable then G has ham-path (s,t).

HAMPATH = $\{\langle G, s, t\rangle \mid$ undirected graph G has a Hamiltonian path from s to $t.\}$

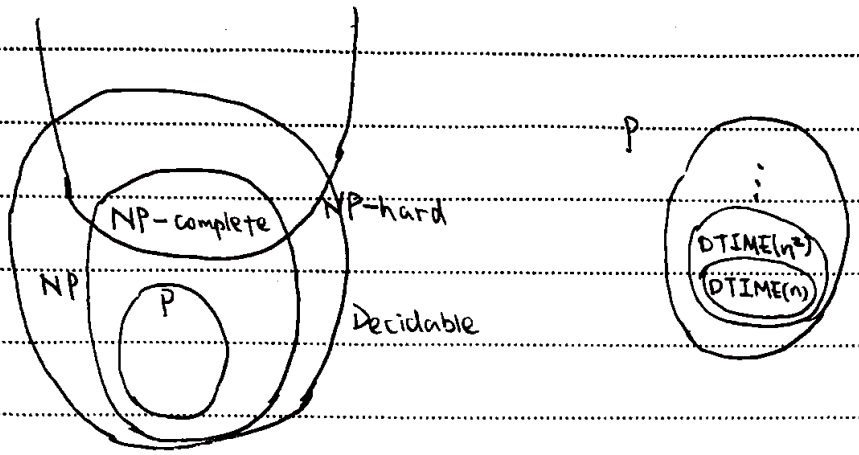
Theorem 7.55 $dHAMPATH \leq_p HAMPATH$

Each node u of G, except for s and t is replaced by a triple u^{in}, u^{mid}, u^{out} in G. Replace s by $s^{out} = s'$, t by $t^{in} = t'$.



For every $u \in V(G) - \{s,t\}$, connect u^{mid} with u^{in} and u^{out}.

If $(u,v) \in E(G)$, connect u^{out} with v^{in}. $\square$



Def (Time constructible). Function $T: \mathbb{N} \to \mathbb{N}$ is time constructible if $T(n) \geq n$ and there is a TM that computes $1^n \mapsto T(n)$ (in binary) in time $O(T(n))$.

Theorem (Time Hierarchy Theorem). If f, g are time constructible functions satisfying $f(n) \cdot \log f(n) = o(g(n))$, then $\mathrm{DTIME}(f(n)) \subsetneq \mathrm{DTIME}(g(n))$ (little-o)

diagonalization.

Proof. $\frac{g(n)}{\log(g(n))} = \omega(f(n)) \iff f(n) = o\left(\frac{g(n)}{\log g(n)}\right)$:

$f(n) \log f(n) = o(g(n)) \Rightarrow f(n) = o\left(\frac{g(n)}{\log f(n)}\right)$.

Case 1: $g(n) \geq f(n)^2$.

$$\frac{g(n)}{\log g(n)} \geq \frac{g(n)}{g(n)^{1/3}} \geq g(n)^{2/3} \geq f(n)^{4/3} = \omega(f(n))$$

Case 2: $g(n) < f(n)^2$.

$$\frac{g(n)}{\log g(n)} \geq \frac{g(n)}{2f(n)} = \omega(f(n))$$

Claim: $t(n) = \left\lfloor \frac{g(n)}{\lfloor \log g(n) \rfloor} \right\rfloor$ is constructible in time $O(g(n))$

Construct the following TM M:

On input $x \in \{0,1\}^*$, simulate M_x on input x for $t(n)$ steps, and flip the output, i.e.

1) If M_x accepts x in $t(n)$ steps, reject.

2) If M_x rejects x in $t(n)$ steps, accept.

3) If M_x does not stop in $t(n)$ steps, accept.

Let $L = L(M)$.

Claim: $L \in \text{DTIME}(g(n))$. Simulation takes $O(t(n) \cdot \log t(n))$

$$t(n) = \left\lfloor \frac{g(n)}{\lfloor \log g(n) \rfloor} \right\rfloor \leq \frac{g(n)}{\lfloor \log g(n) \rfloor} \leq O\left(\frac{g(n)}{\log g(n)}\right)$$

~~$O(t(n))$~~

$$O(t(n) \cdot \log t(n)) = O\left(\frac{g(n)}{\log g(n)} \cdot \log O\left(\frac{g(n)}{\log g(n)}\right)\right)$$

$$= O\left(\frac{g(n)}{\log g(n)} \left(\log\left(\frac{g(n)}{\log g(n)}\right) + O(1)\right)\right)$$

$$= O\left(\frac{g(n)}{\log g(n)} \left(\log g(n) - \log(\log g(n)) + O(1)\right)\right)$$

$$= O(g(n))$$

Claim: $L \notin \text{DTIME}(f(n))$.

Suppose for contradiction that L is decidable by some TM M_α in time $O(f(n))$.

Consider M_α on input α.

Case 1: $\alpha \in L$. by the definition of L, M rejects α, or does not halt in $t(n)$ steps. It contradicts to the assumption that M decides L in time $O(f(n))$.

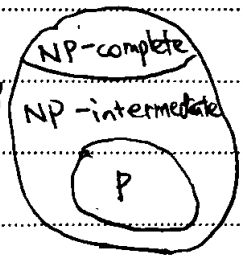
Case 2: $\alpha \notin L$. M_α accepts α. Contradiction. □

Corollary. $\text{DTIME}(n) \subsetneq \text{DTIME}(n^2) \subsetneq \text{DTIME}(n^3) \subsetneq \cdots$,

$P \subsetneq EXP$, ...

NP-intermediate problems.

Theorem (Ladner). Suppose $P \neq NP$. There exists a language $L \in NP \setminus P$ that is not NP-complete.



Proof idea. "padding". 人为填充长度让输入变长

SAT = {ϕ | Boolean formula ϕ has a satisfying assignment}

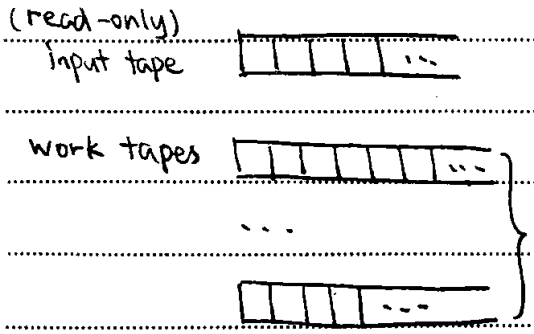
$SAT \in DTIME(2^n)$

$SAT_H = \{\phi 0 1^{nH(n)} \mid \phi \text{ is satisfiable}\}$

Let $H(n) = 2^n$, then $SAT_H \in P$.

Ladner 证明了有适当的 H(n) 作为长度使得既不NP完全又未到达 P.

Def (Space complexity): Turing machine M runs in space S(n) if for every input x, it uses at most S(|x|) cells on its work tapes. (excluding the read-only input tape)



Def. Let $S: \mathbb{N} \to \mathbb{N}$. Language $L \in SPACE(S(n))$ if there is a TM that runs in space $O(S(n))$ and decides L.

Example: $SAT \in SPACE(n)$. On input $\langle\phi\rangle$, TM M enumerates all

truth assignments $\rho: \{x_1, \ldots, x_n\} \to \{0,1\}$. Verify if $\emptyset | \rho$ is true.

Eg Example (Majority Element)

input: n, $a_1, a_2, \ldots, a_n$

output: majority element (i.e., #occurrences $> n/2$)

Pick a random $i \in \{1, \ldots, n\}$. Check if a_i is the majority element. Repeat if not.

time: $\frac{1}{2} \times 1 + \frac{1}{2^2} \times 2 + \frac{1}{2^3} \times 3 + \cdots$

space: $O(\log n)$

Def. Let $S: \mathbb{N} \to \mathbb{N}$. Language $L \in NSPACE(S(n))$ if there is an NTM that runs in space $O(S(n))$ and decides L.

Time and Space

Theorem. $DTIME(S(n)) \overset{①}{\subseteq} SPACE(S(n)) \overset{②}{\subseteq} NSPACE(S(n)) \overset{③}{\subseteq} DTIME(2^{O(S(n))})$

where $S(n) \geq \log n$.

$\subsetneq$ ↓ not yet proved.

①, ②: obvious

③ "Configuration graph".

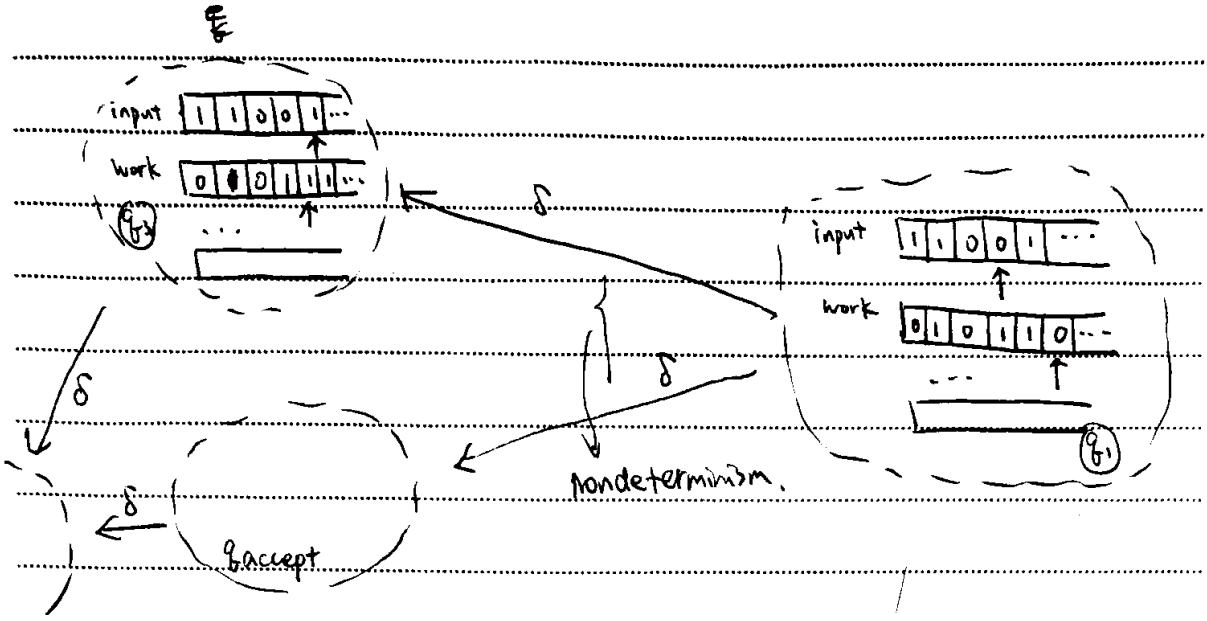
bits to encode a configuration:

$\lceil \log |Q| \rceil + \lceil \log (n+1) \rceil + m \cdot \lceil \log S(n) \rceil + n \lceil \log |\Gamma| \rceil + m \cdot S(n) \cdot \lceil \log |\Gamma| \rceil$

↑ state in Q; ↑ input head; ↑ # work tapes; ↑ work head; ↑ symbol on input tape; ↑ symbol on work tape

$= O_M(1) + O(\log n) + O_M(\log S(n)) + O_M(n) + O_M(S(n))$

$= O_M(S(n))$.



Let $G_{M,x}$ be the configuration graph for NTM M and input $x \in \{0,1\}^*$. $|V(G_{M,x})| \leq 2^{O_M(S(n))}$.

Given two vertices C_1 and C_2, one can decide if $C_1 \to C_2$ in time $O_M(S(n))$.

$|E(G_{M,x})| \leq 2^{O_M(S(n))}$.

~~Given~~ $G_{M,x}$ has one start ~~vertex~~ vertex and one accept state (W.LOG) C_{start} and C_{accept}. Use BFS or DFS to check if C_{start} and C_{accept} is connected in $2^{O_M(S(n))}$ time ($O(|V|+|E|)$). □

Theorem (Variant). $SPACE(S(n)) \subseteq DTIME\left(2^{O(S(n)/\log S(n))}\right)$

Def: (Space constructible). Function $S: \mathbb{N} \to \mathbb{N}$ is space constructible if there exists a TM that on input 1^n outputs the binary representation of $S(n)$ in space $O(S(n))$.

For example, $\lfloor \log n \rfloor$, n, n^2, n^3, 2^n, ... are space constructible.

input : 1^n

output : $\lfloor \log n \rfloor$ in binary

n in binary in time $O(n)$, space $O(\log n)$

$\lfloor \log n \rfloor$ is the # of bits in the binary representation of n.

e.g. $(111)_2 = (7)_{10}$ $\lfloor \log 7 \rfloor = 2$ $(100)_2 = (4)_{10}$, $\lfloor \log 4 \rfloor = 2$.

in space $O(\log \log n)$.

In total, we use space $O(\log n)$

~~Str~~ Space Hierarchy Theorem.

Theorem If f and g are space constructible functions satisfying $f(n) = o(g(n))$, $SPACE(f(n)) \subsetneq SPACE(g(n))$.

Proof. "diagonalization"

Goal: construct $L \in SPACE(g(n))$ and $L \notin SPACE(f(n))$.

Construct a TM as follows.

On input α, simulate M_α on α up to space ~~#~~ $mS(n)$

(For example, for each work tape, mark the ~~Stor~~ $S(n)+1$ th cell using a special symbol. Reject once the symbol is read).

Count the number of steps: Once the number of steps exceeds $|Q| \lceil \log(n+1) \rceil (\lceil \log S(n) \rceil)^m |\Gamma|^{S(n)}$, reject (长时间, 小空间, 这说明无限循环)

If M accepts, reject. If M rejects, accept.

Let $L = L(M)$. Let $S(n)$ be such that $f(n) = o(S(n))$ and $S(n) = o(g(n))$, and $S(n)$ is constructible in space $O(g(n))$.

(For example, $S(n) = f(n) \lfloor \log \frac{g(n)}{f(n)} \rfloor$)

Note that $L \in SPACE(g(n))$.

Claim: $L \notin SPACE(f(n))$.

Suppose $L \in SPACE(f(n))$. By definition, $\exists$ a TM M_β that decides L in ~~alg~~ space $O(g(n))$. Consider M_β on input β.

Case 1: $\beta \in L$. M_β rejects β. contradiction

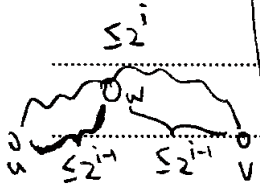
Case 2: $\beta \notin L$. i) M_β accepts β, or,

ii) M_β uses "too much" space or time

Contradiction. □

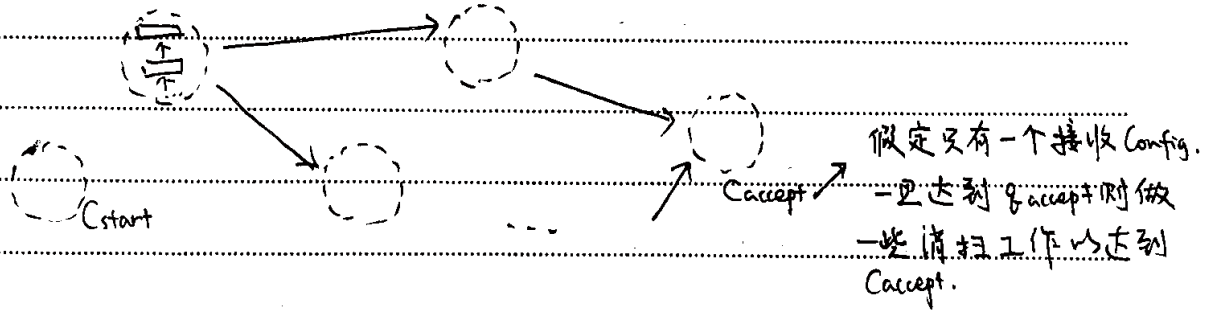
Savitch's Theorem. For any $S: \mathbb{N} \to \mathbb{N}$, where $S(n) \geq \log n$, $NSPACE(S(n)) \subseteq SPACE(S(n)^2)$. (所需空间上所谓 P、NP 相等)

REACH(u,v,i) = 1 if there is a path from u to v of length $\leq 2^i$.




```
REACH(u,v,i)
    if i=0, check if (u,v) ∈ E(G), and return 0/1
    enumerate all w∈V(G)
        if REACH(u,w,i-1) and REACH(w,v,i-1) then
            return 1.
```

Proof idea: "Configuration graph"



Proof. Let $L \in NSPACE(S(n))$. Then $\exists$ an NTM M that decides L in

space $O(S(n))$. Our goal is to design a deterministic TM that decides L in space $O(S(n)^2)$.

Consider configuration graph $G_{M,x}$, where $x \in \{0,1\}^n$ is the input.

1) Each vertex is a configuration that can be described in $O_M(S(n))$ bits.

2) $|V(G_{M,x})| \leq 2^{O_M(S(n))}$.

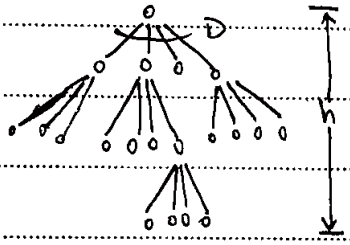
3) Each vertex has at most 2 outgoing edges.

4) Two neighbours can be computed in time / space $O_M(S(n))$

5) $G_{M,x}$ has ~~one and only~~ a start vertex C_{start} and an accept vertex C_{accept}.

6) M accepts x iff. C_{start} and C_{accept} are connected

REACH:



time: $O(D^h)$

space: h× space used in each node

Space used in each recursion node:

$\leq |u| + |v| + |w| + O_M(\log S(n)) \rightarrow$ space used to encode i

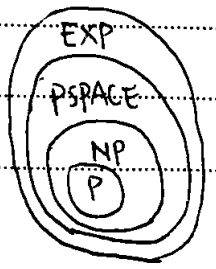
$= O_M(S(n))$.

Total space $O_M(S(n)) \times \text{depth} = O_M(S(n)^2)$ □

The class PSPACE.

Def. $PSPACE = \bigcup_{c \geq 1} SPACE(n^c)$

$NP \subseteq PSPACE$



$NPSPACE = \bigcup_{c\geq 1} NSPACE(n^c) = \bigcup_{c\geq 1} PSPACE(n^{2c}) = PSPACE$

$P \subset NP \subset PSPACE \subseteq EXPTIME$

PSPACE Completeness

Def. L is PSPACE - complete if

1) $L \in PSPACE$

2) ~~every~~ $K \leq_p L$ for every $K \in PSPACE$.
↑ poly-time reduction

The TQBF problem

Def. A quantified Boolean formula (QBF) is a Boolean formula of the form $Q_1 x\, Q_2 x \cdots Q_n x_n\, \varphi(x_1, \ldots, x_n)$ where $Q_i \in \{\forall, \exists\}$

Quantifier: $\forall x\, \varphi(x)$: for all $x \in \{0,1\}$, $\varphi(x)$ is true

$\exists x\, \varphi(x)$: for some $x \in \{0,1\}$, $\varphi(x)$ is true.

Eg. $\varphi(x,y) = (x \vee \bar{y}) \wedge (\bar{x} \vee y)$.

x	y	φ
0	0	1
0	1	0
1	0	0
1	1	1

$\exists x \forall y\, \varphi(x,y)$ = False

$\exists y \forall x\, \varphi(x,y)$ = True.

SAT: $\exists x_1 \cdots \exists x_n\, \varphi(x_1, \ldots, x_n)$.

Game view: $\exists x_1\, \forall x_2\, \exists x_3 \cdots \forall x_k\, \varphi(x_1, \ldots, x_k)$

Two players. Alice ($\exists$), Bob($\forall$). Chose the values of x_i's (0 or 1) in turn. Alice wins iff. φ is true. Then Alice always has a winning strategy. (QBF为真 iff. 先手有必胜策略)

为假则后手必胜.

任何此类游戏(下棋等)都有必胜策略

Tic Tac Toe:

X_6	O_2	X_3
	X_1	X_7
O_4		X_5

0000	0001	0010
0011	0100	0101
0110	0111	1000

a_i, b_i, c_i, d_i encodes a position

$(\exists a_1 \exists b_1 \exists c_1 \exists d_1)(\forall a_2 \forall b_2 \forall c_2 \forall d_2) \cdots (\exists a_q \exists b_q \exists c_q \exists d_q)\ \varphi(a_1, \ldots, d_q)$.

QBF of a game $\in$ TQBF iff. the first player has a winning strategy

Theorem TQBF is PSPACE - complete

Proof. First we prove TQBF $\in$ PSPACE

Let $\psi = Q_1 x_1 \cdots Q_n x_n\ \varphi(x_1, \ldots, x_n)$ be a QBF with n variables and size m.

TQBF ($\psi = Q_1 x_1 \cdots Q_n x_n\ \varphi(x_1, \ldots, x_n)$):

If $n=0$, evaluate ψ

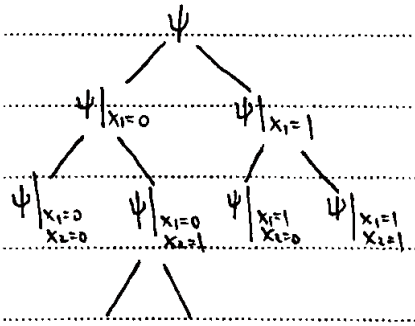
Let $\psi_0 = \psi|_{x_1=0}$

$\psi_1 = \psi|_{x_1=1}$

If $Q_1 = \exists$ and (TQBF(ψ_0) or TQBF(ψ_1) then return 1

If $Q_1 = \forall$ and TQBF(ψ_0) and TQBF(ψ_1) then return 1.

return 0 otherwise.



depth: n.

space per node: $O(m)$.

total space: $O(mn)$

$\Rightarrow$ TQBF $\in$ PSPACE.

4

Next we prove $L \leq_p$ ~~PSPACE~~ TQBF for every $L \in$ PSPACE.

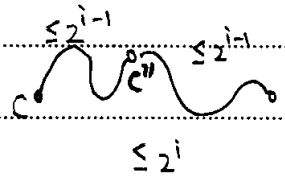
Let M be a TM that decides L in space $S(n)$, and ~~let~~ let $x \in \{0,1\}^*$. Construct a QBF $\psi = \psi_{M,x}$ of size $O_M(S(n)^2)$ that is true iff. M accepts x.

Claim: Let $G_{M,x}$ be the config. graph on input x.

1) Every vertex can be described using $O_M(S(n))$ bits.

2) $\exists$ an $O_M(S(n))$-size CNF formula $\varphi_{M,x}(c,c')$ that decides if c' ~~end~~ is the next configuration.

Let $\psi_i(c,c')$ be true iff there is a path from c to c' of length $\leq 2^i$. If $i=0$, $\psi_0(c,c') = \varphi_{M,x}(c,c')$. Let $m = O_M(S(n))$ be the number of bits needed to encode a vertex in $G_{M,x}$. Let $\psi = \psi_m$.



$\psi_i(c,c') = (\exists c'')(\psi_{i-1}(c,c'') \wedge \psi_{i-1}(c'',c'))$

$size(\psi) \geq 2\, size(\psi_{i-1}) \geq \cdots \geq 2^m$, $m = O_M(S(n))$

$\psi_i(c,c') = (\exists c'')(\forall D)(\forall E)\left((D=c \wedge E=c'') \vee (D=c'' \wedge E=c') \Rightarrow \psi_{i-1}(D,E)\right)$

then $size(\psi_i) = size(\psi_{i-1}) + O(m)$

$size(\psi_m) = O_M(S(n)^2)$ □