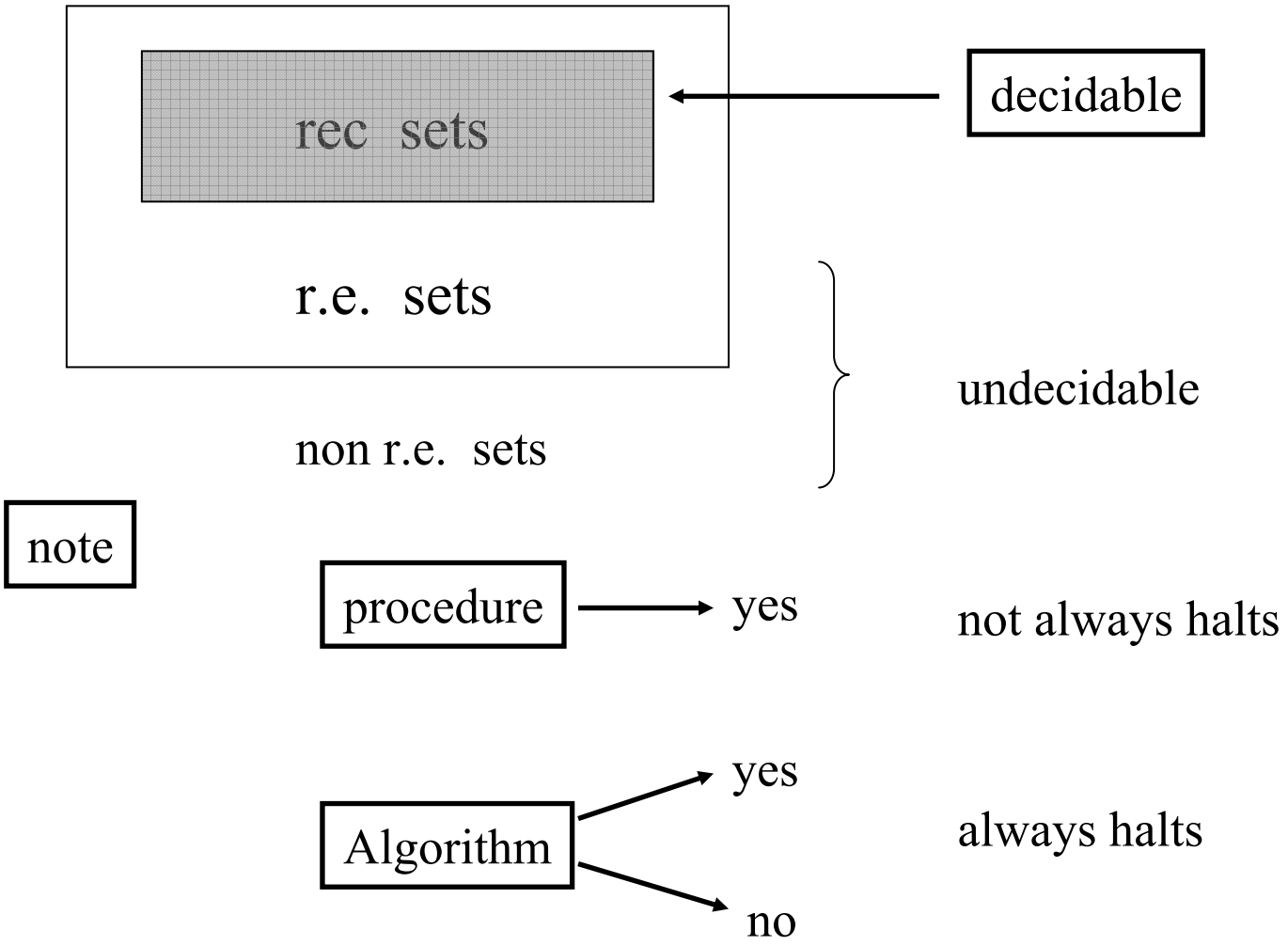
Comparison of several polynomial and exponential time complexity functions

Time complexity function	Size n					
	10	20	30	40	50	60
n	.00001 second	.00002 second	.00003 second	.00004 second	.00005 second	.00006 second
n^2	.0001 second	.0004 second	.0009 second	.0016 second	.0025 second	.0036 second
n^3	.001 second	.008 second	.027 second	.064 second	.125 second	.216 second
n^5	.1 second	3.2 second	24.3 second	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 second	58 minutes	6.5 years	3855 centuries	$2*10^8$ centuries	$1.3*10^{13}$ centuries

Size of Largest Problem Instance Solvable in 1 Hour

Time complexity function	With present computer	With computer 100 times faster	With computer 1000 times faster
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31.6N_2$
n^3	N_3	$4.64N_3$	$10N_3$
n^5	N_4	$2.5N_4$	$3.98N_4$
2^n	N_5	$N_5 + 6.64$	$N_5 + 9.97$
3^n	N_6	$N_6 + 4.19$	$N_6 + 6.29$

Intractable problems



Definition : A problem is said to be **intractable** if it is so hard that
no polynomial time algorithms can **possibly** solve it.

$\therefore$ Certain decidable problems may be intractable !!!

What are **polynomial time** algorithms ?

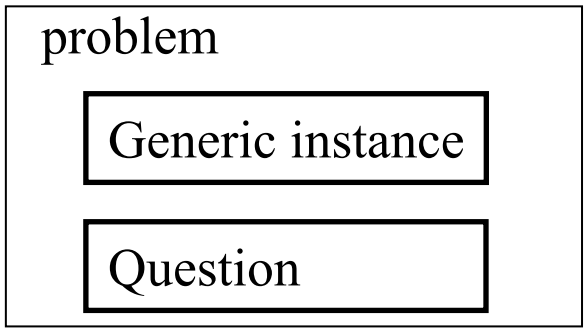
In what ?
In which computer ?

Why "possibly" ?

problem

Given I and P , find S satisfying P.

∴



A generic instance is specified in terms of various **components**(parameters) such as **sets,graphs,functions,numbers**, ….

A question is asked in terms of the generic instance.

Decision problems

"yes" or "no" answers

Optimization problems

Traveling Salesman

Instance: A **finite set** C = $\{c_1, c_2, \ldots, c_m\}$ of cities , a **distance** $d(c_i , c_j) \in Z^+$ for each pair of cities $c_i , c_j \in C$, and a **bound** $B \in Z^+$ (where Z^+ denote the **positive integers**)

Question: Is there a "tour" of all cities in C having total length no more than B.

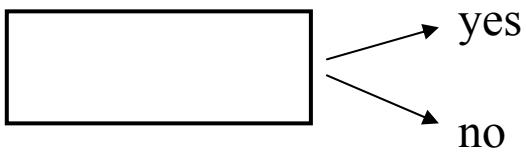
$< c_{\pi(1)}, c_{\pi(2)}, \ldots, c_{\pi(m)} >$

such that

$$\left[\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)})\right] + d(c_{\pi(m)}, c_{\pi(1)}) \leq B.$$

"yes " or "no"

Algorithm



a TM that always halts

An algorithm is a step - by - step porcedure(consisting of a finite sequence of instructions) for solving a problem.

Polynomial time algorithms

exponential time algorithms

Time Complexity T(n)

T(n) ≡ **<u>time requirements</u>** of an algorithm for solving a problem of **<u>size</u>** n

# of steps rather than absolute time

Why?

Problem length

||

Input Length

Related to an encoding scheme

T(n) = max{ t | t is time required by the algorithm to solve a problem instance of size n }

∴ Need two underlying assumptions:

(i) The model of computation

TM

(ii) a reasonable encoding scheme

?

How to encode a problem

"**Reasonable** encoding schemes"

↓

What is this ?

A generally accepted meaning of "**reasonable**" includes :

(i) **Conciseness**

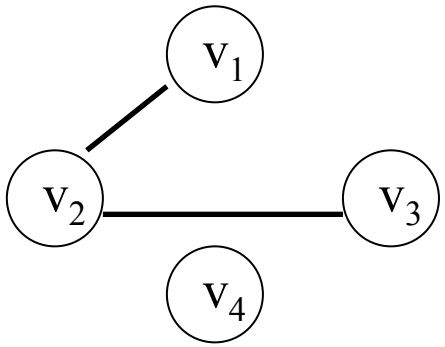
No unnatural padding

(ii) **Decodability**

Existence of a **polynomial time algorithm** capable of **extracting** a description of all components of a problem instance from encoded data.

Unfortunately, no definite way for satisfying this !!!

(even though the existence of a reasonable encoding scheme is intuitively obvious)



Descriptions of the graph G=(V,E) where V= {V_1, V_2 , V_3 , V_4 } and E= { {V_1 , V_2 },{V_2, V_3 }}, under three different encoding schemes.

Encoding Scheme	String	Length
Vertex list , Edge list	V[1]V[2]V[3]V[4](V[1]V[2])(V[2]V[3])	36
Neighbor list	(V[2])(V[1]V[3])(V[2])()	24
Adjacency matrix rows	0100/1010/0100/0000	19

Encoding Scheme	Lower Bound	Upper Bound
Vertex list , Edge list	4v + 10e	4v + 10e + (v + 2e)*[$\log_{10}$v]
Neighbor list	2v + 8e (= 4×2e)	2v + 8e + 2e*[$\log_{10}$v]
Adjacency matrix rows	v^2 +v - 1	v^2 +v - 1

However, most people would agree on
whether or not a particular encoding
scheme is reasonable.
(although the absence of a formal way is somewhat **uncomfortable**)

How can you resolve this difficulty ?
Setting up
“ a standard encoding scheme”

Standard encoding scheme

Flexible enough to accommodate any problem with minor modifications.

Basic assumption

Generic instances of problems consist of basic types of **set theoretic objects.**

Standard encoding scheme

$$f_e : D_\pi \longrightarrow S,$$

D_π= {problem instances }
S = { “structured” strings } $\subset \Sigma^*$
Σ = { 0, 1, -, [,], (,), , }

"structured" strings

Σ = { 0, 1, -, [,], (,), , }

(1) The **binary representation** of an integer k is a structured string representing k.
(put " - " if k is negative.)

(2) If x is the structured string representing an integer k , then
[**x**] is a structured string that can be used as **a name**

(3) If $\mathbf{x_1, x_2 , \ldots , x_m}$ are structured strings representing the objects $X_1, X_2 , \ldots , X_m$, then $(\mathbf{x_1, x_2 , \ldots , x_m})$ is a structured string representing $< X_1, X_2 , \ldots , X_m >$.

Example

integers 0 , -1 , 11 , 100 , -101 , ……

unstructured elements [0],[1], ….

(vertices , elements , cities , ….)

sequences, set } ([0],[1],….)

graphs (x , y)

finite functions ((x_1 ,y_1) , (x_2 ,y_2) ,….., (x_i ,y_i) , ….)

$f : X \rightarrow Y$

$X=\{x_1, x_2 , \ldots, x_n \}$, $Y=\{y_1 , y_2 , \ldots, y_n \}$

rational numbers $\underline{(p,q)}$ ⟵ p/q

O.k. , now , we are comfortable !!!

∴ The length of input can be determined.

Input Length

Length : $D_\pi \longrightarrow Z^+$

- encoding - independent
- polynomially related to the input length that we would obtain from a reasonable encoding scheme.

String lengths , $|x_1|$ and $|x_2|$ are **<u>polynomially related</u>** if there exist two polynomials p_1 and p_2 such that

$$|x_1| \leq p_1[|x_2|] \text{ , and}$$
$$|x_2| \leq p_2[|x_1|] .$$

The function “Length” is not unrealistic !!!

Why ?

An encoding scheme e for π is said to be
reasonable if Length(I) and |x| are polynomially
related for all x , where x is the string encoding for $I \in D_\pi$ under e.

Observation

The intractability of a problem turns out to be essentially independent of :

(1) particular encoding scheme

(2) the computer model used for determining time complexity

Why ?

Independence of computer model

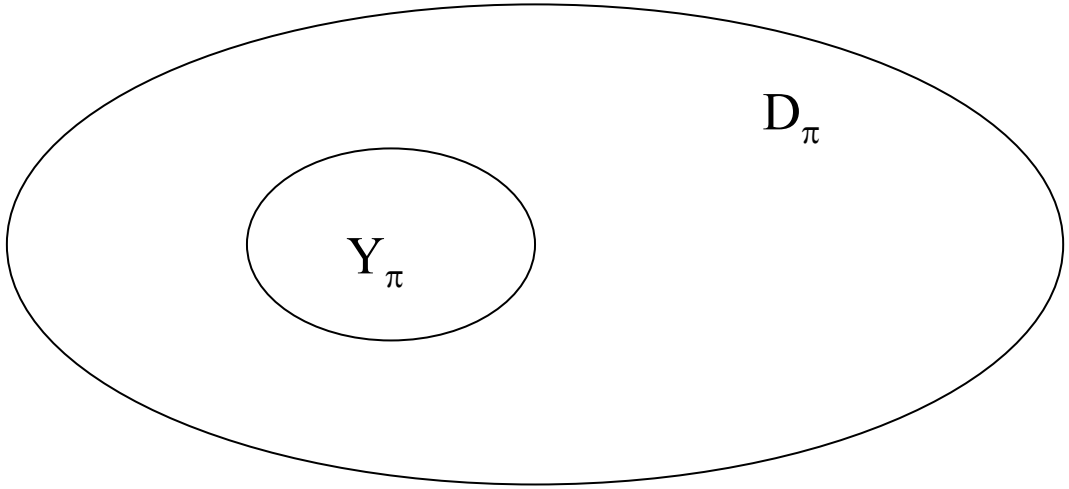
Simulated machine B	Simulating machine A		
	1TM	kTM	RAM
1-Tape Turing Machine (1TM)	-	O(T(n))	O(T(n)logT(n))
k-Tape Turing Machine (KTM)	$O(T^2(n))$	-	O(T(n)logT(n))
Random Access Machine (RAM)	$O(T^3(n))$	$O(T^2(n))$	-

Hopcroft , Ullman (1969)

Aho,Hopcroft , Ullman (1974)

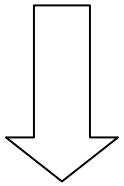
π : a problem

D_π : The set of instances of a decision problem



$Y_\pi = \{ p \mid p \in D_\pi$ and the solution of p is “yes” $\}$

An **instance** I of a problem π belongs to D_π

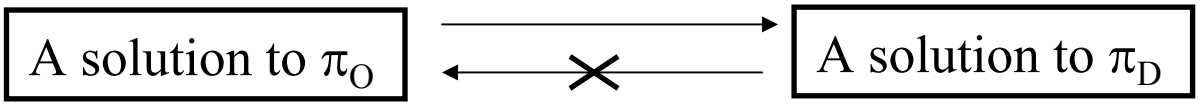


"I" can be obtained from the generic instance by substituting particular objects of specified types for all the generic components

π : a problem

π_O : Optimization version

π_D : Decision version



$\therefore$ A decision problem is no harder than its corresponding optimization problem !!!

If a decision problem is NP-complete, then its corresponding optimization problem is **at least as hard** !!!

$\therefore$ Even though the theory of NP-completeness **restricts attention to only decision problems** , we can **extend the implications of the theory to optimization problems as well.**

Why restricting attention to only decision problems ?

These problems have a very **natural** , **formal counterpart** which is a suitable object to study a mathematically precise theory of computation !!!

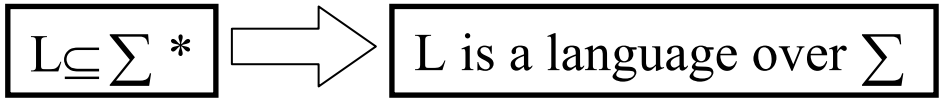
What is this counterpart ?

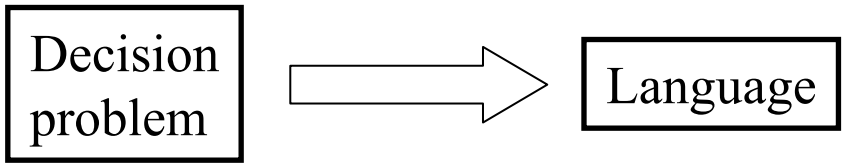
“Language!!!”

Language : L

Σ ≡ a finite set of symbols

Σ^* ≡ the set of all finite strings of symbols from Σ
(including empty string ε)



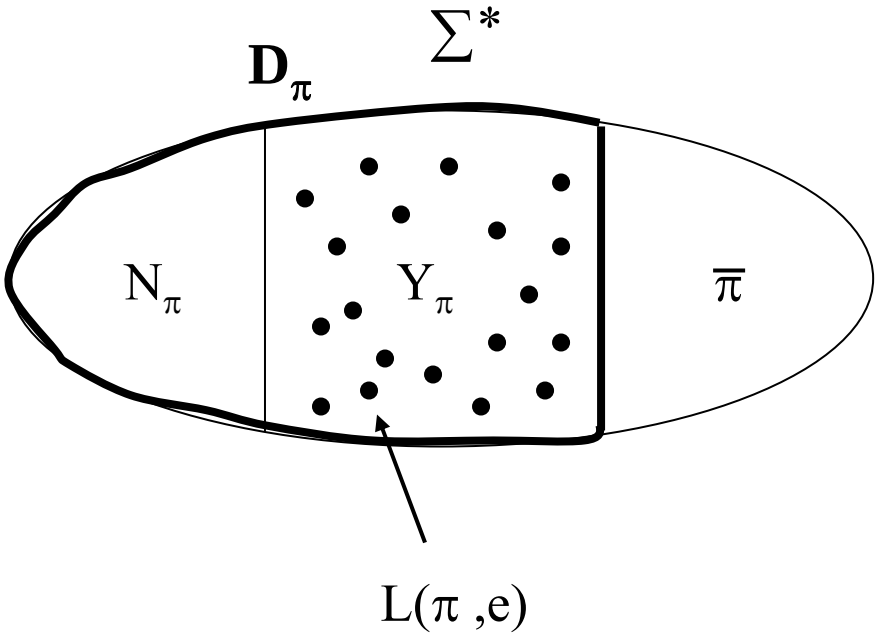


An encoding scheme e for a problem π provides a way of describing each instance $I \in D_\pi$ by an appropriate string of symbols over some alphabet $\sum$!!!

$e: D_\pi \rightarrow \Sigma^*$

$\forall I \in D_\pi, e(I) \in \Sigma^*$

The problem π and its encoding scheme *e* partition Σ* into three class of strings



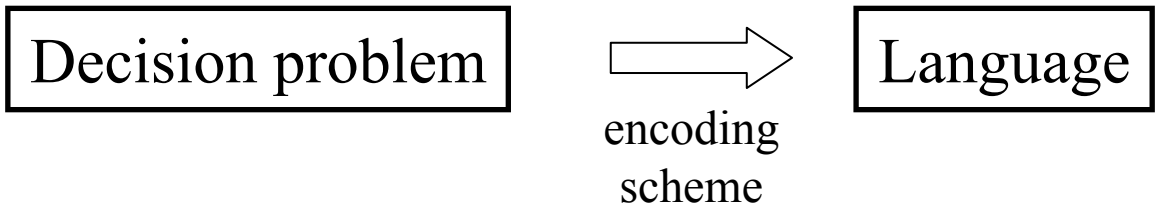
$L(\pi,e) = \{\ x \in \Sigma^* \mid \Sigma$ is the alphabet used by e , and x is the encoding under e of an instance $I \in Y_\pi\}$

$$L(\pi,e) \subseteq \sum^*$$

$\therefore L(\pi,e)$ is a language !!!

$\therefore$ Formal theory of language can be applied to decision problem

Observation



Does the language derived from a decision problem depend on encoding schemes ?

no !!!

Why ?

Reasonable encoding scheme !!!

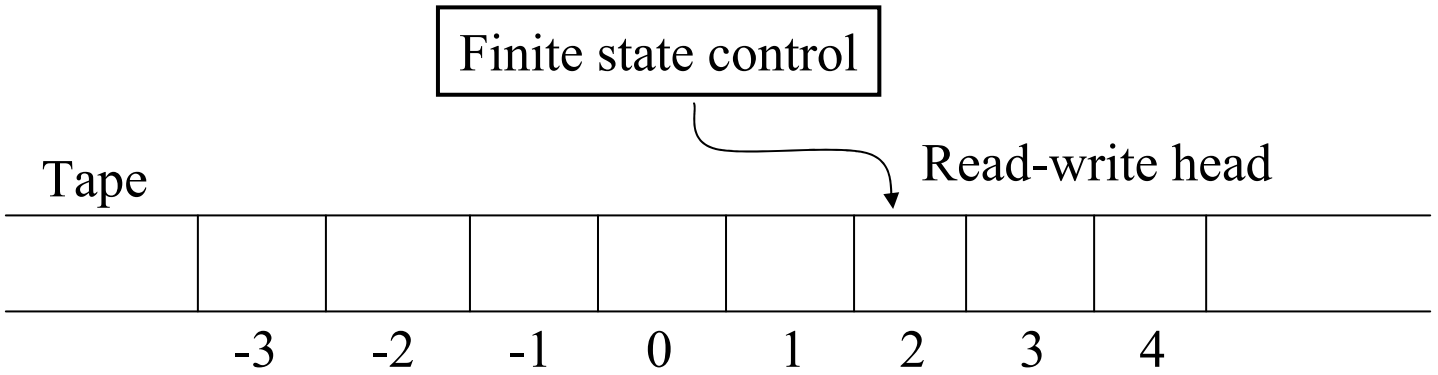
⇩

We may assume a reasonable encoding scheme e for a problem π

⟶ However , if requested , we could specify a particular encoding scheme e

Deterministic Turing machines and the class P

Deterministic one-tape Turing machine



A program for a DTM specifies the following information :

(1) A finite set Γ of tape symbols including a subset $\sum \subset \Gamma$ of input symbols and a distinguished bank symbol $b \in \sum$.

(2) a finite set Q of states, including a distinguished start-state q_0 and two distinguished halt-states q_Y and q_N ;

(3) a transition function $\delta : (Q - \{q_Y , q_N \}) \times \Gamma \longrightarrow Q \times \Gamma \times \{-1 , +1 \}$.

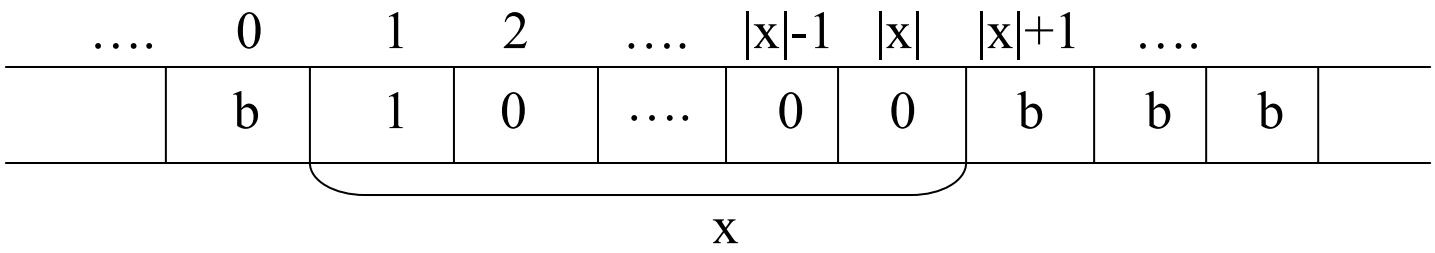
note

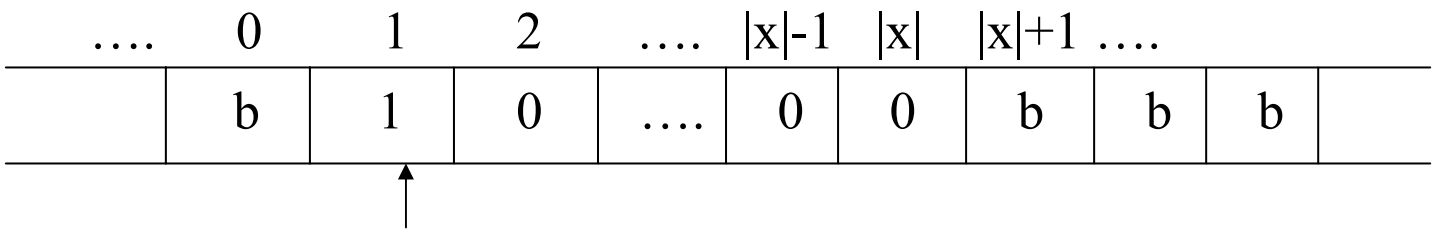
$\{q_Y , q_N \}$

$M=\{ Q, \sum , \Gamma, \delta, q_0 ,b,F\}$

$x \in \sum^*$

initially





$\delta (q_i , x_i) \longrightarrow (q_j , x_k , m)$

± 1

$q_j = q_N$: no

$q_j = q_Y$: yes

$L_M \equiv \{ x \in \sum^* \mid$ A DTM program M accepts x $\}$

The DTM program M halts in state q_Y when applied to x iff $x \in L_M$.

L_M is the language recognized by the DTM program M .

Definition: A DTM program M solves a decision problem π (under encoding scheme e) if :

(i) M halts for all $x \in \sum^*$.
(ii) $L_M = L(\pi,e)$

Definition: The **time** used in the computation of a DTM program M on input x is the number of steps occurring up until a **halt state** is entered.

q_N or q_Y

Definition: (time complexity)

Let a **DTM program M halt for all x** $\in \sum^*$.

Its time complexity function is :

$T_M : Z^+ \rightarrow Z^+$ such that

$T_M(n) = \max\{ m \mid M \text{ takes } m \text{ steps on } x \text{ for } |x| = n. \}$

Why "max" ?

Definition: A DTM program M is said to be a polynomial time DTM program if there exists a polynomial P such that

$$T_M(n) \leq p(n) \text{ for all } n \in Z^+$$

Definition:The class P of languages is defined :

$$P=\{ L \mid \text{there is a polynomial time DTM program } M \text{ for which } L = L_M \}$$

A decision problem π belongs to P (under encoding scheme e)

if $L(\pi,e) \in P$, i.e. ,

there is a polynomial time DTM program M that solves π

(under encoding scheme e)

Non-deterministic Computation and the class NP

Instance:

A finite set of cites , $C=\{c_1, c_2, \ldots , c_n\}$, a distance $d(c_i , c_j) \in Z^+$ for $c_i , c_j \in C$, and $B \in Z^+$, where Z^+ is the positive integers.

Question:

Is there a "tour" of all cities in C having total length no more than B ?

No known polynomial time algorithm !!!

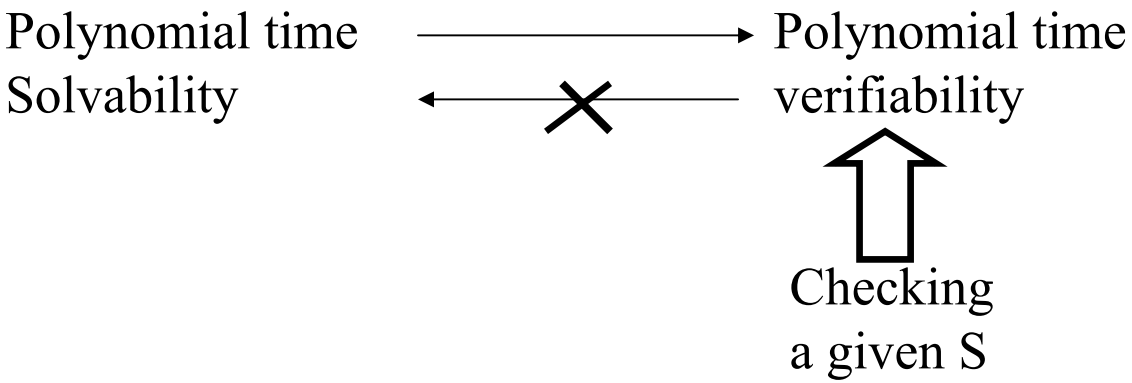
Suppose that someone claims , for a particular instance of problem , its answer is "yes".

Can you verify his/her claim ?

Yes , if he/she provides us a tour having the required properties.

⇩

"Polynomial Verifiability"



"yes" if and only if S satisfies P

Characterization of
an NP class problem

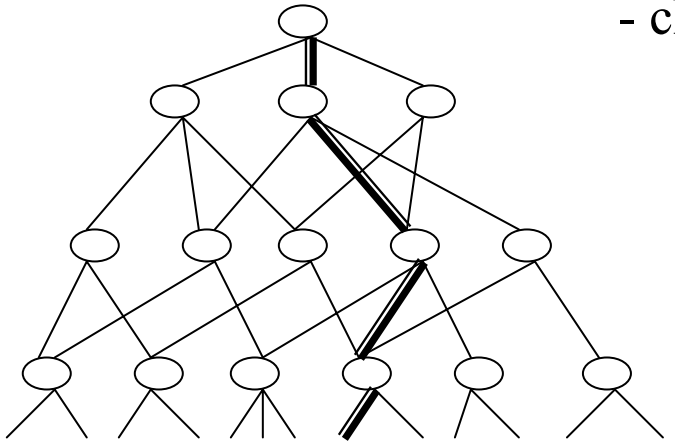
Nondeterministic Program

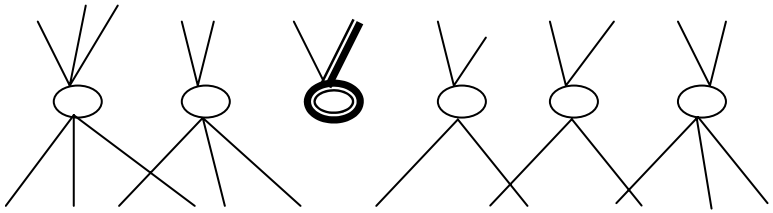
- guessing stage

  Given $I \in D_{\pi}$, , this stage guesses some structure S

- checking stage

  Given I and S , does S lead this stage to respond “yes” for I and S ?



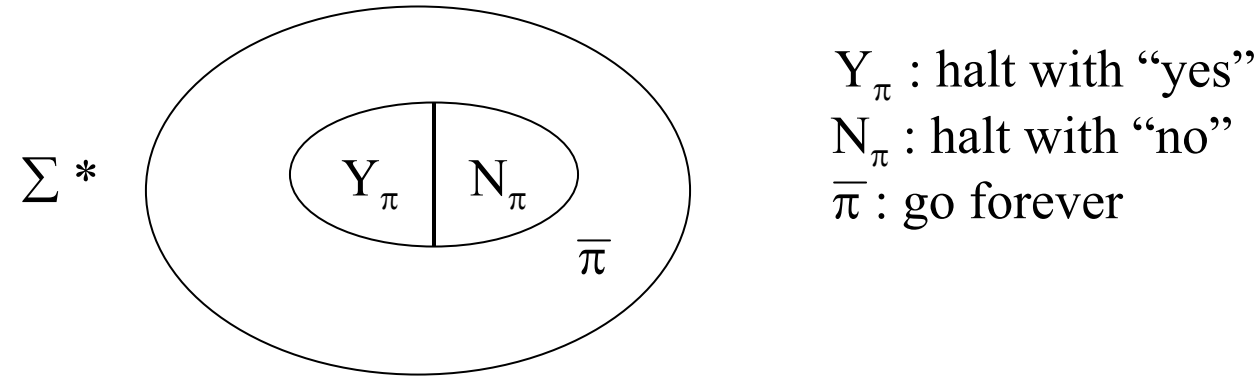


Observation

(i) S does not nec. depend on $I \in D_\pi$.
$S \in \Gamma^*$ ($\therefore$ can possibly be ε)

(ii) The checking stage behaves in a normal deterministic manner.

(iii)



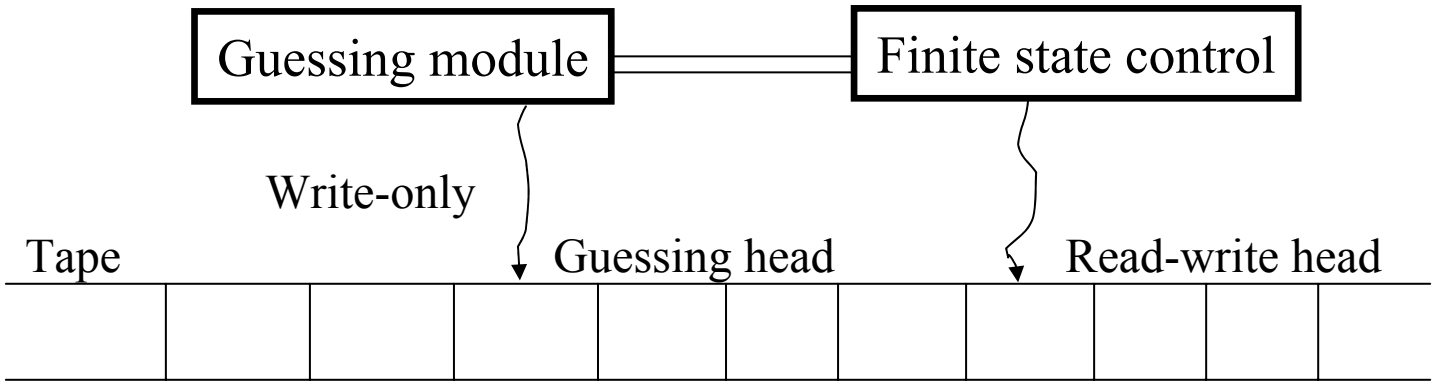
Y_π : halt with "yes"
N_π : halt with "no"
$\overline{\pi}$: go forever

An NDTM(program) M can solve a decision program π
if the following two properties hold true for $I \in D_\pi$:

(i) $I \in Y_\pi$ $\Rightarrow$ $\exists$ some S that, when guessed for I, will lead the checking stage to respond "yes" for I and S

(ii) $I \notin Y_\pi$ $\Rightarrow$ $\nexists$ any S that, when guessed for I, will lead the checking stage to respond "yes" for I and S.

Nondeterministic One-tape Turing Machine(NDTM)

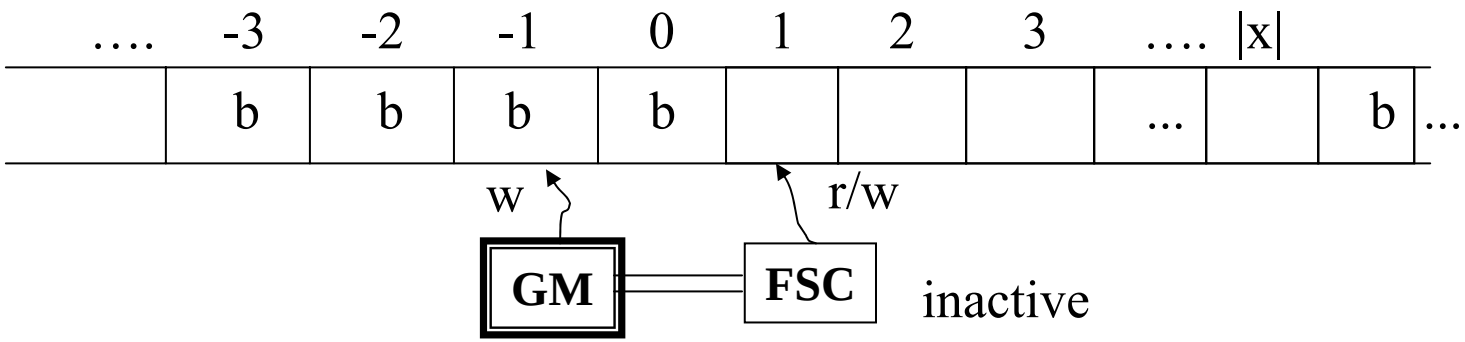


$$\delta : (Q - \{q_r , q_N\}) \times \Gamma \rightarrow Q \times \Gamma \times \{ -1 , +1 \}.$$

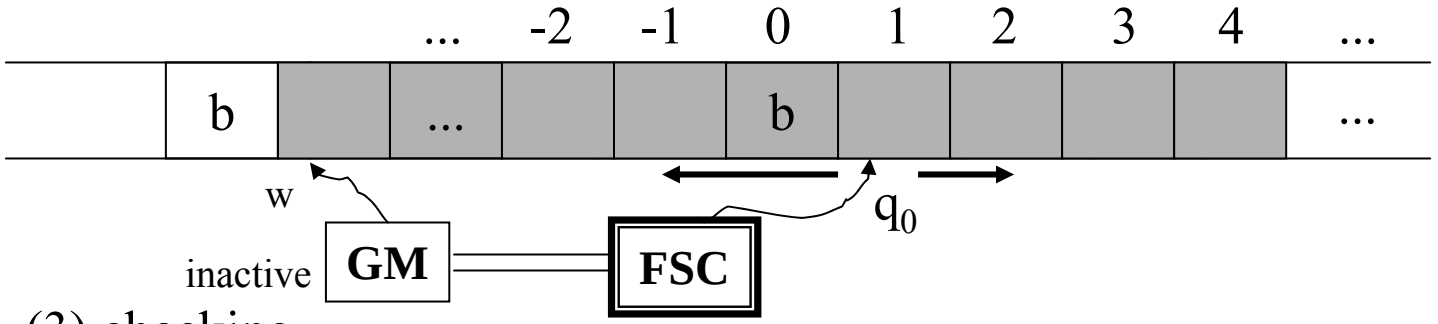
$$M = \{ Q, \Sigma , \Gamma, \delta, q_0 , b, F \}$$

NDTM program

(1) initially ,



(2) guessing ,



(3) checking ,

In the same fashion as DTM program !!!

$\Gamma, \Sigma, b, Q, q_0, q_Y, q_N$

$\delta : (Q - \{q_Y , q_N \}) \times \Gamma \rightarrow Q \times \Gamma \times \{ -1 , +1 \}.$

An NDTM program ceases computation for $x \in \sum^*$
when the finite state control enters one of two halt states , q_Y and q_N .

note

For each $x \in \sum^*$, one set of computations

Definition : A computation for $x \in \sum^*$ is said to be an accepting computation **if it halts in q_Y** . All the others , halting or not , are classed together as non-accepting computations.

Definition: An NDTM program M accepts x, if one of the computations for x is an accepting computation.

$L_M \equiv$ the language recognized by M

$L_M \equiv \{x \in \sum^* | M \text{ accepts } x\}$

Definition: The time required by an NDTM program M to accept the string $x \in L_M$ is defined to be the minimum(over all accepting computations of M for x) number of steps occurring in the guessing and checking stages up until the halt state q_y is entered.

Informally , assume the right guessing !!!

Why ?

Definition: The time complexity function

$$T_M : Z+ \rightarrow Z+$$

for an NDTM program M is :

$$T_M(n) = \max \{ \{1\} \cup \{ m \mid \text{there is an } x \in L_M \text{ with } |x|=n \text{ such that the time to accept x by M is m} \} \}$$

↑ No input of length n are accepted

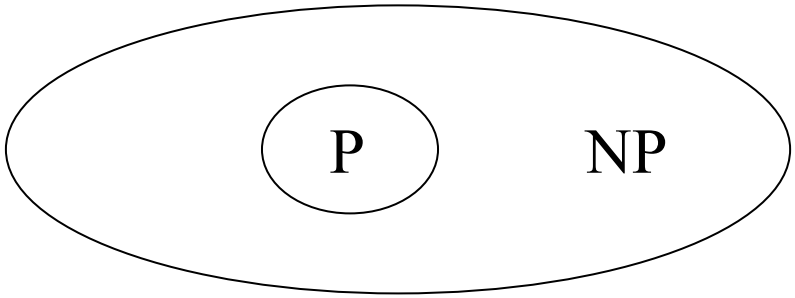
Definition : An NDTM program M is said to be a polynomial time NDTM program if there exist a polynomial p such that $T_M(n) \leq P(n)$, $n \geq 1$.

The class NP is defined :

$$NP = \{ L \mid \text{there is a polynomial time NDTM program M for which } L_M = L \}$$

Definition: A decision program π is said to belong to NP(under encoding scheme e) if $L(\pi,e) \in NP$.

Relationship between P and NP



$P \subseteq NP$

Why ?

A deterministic algorithm can be used as the checking stage of a non-deterministic algorithm !!!

No general methods converting a polynomial time NDTM algorithm to a polynomial time DTM algorithm

Theorem

$\pi \in NP \Rightarrow$ $\exists$ a polynomial P such that π can be solved by a deterministic algorithm having time complexity $O(2^{p(n)})$.

The best known result !!!

[proof] Take any π in NP. Suppose that A is a polynomial time NDTM program for solving π.

q(n) = a polynomial time bound on the time complexity of A.
(WLOG , let $q(n) \le c_1 n^{c_2}$, $c_1 , c_2 > 0$.)

$\forall$ $x \in L(\pi,e)$, there must exist some guessed string
$s \in \Gamma^*$, $|s| \le q(n)$, where $|x| = n$.

$\therefore$ # of possible guesses = $k^{q(n)}$, where $|\Gamma| = k$.
Why ?

$q(n)\, k^{q(n)} \le K^{q(n)+\varepsilon} \le c \cdot 2^{p(n)}$

Is NP = P ?

Well, ……

$P \subseteq NP$ why ?

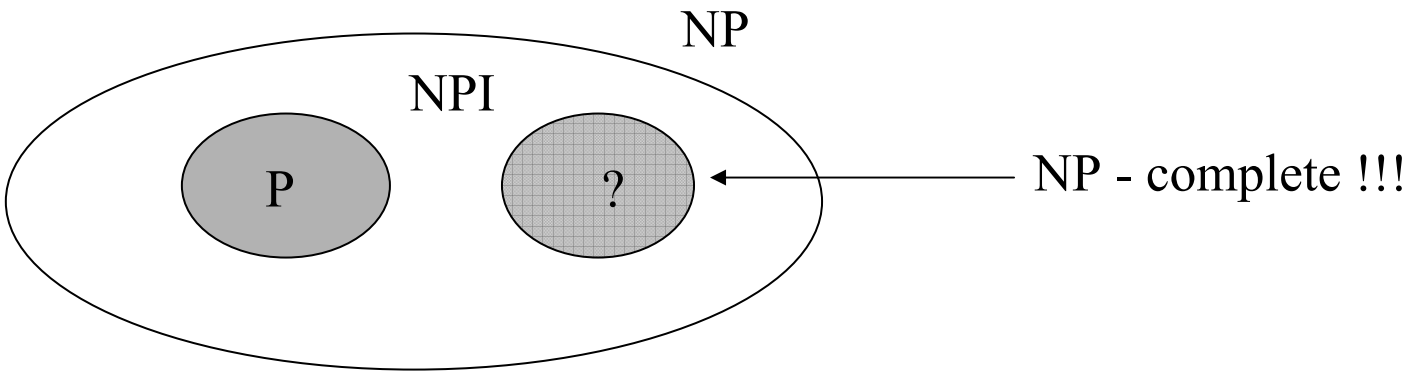
Is $NP \subseteq P$?

If "yes" then P = NP !!!

But no hope !!!

Most people believe

$P \neq NP$!!!



Polynomial Transformability and NP-Completeness

$P \subseteq NP$

However , nobody knows if $NP \subset P$.
In addition, no hope for proving $P = NP$.

$\therefore$ If $P \neq NP$, then $\exists \pi \in$ **NP - P**

This is our main focus !!!

Definition:(polynomial transformation)

A polynomial transformation from a language $L_1 \subseteq \Sigma_1^*$ to a language $L_2 \subseteq \Sigma_2^*$ is a function

$$f: \Sigma_1^* \rightarrow \Sigma_2^*$$

that satisfies the following conditions

(i) There is a polynomial **DTM program** that computes f.

(ii) For all $x \in \Sigma_1^*$, $x \in L_1 \iff f(x) \in L_2$.

Notation

$$L_1 \propto L_2$$

Lemma: If $L_1 \propto L_2$, then

$L_2 \in P \Rightarrow L_1 \in P$, and equivalently

$L_1 \notin P \Rightarrow L_2 \notin P$.

[proof] Suppose that $L_1 \propto L_2$ and $L_2 \in P$.

Let $\sum_1$ and $\sum_2$ be alphabets of L_1 and L_2 , respectively , and let

$$f : \sum_1{}^* \rightarrow \sum_2{}^*$$

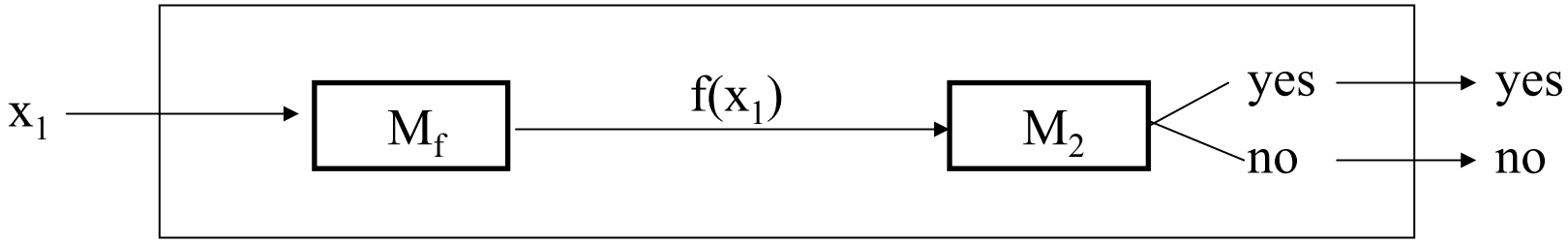
be a polynomial transform from L_1 to L_2 .

Such a transform f exists !!!

Why ?

$M_f \triangleq$ a polynomial time DTM program that computes f.

$M_2 \triangleq$ a polynomial time DTM program that recognizes L_2.



A polynomial DTM program for recognizing L_1 can be constructed by composing M_f with M_2 !!! Why ?

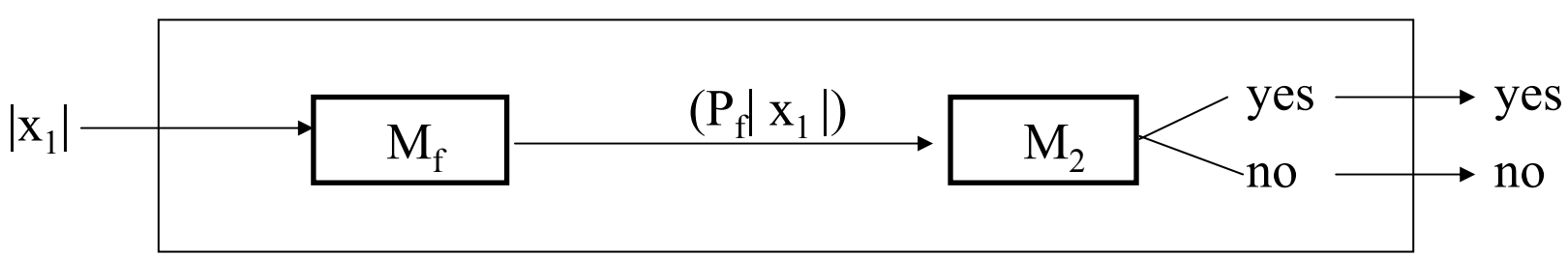
(continue)

$P_f \equiv$ a polynomial function bounding the running time of M_f

$P_2 \equiv$ a polynomial function bounding the running time of M_2

Then , $\forall \; x_1 \in \sum_1{}^*$, $|f(x_1)| \leq P_f(|x_1|)$

$$\therefore O(P_f(|x_1|) + P_2(P_f(|x_1|)))$$



note

$L(\pi_1, e_1) \propto L(\pi_2, e_2) \quad \Leftrightarrow \quad \pi_1 \propto \pi_2$!!!

$\pi_1 \propto \pi_2$ if there exists a polynomial transformation f from a decision problem π_1 to a decision problem π_2

$$f : D_{\pi_1} \rightarrow D_{\pi_2}$$

that satisfies the following two conditions:

(i) f is computable by a **polynomial time algorithm**

(ii) For all $I \in D_{\pi_1}$, $I \in Y_{\pi_1} \Leftrightarrow f(I) \in Y_{\pi_2}$

A Hamiltonian circuit in G=(V,E) is a simple circuit that includes all vertices of V.

HC(Hamiltonian Circuit)

Instance : A Graph $G = (V,E)$

Question : Does G contain a Hamiltonian circuit ?

TS(Traveling Salesman)

Instance : A finite set $C = \{c_1, c_2 , \dots , c_\pi \}$ of cities.
A distance $d(c_i , c_j) \in Z^+$ for each pair of cities $c_i , c_j \in C$, and a bound $B \in Z^+$.

Question : Is there a tour of all cities in C having total length no more than B ?

Is $HC \propto TS$?

Yes !!!

Why ?

$C := V \quad (c_i := v_i)$

$$d(c_i, c_j) := \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E \\ 2 & \text{otherwise} \end{cases}$$

$B := |V|$

$f : D_{HC} \rightarrow D_{TS}$

(i) f can be computable by a polynomial time algorithm
why ?

(ii) For all $I \in D_{HC}$, $I \in Y_{HC} \Leftrightarrow f(I) \in Y_{TS}$
why ?

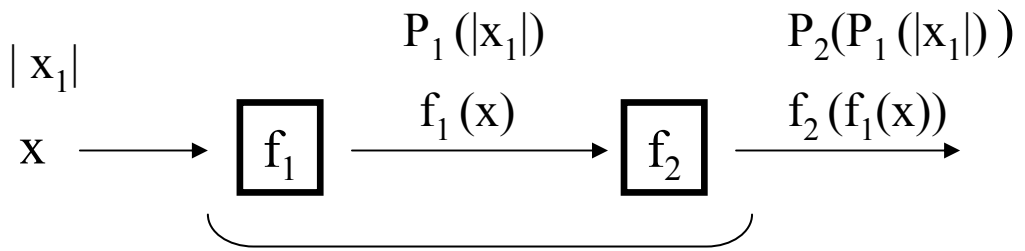
$\therefore \quad HC \propto TS$

Lemma : If $L_1 \propto L_2$ and $L_2 \propto L_3$, then $\mathbf{L_1 \propto L_3}$

[proof] Let Σ_1, Σ_2 , and Σ_3 be the alphabets of languages L_1, L_2 , and L_3 , respectively.

$f_1 : \Sigma_1^* \longrightarrow \Sigma_2^*$
a polynomial transformation from L_1 to L_2.
$f_2 : \Sigma_2^* \longrightarrow \Sigma_3^*$
a polynomial transformation from L_2 to L_3.

Then ,
$f_3 : \Sigma_1^* \longrightarrow \Sigma_3^*$ is $f_2(f_1(x))$ for all $x \in \Sigma_1^*$.



$\forall\ x \in \Sigma_1^*$, $x \in L_1 \iff f_1(x) \in L_2 \iff f_2(\ f_1\ (x)\ ) \in L_3$

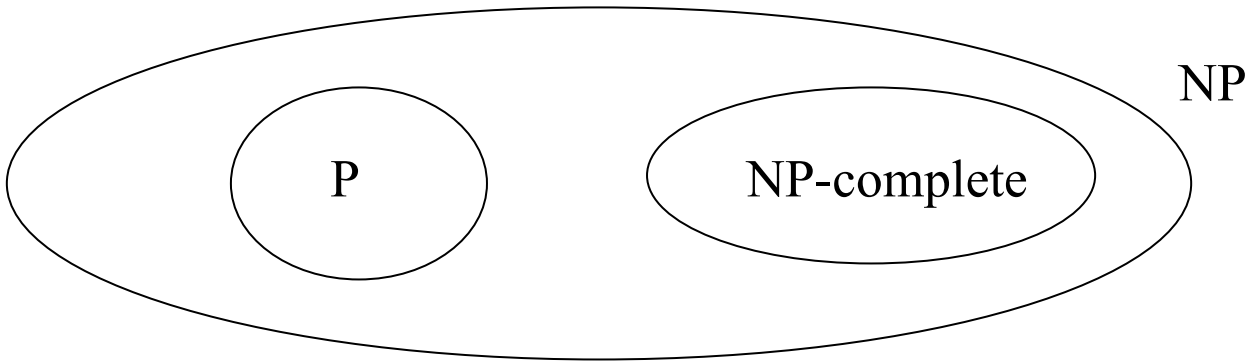
$\therefore\ x \in L_1 \iff f_2(\ f_1\ (x)\ ) \in L_3$

$\therefore\ L_1 \propto L_3$ □

Definition: Two language L_1 and L_2 (two decision problems π_1 and π_2) are said to be polynomially equivalent if

$L_1 \propto L_2$ and $L_2 \propto L_1$

($\pi_1 \propto \pi_2$ and $\pi_2 \propto \pi_1$).



P and **NP-complete** give equivalent classes !!!

Definition: A language L is defined to be NP-complete if

(i) $L \in NP$

(ii) $L' \propto L$ for all $L' \in NP$.

Lemma : If L_1 and L_2 belong to NP, L_1 is NP-complete, and $L_1 \propto L_2$ then L_2 is NP-complete

[proof] Exercise.

Given an NP-complete problem π' and a new problem π ,

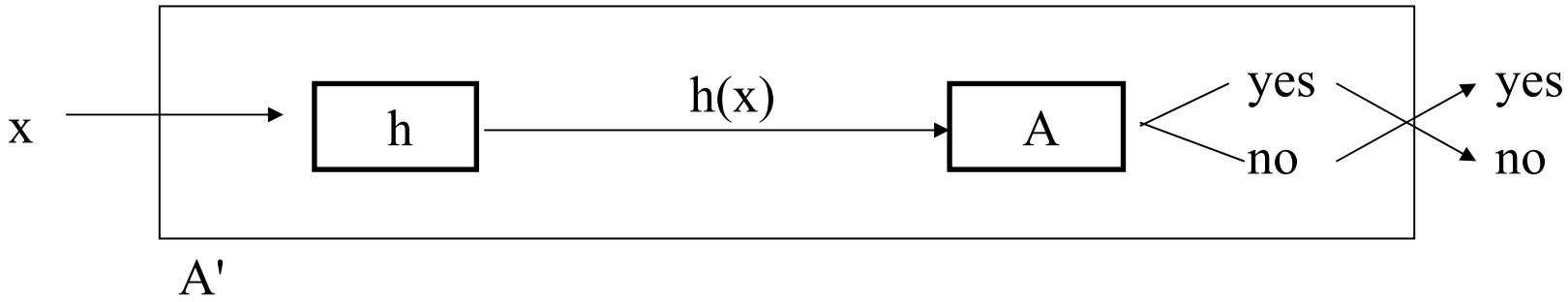
(i) $\pi \in$ NP
(ii) $\pi' \propto \pi$

$\Rightarrow$ $\pi \in$ NP-complete

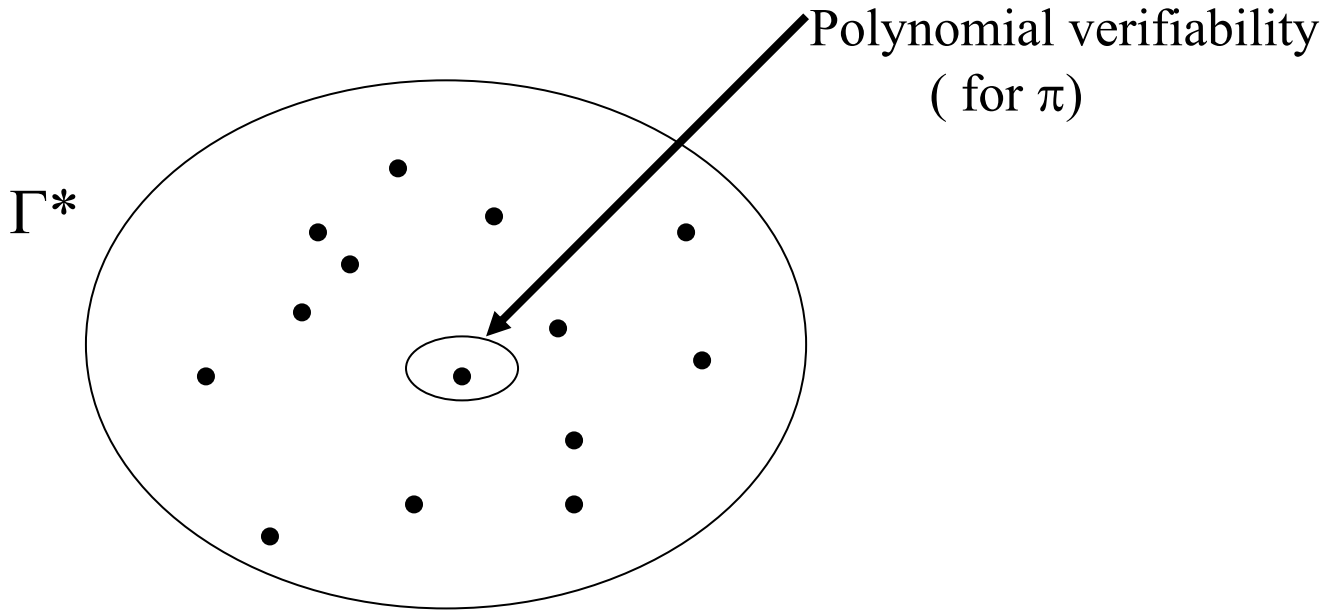
P : Given I, is X true for I ?
Co-P : Given I, is X false for I ?

Given a DTM algorithm A for solving P ,
A can be used for solving Co-P !!!

Co-P $\propto$ P



Is the same true for an NDTM algorithm ?
Well , …….



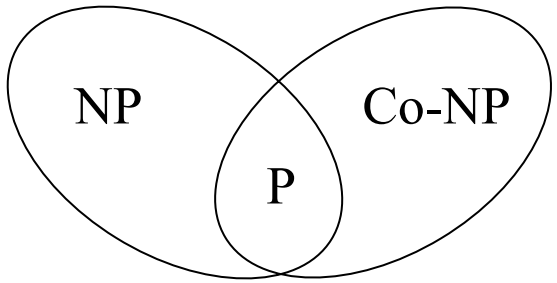
However , to answer π^c we need to examine all possible solutions !!!

TSP (Traveling Salesman)

TSP : Given a set of cities , the intercity distance and a bound B , is it
true that there is a tour of all cities having total length no more than B ?

Co-TSP : ---------- ,
Is it true that every tour of all cities has total length more than B ?

P ≠ NP



Observation

$P \propto$ Co-P

Co-P $\propto P$

$\therefore \quad P =$ Co-P

Suppose that $P = NP$.

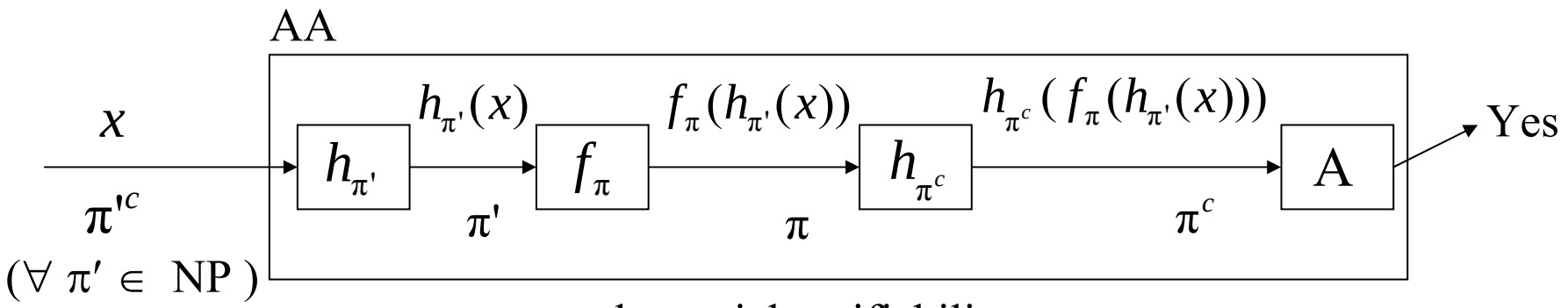
Then ,

$NP =$ Co-NP

What if $P \neq NP$?

Theorem : If there exists an NP-complete problem π such that $\pi^c \in$ NP, then NP = Co-NP.

Why ?



The NDTM program AA accepts $\pi'^c \quad \forall\ \pi' \in$ NP

$\therefore$ NP = Co-NP

SAT(satisfiability)

Instance : A set U of Boolean variables and a collection of clauses C over U.

Question : Is there a satisfying truth assignment for C ?

$$(u_1 \vee u_2 \vee \overline{u}_5) \wedge (u_2 \vee \overline{u}_3 \vee u_7 \vee \overline{u}_9) \wedge (u_4 \vee u_6) \wedge (\overline{u}_1 \vee u_7 \vee u_8 \vee u_9).$$

" wff in CNF"

$U = \{u_1, u_2, u_3, \ldots, u_9\}$

$C = \{\{u_1, u_2, \overline{u}_5\}, \{u_2, \overline{u}_3, u_7, \overline{u}_9\}, \{u_4, u_6\}, \{\overline{u}_1, u_7, u_8, u_9\}\}$

Cook's Theorem

Theorem: (Cook's Theorem)
SAT(satisfiability) is NP-complete.

[proof]

(i) $SAT \in NP$ **Why ?**

(ii) $\forall \pi \in NP \quad , \quad \pi \propto SAT$

How ?

Well, …..

$L_{SAT} \triangleq L(SAT,e)$

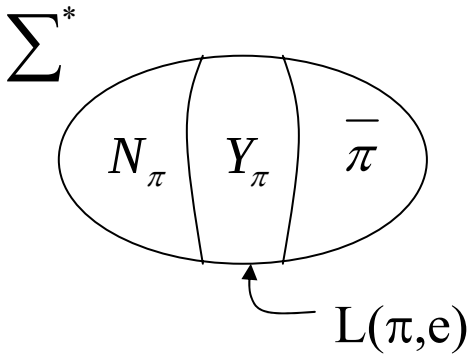
$\forall \; L \in NP \; , L \propto L_{SAT}$

There are infinitely many problems in NP !!!
How to get around this problem ?

$\forall\ \pi \in NP,\ \pi \propto SAT.$ How ?

$\pi \in NP$ $\equiv$ An polynomial time NDTM program M accepting π ,i.e., $L_M = L(\pi,e)$ $\propto$ SAT

(in polynomial time)



Describe M's moves using
the generic instance of SAT !!!

Why ?

$\pi \in NP \Leftrightarrow \exists$ an NDTM program M accepting π in p(n) time,

i.e., $L_M = L(\pi\ ,\ e)$

$f_L : \Sigma^* \rightarrow \Sigma_1^*$

$L(\pi,e) \subseteq \Sigma^*$
$L(SAT,e_1) \subseteq \Sigma_1^*$

$\forall\, x \in \Sigma^*\,,\, x \in L(\pi,e) \Leftrightarrow f_L(x) \in L(SAT,e_1)$

Let $L_M = L(\pi,e)$ for some NDTM M.
Since $M = (Q, \Sigma, \Gamma, \delta, q_0, b, F)$, f_L will be derived in terms of $P, Q, \Sigma, \Gamma, \delta, q_0, q_N, q_Y$, !!!

Basic idea

U and C are constructed so that an NDTM program M accepts x in $\sum^*$ iff $f_L(x)$ has a satisfying truth assignment.

- A generic instance of SAT consists of U and C
- $f_L(x)$?

How ?

Suppose that $x \in \sum^*$ is accepted by M.

⇩

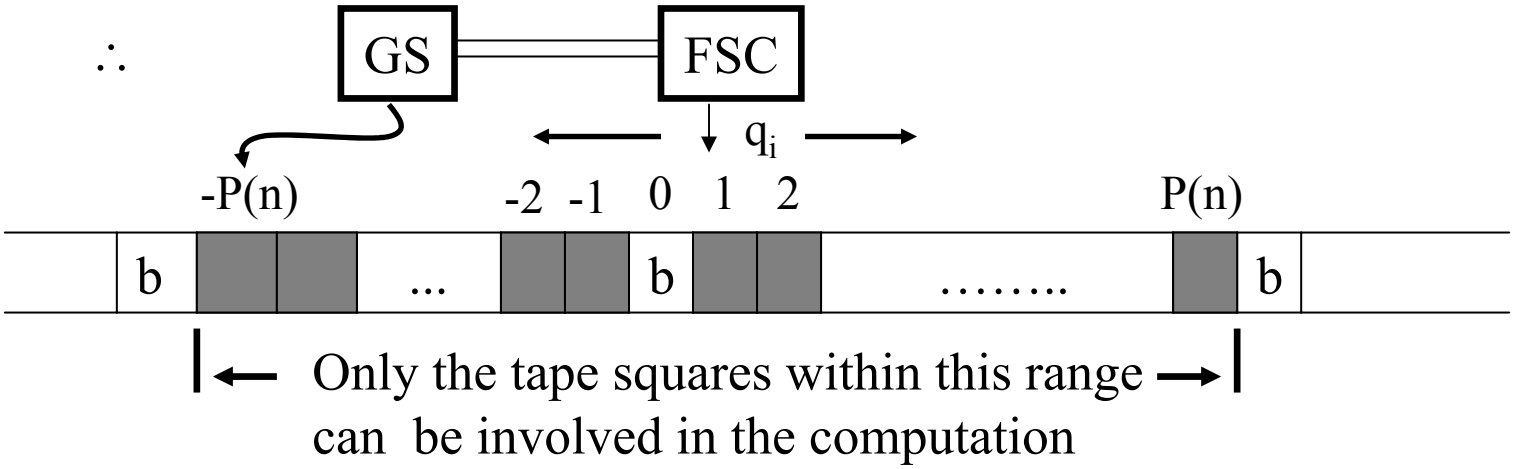
There is an accepting computation for M on x such that

(i) The number of steps in the checking stage is bounded by some polynomial P(n) , n = |x|.

(ii) The number of steps in the guessing stage is bounded by P(n) , n = |x|.

(ii) implies that the number of symbols in the guessing string is bounded by P(n), n = |x|

∴



The status of checking computation at any one time can be specified completely by giving

(i) The content of tape squares
(ii) The current state
(iii) The position of W/R head

} ID

<u>There are at most P(n) + 1 times to be considered !!!</u>

Why ?

∴ Such a computation can be described completely using only

(i) a limited number of Boolean variables

(ii) a truth assignment to them

Variable	Range	Intended meaning
Q[i,k]	$0 \le i \le p(n)$ $0 \le k \le r$	At time i (upon completion of the ith step),M is in state q_k.
H[i,j]	$0 \le i \le p(n)$ $-p(n) \le j \le p(n)+1$	At time i,the read-write head is scanning tape square j.
S[i,j,k]	$0 \le i \le p(n)$ $-p(n) \le j \le p(n)+1$ $0 \le k \le v$	At time i, the contents of tape square j is symbol s_k.

$|U| = O(P(n)^2)$

$q_0 , q_1 = q_N , q_2 = q_Y , q_3, \ldots , q_r$

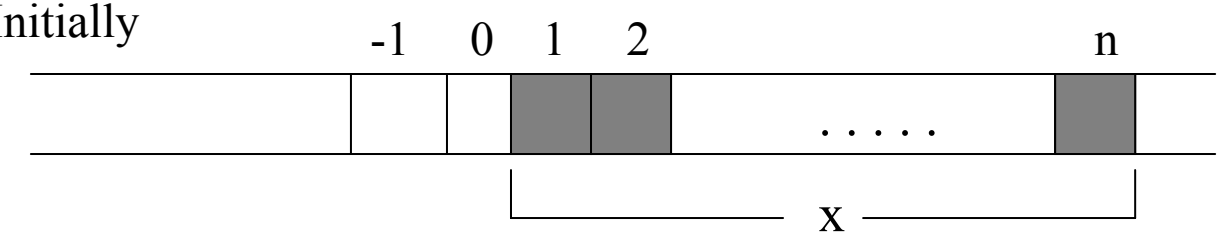
$r = |Q| - 1$

$s_0 = b , s_1 , s_2 , s_3 , \ldots , s_v$

$v = |\Gamma| - 1$

note

Initially



What if the program M halts before time P(n) ?

$x \in L_M$ $\Leftrightarrow$ there is an accepting computation of M on x with p(n) or fewer steps in its checking stage and with a guessed string w of length at most p(n)

$\Leftrightarrow$ there is a satisfying truth assignment for the collection of clauses in $f_L(x)$.

Now , using previous Boolean variables construct clauses for $f_L(x)$.

Restricting the behavior of the polynomial NDTM program M !!!

$\{ Q[i,0] , Q[i,1], \ldots , Q[i,r] \} , \ 0 \le i \le p(n)$ ← at least one state

$(P(n)+1)\ r$

$\{ \overline{Q[i,j]} , \overline{Q[i,j']} \} , \ 0 \le i \le p(n), \ 0 \le j < j' \le r$ ← no more than one state

$(P(n)+1)\,(r+1)(r/2)$

$$||Q|| = r + 1$$

Clause groups	Restriction imposed
G_1	At each time i, M is in exactly one state.
G_2	At each time i, the read-write head is scanning exactly one tape square.
G_3	At each time i, each tape square contains exactly one symbol from Γ.
G_4	At each time 0, the computation is in the initial configuration of its checking stage of input x.
G_5	By time p(n), M has entered state q_Y and hence has accepted x.
G_6	For each time i, $0 \leq i < p(n)$, the configuration of M at time i + 1 follows by a single application of the transition function δ from the configuration at time i.

Now , can you see what we are going to do ?

$$\delta : (Q - \{Q_N, Q_Y\}) \times \Gamma \rightarrow Q \times \Gamma \times \{ -1 , +1 \}$$

Clause group	Clauses in group
G_1	$\{ Q[i,0], Q[i,1], \ldots, Q[i,r] \}$, $0 \le i \le p(n)$
	$\{ \overline{Q[i,j]}, \overline{Q[i,j']}, 0 \le i \le p(n), 0 \le j < j' \le r$
G_2	$\{ H[i,-p(n)], H[i,-p(n)+1], \ldots, H[i,p(n)+1] \}$, $0 \le i \le p(n)$
	$\{ \overline{H[i,j]}, \overline{H[i,j']} \}$, $0 \le i \le p(n)$, $-p(n) \le j < j' \le p(n)+1$
G_3	$\{ S[i,j,0], S[i,j,1], \ldots, S[i,j,v] \}$, $0 \le i \le p(n)$, $-p(n) \le j \le p(n)+1$
	$\{ \overline{S[i,j,k]}, \overline{S[i,j,k']} \}$, $0 \le i \le p(n)$, $p(n) \le j \le p(n)+1$, $0 \le k < k' \le v$
G_4	$\{Q[0,0]\}$, $\{H[0,1]\}$, $\{S[0,0,0]\}$, $\{S[0,1, k_1]\}, \{S[0,2, k_2]\}, \ldots, \{S[0,n, k_n]\}$, $\{ S[0,n+1,0]\}$, $\{S[0,n+2,0]\}, \ldots, \{S[0,p(n)+1,0]\}$. Where $x = S_{k_1} S_{k_2} \cdots S_{k_n}$
G_5	$\{ Q[p(n),2] \}$

G_6:

$\{\overline{S[i,j,l]}\ ,\ H(i,j)\ ,\ S[\ i+1\ ,\ j\ ,\ l]\ \}\ ,$ $\qquad 2(p(n)+1)^2(v+1)$

$$0 \le i \le p(n)$$
$$-p(n) \le j \le p(n)+1$$
$$0 \le l \le v$$

$\{\overline{H[i,j]}\ ,\ \overline{Q[i,k]},\overline{S[i,j,l]},H[i+1,j+\Delta]\ \}$

$\{\overline{H[i,j]}\ ,\ \overline{Q[i,k]},\overline{S[i,j,l]},Q[i+1,k']\ \}$

$\{\overline{H[i,j]}\ ,\ \overline{Q[i,k]},\overline{S[i,j,l]},S[i+1,j,l']\ \}$

$$0 \le i < P(n)$$
$$-P(n) \le j \le P(n)+1$$
$$0 \le k \le r$$
$$0 \le l \le v$$
$$6(p(n)+1)^2(r+1)(v+1)$$

$$q_k \in Q\ -\ \{q_N\ ,\ q_Y\ \}$$

$$\delta(q_k\ ,\ s_l) \longrightarrow (q_{k'}\ ,\ s_{l'}\ ,\ \Delta\ )$$

$\therefore x \in L_M \Leftrightarrow$ There is an **accepting computation** of M on x of steps p(n) or less , and this computation , given the interpretation of variables , **imposes the behavior of M so that there exist a truth assignment satisfying all the clauses in**

$\mathbf{C = G_1 \cup G_2 \cup G_3 \ldots\ldots \cup G_L}$

Polynomial time for constructing $f_L(x)$!!!

Why ?

$|U|\,|C| = O(p(n)^4)$

Check it !!!

NP - Hard

Exercise