Discussion on Cook's Theorem



Cook's Theorem : Satisfiability of boolean formula is NP-complete.

How to define a problem ?

- A problem is denoted as Π and is defined as two tuple (D, S)

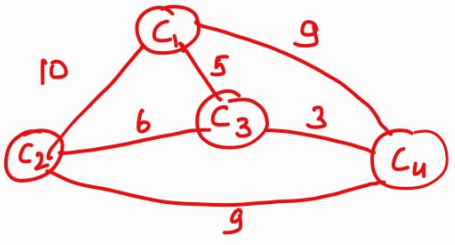
where D : description of all its parameters and

S : properties that solution must satisfy.

$\Pi : (D, S)$

problem ↑ Description ↑ Property to satisfy

- Instance of a problem : We create an instance of a problem by specifying values of all its parameters.
- Example of a problem : Traveling-salesman problem (TSP)
  - D (TSP) : A set of cities $C : \{c_1, c_2, \ldots, c_n\}$ and distance between each pair of cities i.e. $\forall c_i, c_j$ $i \neq j$ and $i, j \leq n$, $dist(c_i, c_j)$ is given.
  - S (TSP) : Find a tour a of the cities such that cost(tour) is minimized.



$D : C = \{c_1, c_2, c_3, c_4\}$

$dist : \{(c_1, c_2) = 10, (c_2, c_3) = 6, (c_3, c_4) = 3, (c_4, c_1) = 9, (c_2, c_4) = 9, (c_1, c_3) = 5\}$

S :

What is an algorithm ??

- An algorithm is a step-by-step procedure to solve a problem Π.
- An algorithm solves a problem Π (iff) it can be applied to any instance $I \in \Pi$ and it is guaranteed to give a solution for every instance $I \in \Pi$.
- For any problem, we wish find an efficient algorithm.

Efficient : Fastest and space is optimized.

Encoding scheme (e) : It's a function that maps instances of a problem to strings that computers can understand. i.e. $e : I \longrightarrow \{0,1\}^*$ $\Sigma = \{0,1\}$

encoding function; I: instance; Σ^* : strings of 0s & 1s

- Length of an input is the number of symbols used to encode I using e and Σ.

Time complexity function of an algorithm : Maximum of time taken by the algorithm for all possible input strings.

Polynomial time algorithm : Let $|I| = n$ i.e. n is the length of $e(I)$, then time complexity of a poly-time algorithm denoted as $T(n) = O(p(n))$ i.e.

$T(n) \leq c \cdot p(n)$ where c is some constant ≥ 1 and $p(n)$ polynomial function in length of input.

Exponential algorithm : $T(n) = c \cdot 2^n$.

Intractable problems : No poly-time algorithms are known till now.

Decision Problem : It has only two possible solutions 'Yes' or 'No'.

- A decision problem consists of two sets :
  1) D_Π : set of all instances
  2) $Y_\Pi \subseteq D_\Pi$: set of yes instances
- An instance belongs to D_Π (iff) it can be obtained from generic instance by substituting particular object or parameters of specific types, for all generic components.

- An instance belongs to Y_Π (iff) the answer to the stated question for this particular instance is 'YES'.

Definition of Σ^* : Let Σ be finite set of symbols. Then $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots$ = set of all finite strings over Σ.

Language : $L \subseteq \Sigma^*$

#

Decision Problem $\xleftrightarrow{\text{under encoding } e}$ Language

- Decision problem Π : $\{I_1, I_2, I_3 \ldots\ldots\ldots\}$

↓ Apply encoding e using $\Sigma = \{0,1\}$

Language L : $\{e(I_1), e(I_2), e(I_3), \ldots\ldots\}$

where $e(I_i) \in \Sigma^*$ or $\{0,1\}^*$

- Decision problem Π partitions Σ^* into three sets
  1) Those that are not encodings of instances of Π
  2) Those that are encodings of instances of Π for which the answer is 'yes' (denoted by Y_Π).
  3) Encodings of instances for which the answer is 'no'.

- Set of all $I \in \Pi$ for which the answer is 'YES' is the set of strings of Σ^* that we associate with Π and encoding e.

$\Rightarrow$ $L[\Pi, e] = \{x \in \Sigma^* \mid e(I) = x, I \in \Pi \text{ and } I \in Y_\Pi\}$

- If a result holds for $L[\Pi, e]$ then it holds for problem Π and e. i.e.

$$e(I) = x \in L \iff I \text{ is a yes instance of } \Pi$$

Deterministic Turing Machine

- Denoted as DTM is a 7-tuple $(Q, \Sigma, \gamma, \delta, q_0, F, \square)$

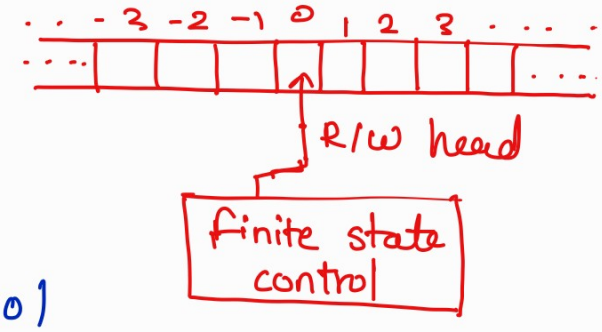
  Q : set of states, Σ : alphabet

  γ : tape alphabet, q_0 : initial state $\in Q$

  δ : transition function : $Q \times \gamma \longrightarrow Q \times \gamma \times \{L, R\}$

  F : set of final states $\subseteq Q$

  $\square$: Blank symbol



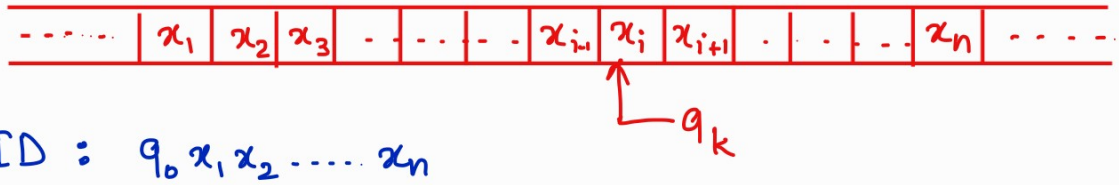
- Write the input string x from tape square 1 to $|x|$, one symbol per square. All other squares contain blank.
- Acceptance : $q = q_f$ (halt)

Program / algorithm / TM

- Language of a turing machine M :

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

- Instantaneous Description (ID)

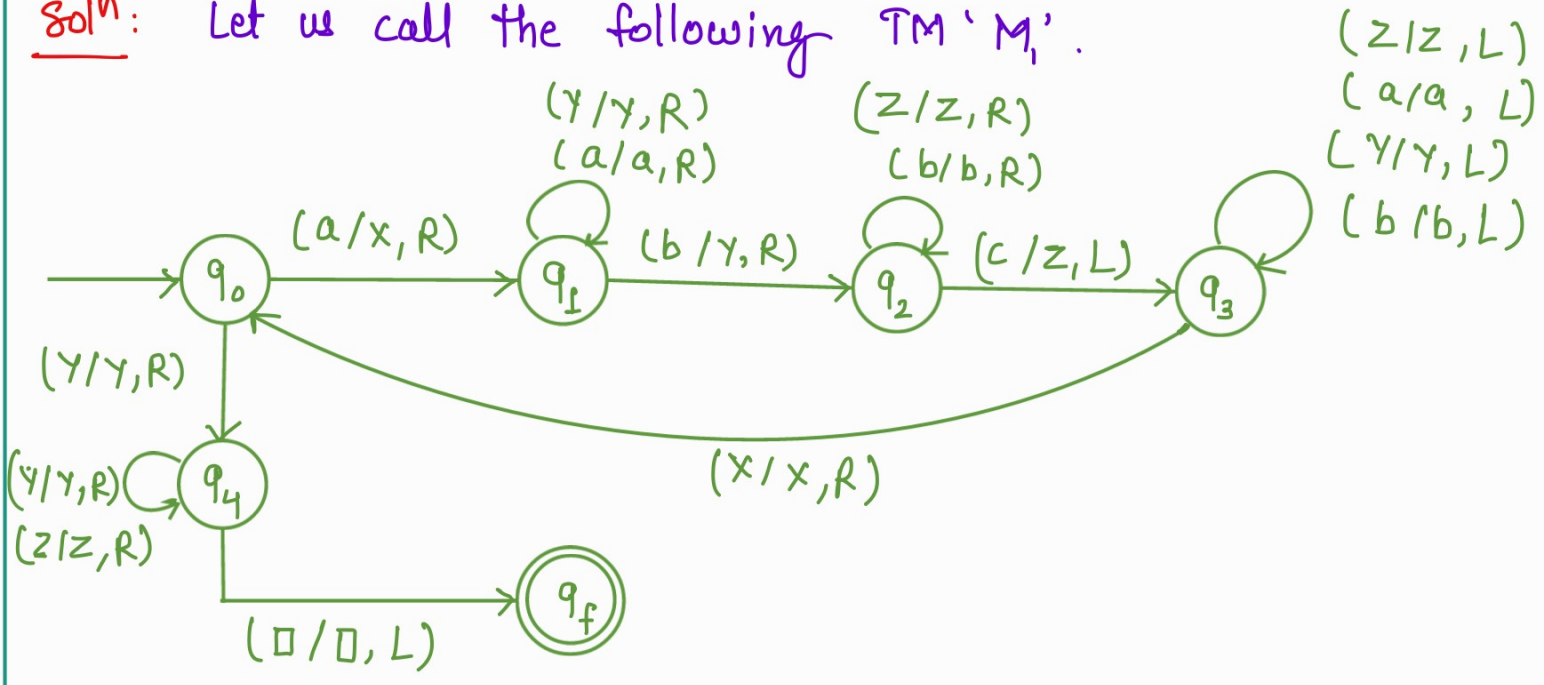


- Initial ID : $q_0 x_1 x_2 \ldots x_n$
- Accepting ID : $\alpha q_f \beta$ where $\alpha \in \gamma^*$, $\beta \in \gamma^*$ and $q_f \in F$

Example of a DTM program :

Let $L = \{a^n b^n c^n \mid n \geq 1\}$ over $\Sigma = \{a, b, c\}$. Design a deterministic turing machine for L.

Soln: Let us call the following TM 'M_1'.



- How does machine M_1 behaves with input strings

Example 1: Input $x = abc$

$q_0abc \vdash Xq_1bc \vdash XYq_2c \vdash Xq_3YZ \vdash q_3XYZ \vdash Xq_0YZ \vdash XYq_4Z \vdash XYZq_4\square \vdash XYq_fZ$

(halt at q_f. Hence accepted) $\Rightarrow abc \in Lang(M_1)$.

Example 2: Input $x = abbc$

$q_0abbc \vdash Xq_1bbc \vdash XYq_2bc \vdash XYbq_2c \vdash XYq_3bZ \vdash Xq_3YbZ \vdash q_3XYbZ \vdash Xq_0YbZ \vdash XYq_4bZ$

(halt at q_4 because $\delta(q_4, b)$ not defined. Hence rejected.
$\Rightarrow abbc \notin Lang(M_1)$)

Note: ① program $\Longleftrightarrow$ Algorithm $\Longleftrightarrow$ Turing Machine

② Solving a decision problem $\Longleftrightarrow$ Recognizing a language by a Turing Machine.

We say that a DTM program M solves the decision problem Π under encoding scheme e

(iff)

M halts for all input strings over its input alphabet and

$Lang(M) = L[\Pi, e]$. i.e. set of all yes instances.

Time complexity of a DTM 'M':

$$T_M(n) = \max \left\{ m \;\middle|\; \begin{array}{l} \text{there is an } x \in \Sigma^* \text{ with } |x| = n, \\ \text{such that the computation of } M \text{ on} \\ x \text{ takes time } m. \end{array} \right.$$

- Computation of M on x means, steps occurring in that computation until a halt state is entered.

- Polynomial-time DTM

  A DTM program M is called a poly-time DTM (iff) for all $n \in Z^+$

  $T_M(n) \leq p(n)$ where p is a polynomial

- Class P of languages or problems

$P = \{L \mid$ there is a poly-time DTM program M for which $L = Lang(M)\}$

i.e. A decision problem Π belongs to P under the encoding e

(iff)

$L[\Pi, e] \in P$

(iff)

There is a poly-time DTM program that solves Π under encoding scheme e.

⇒ Poly-time algorithm $\longleftrightarrow$ poly-time DTM

Polynomial-time verification

Given an instance and a certificate, we can verify whether the instance is a yes instance in polynomial time.

Non-deterministic algorithm for poly-time verification

- stage 1 : Guessing stage - Given a problem instance I, algo. first guesses a structure S.
- Stage 2 : Checking stage - We provide (I, S) to the checking stage which proceeds to compute in a normal deterministic manner with 3 possible outcomes :
  - Either eventually halting with answer "YES"
  - Either eventually halting with answer "NO"
  - Computing forever without halting.

When does a non-deterministic algorithm solves a decision problem??

$\forall$ instances $I \in D_\Pi$, the following two properties hold :

i) If $I \in Y_\Pi$, the there exists some structure S that, when guessed for input I, will lead the checking stage to respond "YES" for I and S.

ii) If $I \notin Y_\Pi$, then there exists no structure S that, when guessed for input I, will lead the checking stage respond "YES" for I and S.

Example : TSP

Polynomial time non-deterministic algorithm

A non-deterministic algorithm that solves a decision problem Π is said to operate in poly-time

iff

there exists a polynomial p such that, for every instance $I \in Y_\Pi$, there is some guess S that leads the deterministic checking stage to respond "YES" of (I, S) within $p(|I|)$.

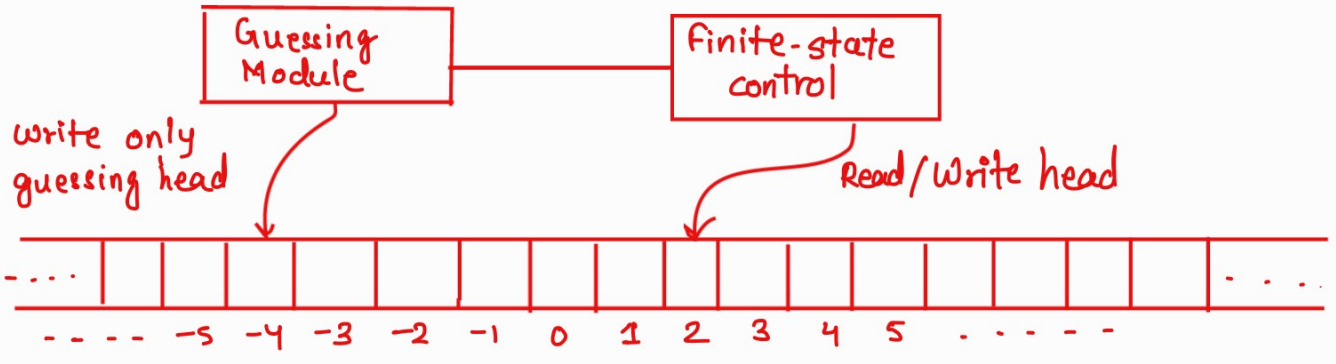
Class NP : Set of all decision problems Π that, under reasonable encoding scheme, can be solved by a polynomial time non-deterministic algorithm.

How to get a non-deterministic algorithm ??

Non-deterministic Turing Machine (NDTM)

An NDTM is defined as a 7-tuple $(Q, \Sigma, \gamma, \delta, q_0, F, \square)$ where the components are same as DTM.

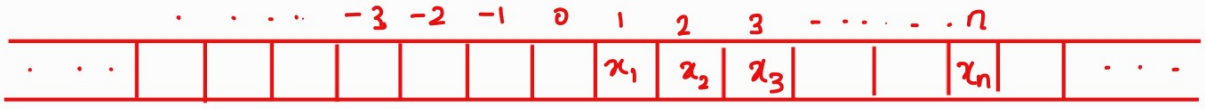
- NDTM = DTM + a guessing module having its own write-only (WO) head



- The guessing module provides a mean for writing down a "guess".
- Computation of NDTM : Stage 1 : (Guessing stage)
  Stage 2 : (Checking stage)

Stage 1: Guessing stage

(i) Initially the input string x is written in the tape square 1 through $|x|$, other squares are blank.



$x = x_1 x_2 x_3 \ldots x_n \quad |x| = n$

(ii) The read-write head is scanning tape-square 1.

(iii) The write-only head is scanning square -1.

(iv) Finite state control is inactive.

(v) The guessing module then directs the wo head, one step at a time

- either to write some symbol from γ in the tape square being scanned, and move one square to left
- or to stop at which point the guessing module becomes inactive and finite state control is activated in state q_0.

(vi) The choice of whether to remain active and if so, which symbol to write from γ, is made by guessing module in a totally arbitrary manner.

(vii) Thus the guessing module can write any string from γ^* before it halts or never halts.

Stage 2: Checking stage :

(i) The checking stage begins when the finite state control is activated in state q_0.

(ii) From this point onward, the computation proceeds solely under the direction of NDTM program according to same rule as DTM.

(iii) The computation ceases if

the finite state control enters q_f or no move found on some symbol.

(iv) A computation is said to be an accepting computation iff it halts in q_f.

- All other computations (non-final state halting, infinite loop) are non-accepting computations.

When does an NDTM accepts a string ?

- An NDTM program accepts a string $x \in \Sigma^*$ iff there is at least one accepting computation.
- $Lang(M) = \{x \in \Sigma^* \mid M \text{ accepts } x\}$

Time required by an NDTM :

The time required by an NDTM program M to accept a string $x \in Lang(M)$ is the minimum over all accepting computations of M on x, of the number of steps occurring in the guessing and checking stage until the halt state $q_f \in F$ is entered.

Time complexity of an NDTM :

$$T_M(n) = \max \left\{ \{1\} \cup \left\{ m \;\middle|\; \begin{array}{l} \text{there is an } x \in Lang(M) \text{ with} \\ |x| = n \text{ such that the time} \\ \text{to accept } x \text{ by } M \text{ is } m. \end{array} \right\} \right\}$$

Polynomial time NDTM

The NDTM program M is poly-time NDTM iff there exists a polynomial p such that

$$T_M(n) \leq p(n), \quad \text{for all } n \geq 1$$

$NP = \{ L \mid$ there is a poly-time NDTM program M for which $Lang(M) = L \}$

Relationship between P and NP -

① $\Pi \in P \implies \Pi \in NP$

Proof : $\Pi \in P \Rightarrow$ there is a polynomial time DTM say A for Π. We can obtain a poly-time NDTM for Π by using A as the checking stage and ignoring the guessing stage.

$\Rightarrow \Pi \in NP$

② If $\Pi \in NP$, then $\exists$ a polynomial p such that Π can be solved by a DTM algorithm having time complexity $O(2^{p(n)})$.

Polynomial time transformation

A polynomial time transformation from a language $L_1 \in \Sigma_1^*$ to a language $L_2 \in \Sigma_2^*$ is a function $f : \Sigma_1^* \longrightarrow \Sigma_2^*$ that satisfies the following conditions :

(i) There is a poly-time DTM program that compute f.

(ii) $\forall x \in \Sigma_1^*$, we have

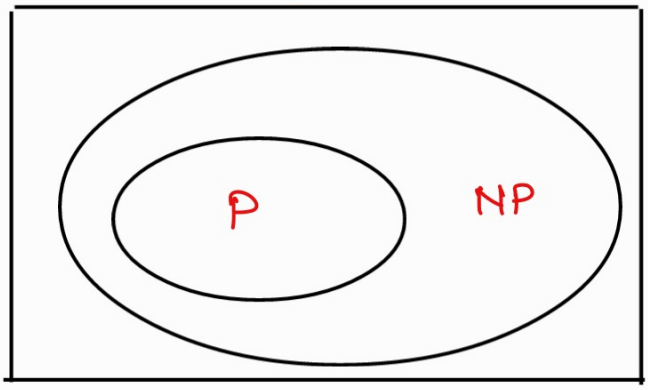
$x \in L_1$ iff $f(x) \in L_2$

Notation : If there is a poly-time transformation from L_1 to L_2, we write $L_1 \leq_P L_2$ (L_1 transforms to L_2)

Lemma : Let $L_1 \leq_P L_2$. Then

① If $L_2 \in P$ then $L_1 \in P$

② If $L_1 \notin P$ then $L_2 \notin P$



Note : (1) Let π_1 and π_2 be two decision problems, with associated encodings e_1 and e_2 respectively. Then, $\pi_1 \leq_p \pi_2$ with respect to given encoding schemes

iff

there exists a polynomial transformation from

$L[\pi_1, e_1]$ to $L[\pi_2, e_2]$

Note : (2) We regard a polynomial time transformation from the decision problem π_1 to the decision problem π_2 as a function

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

that satisfies the two conditions :

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

(ii) For all $I \in D_{\pi_1}$, we have

$I \in Y_{\pi_1}$ iff $f(I) \in Y_{\pi_2}$

NP-complete (Language level)

A language L is NP-complete iff

(i) $L \in NP$

(ii) $\forall L' \in NP$, we have $L' \leq_p L$

NP-complete (Problem level)

A decision problem π is NP-complete iff

(i) $\pi \in NP$

(ii) $\forall \pi' \in NP$, we have $\pi' \leq_p \pi$

Lemma : Let $L_1, L_2 \in NP$. Let L_1 is NP-complete and $L_1 \leq L_2$. Then L_2 is NP-complete.

How to prove that a decision problem π is NP-complete?

Some notations :

- Let $U = \{u_1, u_2, \ldots, u_n\}$ be a set of Boolean variables.
- Truth assignment : $t : U \longrightarrow \{T, F\}$
- Literal : If u is a variable in U, then u and $\bar{u}$ are literals over U.
- Observations :

(1) [The literal u is true under t] (iff) [the variable u is true under t]

(2) [The literal $\bar{u}$ is true under t] (iff) [the variable u is false under t]

- Clause : A clause over U is a set of literals over U. A clause represents the disjunction of those literals.
  - A clause is satisfied by a truth assignment iff at least one literal of the clause is true.

Example : Let $U = \{u_i \mid i \in [1, 9]\}$. Let clause $C_1 = \{u_1, \bar{u}_3, u_8\}$ i.e. $C_1 = u_1 \vee \bar{u}_3 \vee u_8$. The clause will be satisfied by t unless $t : u_1 \longrightarrow F$, $u_3 \longrightarrow T$, $u_8 \longrightarrow F$

u_1	u_3	u_8	$u_1 \vee \bar{u}_3 \vee u_8$
0	0	0	$0 \vee 1 \vee 0 \equiv 1$
0	0	1	$0 \vee 1 \vee 1 \equiv 1$
0	1	0	$0 \vee 0 \vee 0 \equiv 0$
0	1	1	$0 \vee 0 \vee 1 \equiv 1$
1	0	0	$1 \vee 1 \vee 0 \equiv 1$
1	0	1	$1 \vee 1 \vee 1 \equiv 1$
1	1	0	$1 \vee 0 \vee 0 \equiv 1$
1	1	1	$1 \vee 0 \vee 1 \equiv 1$

Satisfiable formula : A collection of clauses ($\mathcal{C}$) over U is satisfiable iff there exists some truth assignment for U that simultaneously satisfies all clauses in C.

Satisfying truth assigment :

Assume $\mathcal{C}$ is a collection of clauses. Let $\mathcal{C}$ is satisfiable by some truth assignment t. Then t is called satisfying truth assignment for $\mathcal{C}$.

Problem statement (satisfiability)

- Instance : A set U of variables and a collection $\mathcal{C}$ of clauses over U.
- Question : Is there a satisfying truth assignment for $\mathcal{C}$?

Example (1) $U = \{u_1, u_2\}$

$\mathcal{C}_1 = \{\{u_1, u_2\}, \{u_1, \overline{u_2}\}, \{u_1\}\} = (u_1 \vee u_2) \wedge (u_1 \vee \overline{u_2}) \wedge \underline{u_1}$

Answer : No.

Example (2) $U = \{u_1, u_2\}$

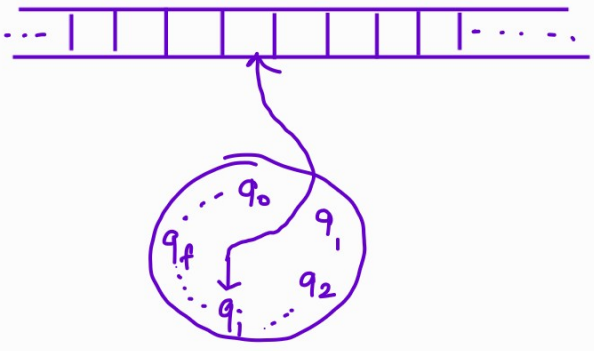
$\mathcal{C}_2 = \{\{u_1, \overline{u_2}\}, \{\overline{u_1}, u_2\}\} = (u_1 \vee \overline{u_2}) \wedge (\overline{u_1} \vee u_2)$

Answer : Yes

Cook's Theorem : Satisfiability of boolean formula is NP-complete.

Proof : Assumption

Turing machine - Basic Turing Machine (2-way ∞ tape)



other models → 1-way ∞ tape
→ multi-track
→ multi-tape
→ counter m/c
→ etc.

Turing machine (M) : $(Q, \Sigma, \gamma, \delta, q_0, F, \square)$ where

i) Q : a set of states

ii) Σ : input alphabet

iii) γ : tape alphabet

iv) δ : partial transition function

$$\delta : Q \times \gamma \longrightarrow Q \times \gamma \times \{L, R\}$$

v) q_0 : initial state $\quad q_0 \in Q$

vi) F : a set of finite state $\quad F \subseteq Q$

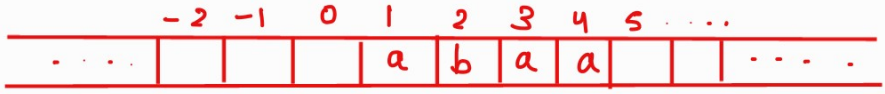
vii) $\square$: blank symbol $\quad \square \in \gamma$

Assumptions : ① $Q = \{q_0, q_1 = q_f, q_3, q_4, \ldots\ldots, q_r\}$

$$|Q| = r + 1$$

② $\gamma = \{s_0 = \square, s_1, s_2, \ldots\ldots, s_v\}$

$$|\gamma| = v + 1$$



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

$\delta : (q_k, s_l) \longrightarrow (q_{k'}, s_{l'}, \Delta)$ where $\Delta \in \{L, R\}$

- How to capture or represent δ function using boolean variables?

At time i, what is the configuration of TM

↓ apply δ

At time $i+1$, what is the configuration of TM

Question 1 : At time i what is the state of TM ?

Question 2 : At time i, which cell is being scanned ?

Question 3 : At time i, which symbol is being scanned ?

Question	What do we get
1	Which state
2	Where is R/w head
3	What is the symbol being scanned

Question	How to capture using boolean variables
1	$Q[i,k]$ = at time i, the TM is in state q_k
2	$H[i,j]$ = at time i, the R/w head is at cell j
3	$S[i,j,k]$ = at time i, the content of cell j is s_l
we get	$\delta(q_k, s_l)$

Goal : To simulate the TM, for any input according the transition function, in a systematic way.

We want to capture the following situations of TM in a systematic way

SW1] Represent the prototype of a generic transition function using boolean variables

SW2] Represent the starting configuration of TM using boolean variables.

SW3] Represent the finishing configuration of TM using boolean variables.

SW4] Represent the specific transition function using our ingredients.

SW1] Represent the prototype of generic transition function

(a) At each time i, the TM is in exactly one state.

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

Q: defining component 1 of the cartesian product

(b) At each time i, the read-write head is scanning exactly one tape square

(c) At each time i, each tape square contains exactly one symbol from Γ

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

Γ: defining component 2 of cartesian product

sw 2 : Represent the starting configuration of the TM using boolean variables.

- At time 0, the TM is in the initial configuration of its checking stage for input x.

sw 3 : Represent the finishing configuration of TM1 using boolean variables

- By time $p(n)$, the TM has entered state q_f and hence accepted x.

sw 4] Represent the specific transition function using boolean variables.

- for each time i, $0 \le i \le p(n)$, the configuration of TM at time $(i+1)$, is followed by a single application of the transition function δ, from the configuration at time i.

Next Goal : We want to represent sw1(a), sw1(b), sw1(c), sw(2), sw(3) and sw(4) using the three boolean variables.

Formula for sw1(a)

- Observation 1 : At time i, the TM is in exactly one state iff
  - i) At time i, the TM is in at least one state, and
  - ii) At time i, the TM is in at most one state.

- Recall : $|Q| = r+1$ = # of states of the TM.

- Observation (2) : At any time i, the TM is in at least one state

(iff)

$$Q[i,0] \vee Q[i,1] \vee Q[i,2] \vee \dots\dots \vee Q[i,r] = 1$$

(iff)

$$\bigvee_{a=0}^{r} Q[i,a] \quad \text{is a satisfiable clause}$$

SW 1 (a) (i) done

Claim : Let $M = (Q, \Sigma, \gamma, \delta, q_0, F, \square)$ be any basic TM such that $|Q| = r+1$. Let the boolean variable $Q[i,k]$ denotes the fact that at i^{th} time of the computation, the TM is is in state q_k.

Then at time i, the TM is in at most on state

(iff)

$$\bigvee_{a=\{k,k'\}} \neg Q[i,a] \quad \text{is a satisfiable clause } \forall\, 0 \le k < k' \le r.$$

Proof of the claim : Consider time i. There are $|Q| = r+1$ states. Without loss of generality, we can assume any total order on the states because this will never affect the language of the TM.

Let $q_0, q_1, q_2, \dots\dots, q_{k-1}, q_k, q_{k+1}, \dots\dots, q_{k'-1}, q_{k'}, q_{k'+1}, \dots, q_r$ be any such total order.

Then for any $0 \le k < k' \le r$, we have,

The TM is in at most 1 state

iff

The TM is in 0 state or 1 state

iff

if (the TM is in state q_k) then (the TM is not in state $q_{k'}$ where $k'>k$)

$p = Q[i,k]$ $\quad$ $q = Q[i,k']$

iff

$Q[i,k] \longrightarrow \neg Q[i,k']$ is true

iff

$\neg Q[i,k] \vee \neg Q[i,k']$ is true or satisfiable.

iff

$\bigvee \neg Q[i,a]$ is a satisfiable clause

$a = \{k, k'\}$

SW(1) (a) (ii) proved

We formed a boolean formula which is satisfiable iff the TM is in exactly one state

Systematic way	Our status
SW 1(a) (i)	Done
SW 1 (a) (ii)	
SW 1 (b)	Remaining
SW 1 (c)	
SW 2	
SW 3	
SW 4	

What we proved ?

- Let $G_i^{1,1} = \bigvee_{a=0}^{\gamma} Q[i,a]$, $\forall\ 0 \le i \le p(n)$

- Let $G_i^{1,2} = \bigvee_{a=\{k,k'\}}^{a=0} \neg Q[i,a]$, $\forall\ 0 \le i \le p(n)$, $\forall\ 0 \le k < k' \le \gamma$

- Then $G_i^{1,1}$ consists of $p(n)+1$ clauses and $G_i^{1,2}$ consists of $\frac{(p(n)+1)(\gamma+1)\gamma}{2}$ clauses. And we have,

At each time i, the TM is in exactly one state iff $G_i^{1,1}$ and $G_i^{1,2}$ can are satisfied simultaneously.

i.e. $\left(G_i^{1,1} \wedge G_i^{1,2} \right)$ is a satisfiable clause.

- Observation 3 : At time i the read write head is scanning exactly one tape square

iff

i) At time i, the R/W head is scanning at least one tape square, and

ii) At time i, the R/W head is scanning at most one tape square.

- Observation 4: At time i, the R/W head is scanning at least one tape square

iff

$$H[i,-p(n)] \vee H[i,-p(n)+1] \vee \ldots\ldots \vee H[i,p(n)+1] = 1$$

iff

$$\bigvee_{a=0}^{2p(n)+1} H[i,-p(n)+a] \text{ is a satisfiable clause.}$$

SW1(b)ii) done.

Claim : Let M be any basic TM. Let the boolean variable $H[i,j]$ denotes the fact that at time i of the computation, the read/write head is scanning tape square j.

Then, at time i, the TM is scanning at most one tape square

iff

$$\bigvee_{a=\{j,j'\}} \neg H[i,a] \text{ is a satisfiable clause.}$$

$$\forall\ 0 \le i \le p(n) \quad \text{and} \quad -p(n) \le j < j' < p(n)+1$$

- **Proof of the claim :** Consider time i.
  - Recall : $p(n)$ is a polynomial over the integers that bounds the time complexity function $T_M(n)$.
  - Recall : for input $x \in \Sigma^*$ with $|x| = n$, the computation can not involve any tape squares except for the numbers $-p(n)$ to $p(n)+1$, since the R/W head begins at square 1 and moves atmost one square in any single step.



Then, we have at time i,

the R/W head is scanning at most 1 tape square

iff

if (the R/W head is scanning tape square j) then (the R/W head is not scanning tape square $j' \neq j$)

iff

$$H[i,j] \longrightarrow \neg H[i,j'] \quad , \quad -p(n) \leq j < j' \leq p(n)+1$$

iff

$\neg H[i,j] \vee \neg H[i,j']$ is a satisfiable clause

iff

$\bigvee \neg H[i,a]$ is a satisfiable clause.

$a = \{j, j'\}$

SW 1 (b) (ii) Proved.

What we proved ??

- Let $G_i^{2,1} = \bigvee_{a=0}^{2p(n)+1} H[i, -p(n)+a]$, for all $0 \le i \le p(n)$, for all $-p(n) \le j < j' \le p(n)+1$

- Let $G_i^{2,2} = \bigvee_{a=\{j,j'\}} \neg H[i,a]$, $\forall\ 0 \le i \le p(n)$, $\forall\ 0 \le j < j' \le p(n)+1$

- Then $G_i^{2,1}$ consists of $p(n)+1$ clauses.
- $G_i^{2,2}$ consists of $\frac{p(n)\,(p(n)+1)^2}{2}$ clauses. And we have,

at each time i, the TM is scanning exactly one tape square

iff

$G_i^{2,1}$ and $G_i^{2,2}$ are satisfied simultaneously.

- Observation 5: At time i, each tape square contains exactly one symbol from Γ

iff

(i) Each tape square contains at least one symbol

(ii) Each tape square contains at most one symbol

- Observation 6: At any time i, each tape square contains at least one tape symbol

iff

$$S[i,j,0] \vee S[i,j,1] \vee \ldots \ldots \vee S[i,j,v] = 1$$

iff

$$\bigvee_{a=0}^{v} S[i,j,a] \quad \text{is a satisfiable clause.}$$

swl(c)(i)

Done.

- Claim: Let M be any basic TM such that $|\Gamma| = (v+1)$. Let $S[i,j,l]$: at time i of the computation, the content of the tape square j is s_l.

Then at time i, each tape square contains atmost one symbol from Γ

iff

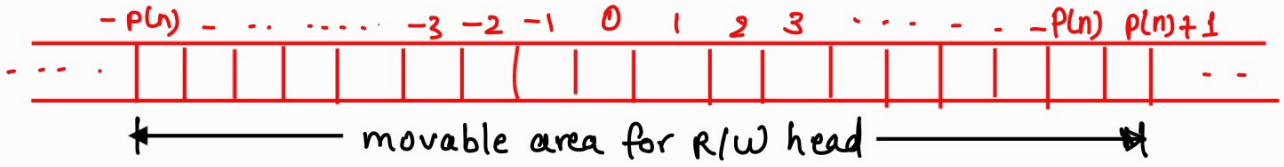
$$\bigvee_{a=\{l,l'\}} \neg S[i,j,a] \quad \text{is a satisfiable clause.}$$

$\forall\ 0 \le i \le p(n)$, $\forall\ -p(n) \le j \le p(n)+1$, $\forall\ 0 \le l < l' \le v$.

• Proof of the claim : Consider time i,

Recall : $p(n)$ is a polynomial over integers that bound $T_M(n)$.

Recall : for input $x \in \Sigma^*$ with $|x| = n$, a computation can not involve tape squares except for those numbered $-p(n)$ to $p(n)+1$, since the R/W head begins at square 1 and moves at most one square in any single step.



Then, we have that, at time i,

each tape square contains at most one symbol from τ

iff

if (at square j, the symbol is s_l) then (at square j, the symbol can not be $s_{l'}$, $l' \neq l$.)

iff

$$S[i,j,l] \longrightarrow \neg S[i,j,l'], \quad \forall\, 0 \le i \le p(n), \ \forall\, -p(n) \le j \le p(n)+1, \ \forall\, 0 \le l < l' \le v$$

iff

$\neg S[i,j,l] \vee \neg S[i,j,l']$ is a satisfiable clause

iff

$\bigvee \neg S[i,j,a]$ is a satisfiable clause

$a = \{l, l'\}$

sw1(c) (ii) done

Where are we standing now?

Systematic way	Our status
sw 1(a)(i)	Done
sw 1(a)(ii)	
sw 1(b)	
sw 1(c)	
sw 2	Remaining
sw 3	
sw 4	

What we just proved?

- Let $G_i^{3,1} = \bigvee_{a=0}^{v} s[i,j,a]$, $\forall\ 0 \le i \le p(n)$, $\forall\ -p(n) \le j \le p(n)+1$

- Let $G_i^{3,2} = \bigvee_{a=\{l,l'\}} \neg s[i,j,a]$, $\forall\ 0 \le i \le p(n)$, $\forall\ -p(n) \le j \le p(n)+1$, $\forall\ 0 \le l < l' \le v$

- Then $G_i^{3,1}$ consists of $2(p(n)+1)^2$ clauses and
- $G_i^{3,2}$ consists of $v(v+1)(p(n)+1)^2$ clauses. And we, at each time i, each tape square contains exactly one symbol from γ

iff

$G_i^{3,1}$ and $G_i^{3,2}$ are simultaneously satisfied. i.e.

$G_i^{3,1} \wedge G_i^{3,2}$ is a satisfiable clause.

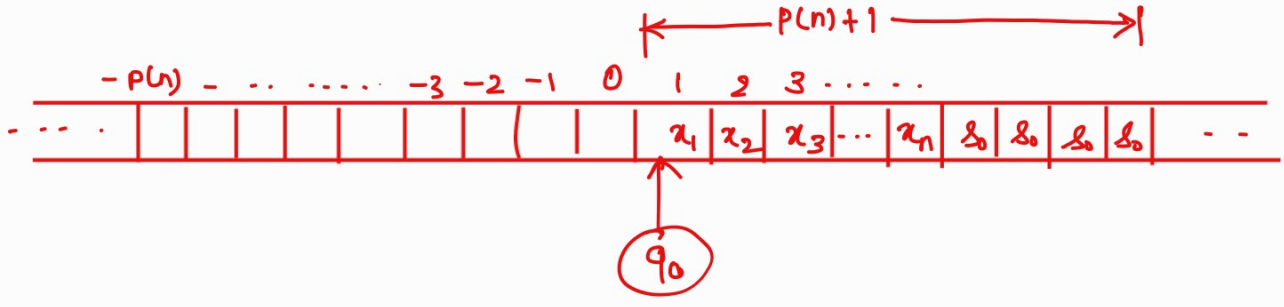
SW 2 : Represent the starting configuration of TM using boolean variables.

i.e. At time 0, the computation is in the initial configuration of its checking stage for input x.

How to represent SW_2 using boolean variables?

Recall : The initial configuration (ID) of any TM, on input string $x \in \Sigma^*$ is :

$$q_0 x = q_0 x_1 x_2 x_3 \ldots\ldots x_n$$



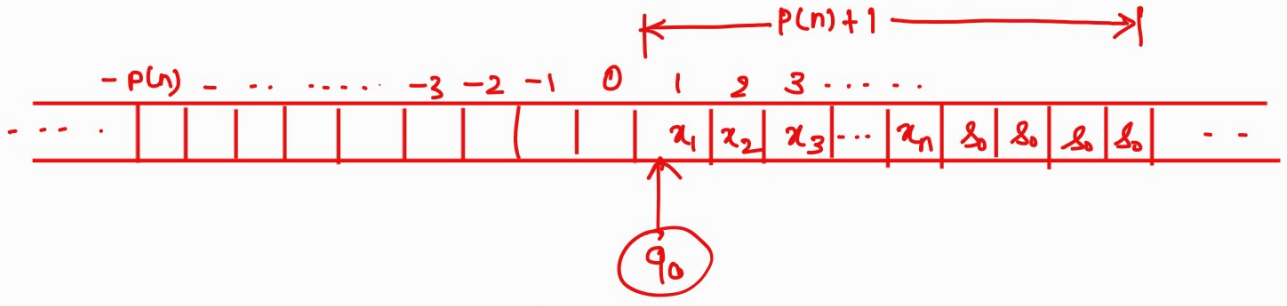
Observation : At time 0, the TM is in state q_0
iff
$Q[i=0, k=0]$ is a satisfiable clause
iff
$Q[0,0]$ is a satisfiable clause.

Observation : At time 0, the R/W head is scanning tape square 1 iff
$H[i=0, j=1]$ is a satisfiable clause
iff
$H[0,1]$ is a satisfiable clause

Observation: At time 0, the input tape contains the input string x starting at tape square 1

iff

at tape square 1, the symbol is s_1, and
at tape square 2, the symbol is s_2, and
$\vdots$
at tape square n, the symbol is s_n, and
after that, upto $p(n)+1$, the symbol is s_0.



iff

$S[i=0, j=1, l=1] \wedge S[i=0, j=2, l=2] \wedge \dots \wedge$
$S[i=0, j=n, l=n] \wedge S[i=0, j=n+1, l=0] \wedge$
$S[i=0, j=n+2, l=0] \wedge \dots \wedge S[i=0, j=p(n)+1, l=0]$
is a satisfiable clause.

iff

$S[0,1,s_1] \wedge S[0,2,s_2] \wedge \dots \wedge S[0,n,s_n] \wedge$
$S[0,n+1,s_0] \wedge S[0,n+2,s_0] \wedge \dots \wedge S[0,p(n)+1,s_0]$
is a satisfiable clause.

iff

$\bigwedge_{a=1}^{n} S[i,a,s_a] \wedge \bigwedge_{b=1}^{p(n)+1-n} S[0,n+b,0]$ is a satisfiable clause

SW2 done.

What we just proved?

At time 0, the computation is in the initial configuration of its checking stage for input x

iff

$$Q[0,0] \wedge H[0,1] \wedge \left(\bigwedge_{a=1}^{n} S[0,a,s_a] \wedge \bigwedge_{b=1}^{p(n)+1-n} S[0,n+b,0] \right)$$

is a satisfiable clause.

- Where are we standing now?

Systematic way	Our status
SW 1(a)(i)	Done
SW 1 (a) (ii)	
SW 1 (b)	
SW 1 (c)	
SW 2	
SW 3	Remaining
SW 4	

SW 3 : Represent the finishing configuration of TM using boolean variables

- By time $p(n)$, the TM has entered state q_f and hence accepted x.

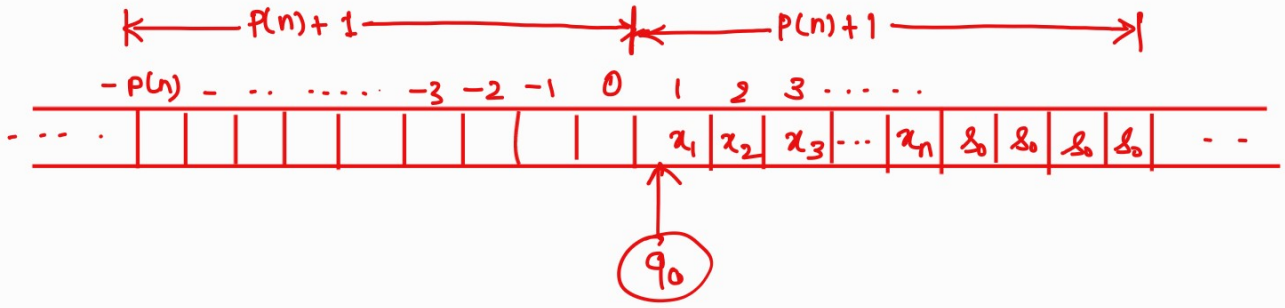
How to represent this using boolean variables?

- Recall that TM accepts string x

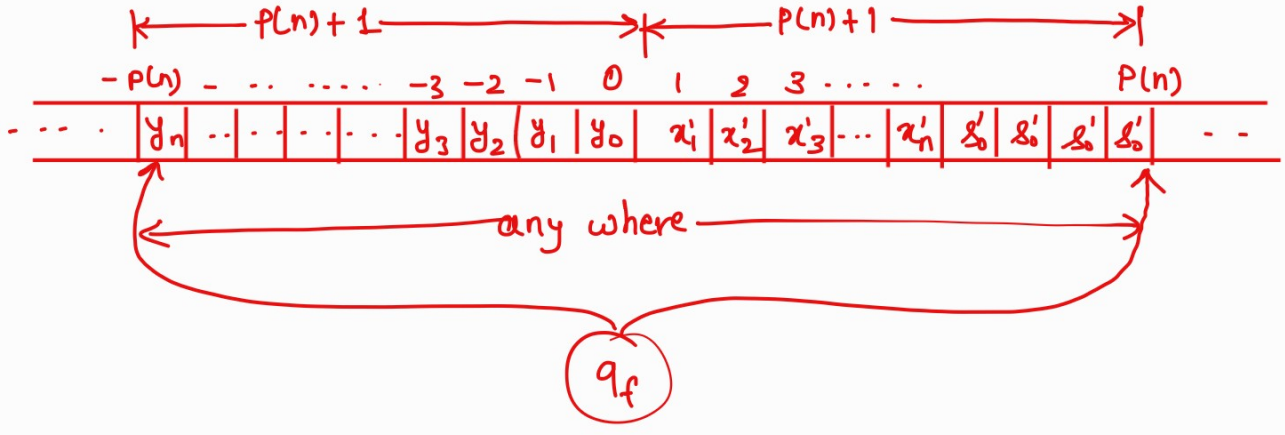
iff

$q_0 x \vdash^* \alpha q_f \beta$

iff



$\Downarrow$ * mutiple steps



iff

at time $p(n)+1$, the state is q_f (which is q_1).

iff

$Q[p(n)+1, 1]$ is a satisfiable clause. SW 3 done.

Where are we standing now?

Systematic way	Our status
SW 1(a)(i)	Done
SW 1(a)(ii)	
SW 1(b)	
SW 1(c)	
SW 2	
SW 3	
SW 4	Remaining

SW 4 : Represents the specific transition function using boolean variables.

- For each time i, where $0 \leq i \leq p(n)$, the configuration of the TM at time $(i+1)$, follows by a single application of the transition function δ, from the configuration at time i.

Recall the transition function

$$\delta(q_k, s_l) = (q_{k'}, s_{l'}, \Delta), \text{ if } q_k \in Q \setminus f$$
$$= (q_k, s_l, \text{halt}), \text{ otherwise.}$$

How to represent SW 4 ?

SW_4 is satisfied

iff

At time i, if (the state of turing machine is q_k, and the R/W head is scanning tape square j, and the content of square j is symbol s_l)

then

at time $(i+1)$, (the state of TM will be $q_{k'}$, according to δ, and the R/W head will be scanning tape square $j+\Delta$ according to δ, and the content of tape square j will be $s_{l'}$.

iff

$$\Big[(Q[i,k] \wedge H[i,j] \wedge s[i,j,l]) \longrightarrow (Q[i+1,k])\Big]$$

$$\wedge$$

$$\Big[(Q[i,k] \wedge H[i,j] \wedge s[i,j,l]) \longrightarrow (H[i+1,j+\Delta])\Big]$$

$$\wedge$$

$$\Big[(Q[i,k] \wedge H[i,j] \wedge s[i,j,l]) \longrightarrow (s[i+1,j,l'])\Big]$$

is a satisfiable clause

iff

$$\Big[\neg(Q[i,k] \wedge H[i,j] \wedge s[i,j,l]) \vee (Q[i+1,k])\Big]$$

$$\wedge$$

$$\Big[\neg(Q[i,k] \wedge H[i,j] \wedge s[i,j,l]) \vee (H[i+1,j+\Delta])\Big]$$

$$\wedge$$

$$\Big[\neg(Q[i,k] \wedge H[i,j] \wedge s[i,j,l]) \vee (s[i+1,j,l'])\Big]$$

iff

$$\Big(\neg Q[i,k] \vee \neg H[i,j] \vee \neg s[i,j,l] \vee Q[i+1,k]\Big)$$

$$\wedge$$

$$\Big(\neg Q[i,k] \vee \neg H[i,j] \vee \neg s[i,j,l] \vee H[i+1,j+\Delta]\Big)$$

$$\wedge$$

$$\Big(\neg Q[i,k] \vee \neg H[i,j] \vee \neg s[i,j,l] \vee s[i+1,j,l']\Big)$$

is a satisfiable clause

SW 4 Proved

Where are we standing now?

Systematic way	Our status
SW 1(a) (i)	Done
SW 1 (a) (ii)	
SW 1 (b)	
SW 1 (c)	
SW 2	
SW 3	
SW 4	

Using SW1, SW2, SW3 and SW 4, we can get a transformation or reduction.

The transformation is polynomial time.