Complexity Theory I Jan – May, 2023

Lecture 2: Cook-Levin Theorem

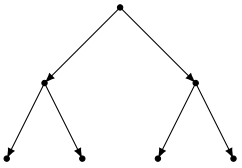
Lecturer: Partha Mukhopadhyay *Scribe: Soham Chatterjee*

A turing machine is a $5-$tuple $(Q, \Gamma, \delta, q_0, q_{accept})$ where $\Gamma = \Sigma \cup \{\#\}$ and $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
For a Nondeterministic Turing Machine (NTM) $\delta : Q \times \Gamma \to P(Q \times \Gamma \times \{L, R\})$

Note:-

For all these turing machines the machine halts on every path

.



The all possible paths of a Non-deterministic Turing Machineis showed like a directed tree where each edge is directed from parent node to its childs.

Let $f : \mathbb{N} \to \mathbb{N}$ is a function. A Deterministic Turing Machinewith runtime $f(n)$ means for input x, $|x| = n$, machine halts in time $O(f(n))$. A Non-deterministic Turing Machinewith runtime $f(n)$ means for every path the machine halts within $O(f(n))$ time
$\boldsymbol{DTIME(n^c)} \coloneqq \{L \mid$ Such L can be decided by a Deterministic Turing Machinewith runtime $O(n^c)\}$ $c =$constant (Sometimes also written only $TIME(n^c)$)
$\boldsymbol{P} \coloneqq DTIME(n^c)$ $\boldsymbol{NP} \coloneqq NTIME(n^c)$ $\boldsymbol{EXP} \coloneqq DTIME(2^{n^c})$

1 Logical Characterization of NP

NP : Class of problems that have a deterministic poly time verifier. $L \in NP$ then there is a polynomial p and polynomial time computable algorithm $v(\cdot, \cdot)$ such that $x \in L \iff \exists\, y$, $|y| \leq p(|x|)$ such that $v(x, y)$ accepts.

But not all problems have a easy way to get short verifier. There are problems for which it is hard to get a certificate. EG: Show that two finite groups are non-isomorphic, Show that a boolean formula is not satisfiable.

Claim 1

Efficient verifier $\iff$ $NP-$machine

Proof. $\implies$

On input x the $NP-$ machine gives y, $|y| \leq p(|x|)$ for some polynomial function and run the verifier $v(x, y)$

$\impliedby$

On input x if $\exists$ a correct path of the $NP-$machine then we can encode the path in a bit string and it will be a poly$(|x|)$ and give that as a verifier. □

2 Reduction

Definition 1: Polynomial Time Reduction

$f : \Sigma^* \to \Sigma^*$ is a polynomial time computable function

$$A \leq_P B$$

Polynomial Time Reducible

If there is a polynomial time computable $f : \Sigma^* \to \Sigma^*$ such that

$$x \in A \iff f(x) \in B$$

Definition 2: $NP - Hard$ and $NP - complete$

L is $NP - hard$ if $\forall\ A \in NP\ A \leq_P L$. If $L \in NP$ then L is $NP - complete$

If L_1 and L_2 both are $NP - complete$ then

$$\left.\begin{array}{l} L_1 \leq_P L_2 \\ L_2 \leq_P L_1 \end{array}\right\} \implies L_1 \equiv_P L_2$$

Question 1

Prove A is $NP - complete$

Solution: Let L be a known $NP - complete$ problem then show that $L \leq_P A$

□

3 Cook-Levin Theorem

Before we can even show that a problem is $NP - complete$ we have to first show existence of a $NP - complete$ problem. Cook and Levin they independently showed that SAT is an $NP - complete$ problem.

Theorem 1 Cook-Levin Theorem [Coo71] and [Lev73]
SAT is $NP - complete$

The nodes showed in the tree-diagram of a Non-deterministic Turing Machineare called configurations at different point of time of the Non-deterministic Turing Machine.

Definition 3: Configuration of a Turing Machine

A configuration of a Turing Machine at any point of time t is basically the snapshot of the turning machine at that time. It is an ordered pair of the head position, current state, and tape content. Any configuration is initialized and ended with a special symbol, let # mark its ending. Then the contents of the tape at the left of the head, then the current state then the content of the cell where the head is pointed and the content of the tape at the right of the head. So if any configuration is like $(\#, w_1, w_2, q, w_3, w_4, w_5, w_6, \#)$ then the head is pointed at 3rd letter in state q and the content of the tape is $w_1w_2w_3w_4w_5w_6$

In the proof of the theorem we use a concept called Computation Tableau

Definition 4: Computation Tableau of a Turing Machine

It is a table of configurations where form $i-$the row to $i+1-$the row it follows the turning machine's transition function at time i on any input x and the first row is the starting configuration. A tableau is accepting if any in the tableau is an accepting configuration.

Now comes to the proof of the theorem.

Proof. Let L be any language in NP. Then there exists a nondeterministic turning machine M with runtime n^k and $L = \mathcal{L}(M)$.

Idea: The idea is to encode the computation tableau of M on input x, $|x| = n$ to a boolean formula φ such that there is an accepting configuration iff $\varphi \in SAT$

Let $x = w_1 w_2 \dots w_n$. Let $C = Q \cup \Gamma \cup \{\#\}$ where Q is the set of states if M and Γ is set of alphabets of M. Since the turning machine runs for time n^k it can at most access n^k space, So the computation tableau will have n^k rows and n^k columns. Now the initial configuration is

$$(\#, q_0, w_1, w_2, \dots, w_n, \sqcup, \dots, \sqcup, \#)$$

Now any cell in the tableau contains a symbol from C. So for each $(i,j)-$the cell and $s \in C$ introduce a variable $x_{i,j,s}$. $x_{i,j,s} = 1$ iff the $(i,j)-$the cell in the tableau has the symbol s, otherwise $x_{i,j,s} = 0$.

Therefore we can encode the starting configuration as

$$\varphi_{start} = x_{1,1,\#} \wedge \left(\bigwedge_{j=2}^{n+1} x_{1,j,w_{j-1}} \right) \wedge \left(\bigwedge_{j=n+2}^{n^k-1} x_{1,j,\sqcup} \right) \wedge x_{1,n^k,\#}$$

Now setting $x_{i,j,s} = 1$ corresponds to placing s in $(i,j)-$the cell. To obtain a correspondence between an assignment and a tableau, we must ensure that the assignment sets to 1 exactly one variable for each cell. First, we need to ensure each cell has at least one symbol placed in it. We do it by $\bigvee_{s \in C} x_{i,j,s}$. Now we have to ensure that the cell does not have more than two symbols placed in it. We do it by $\bigvee_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \wedge \overline{x_{i,j,t}})$. Hence for each cell, these two conditions both should follow, therefore

$$\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigvee_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \wedge \overline{x_{i,j,t}}) \right)$$

There condition should follow for every cell, therefore, we get the second part for our desired boolean function

$$\varphi_{cell} = \bigwedge_{1 \leq i,j \leq n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigvee_{s,t \in C, s \neq t} (\overline{x_{i,j,s}} \wedge \overline{x_{i,j,t}}) \right) \right]$$

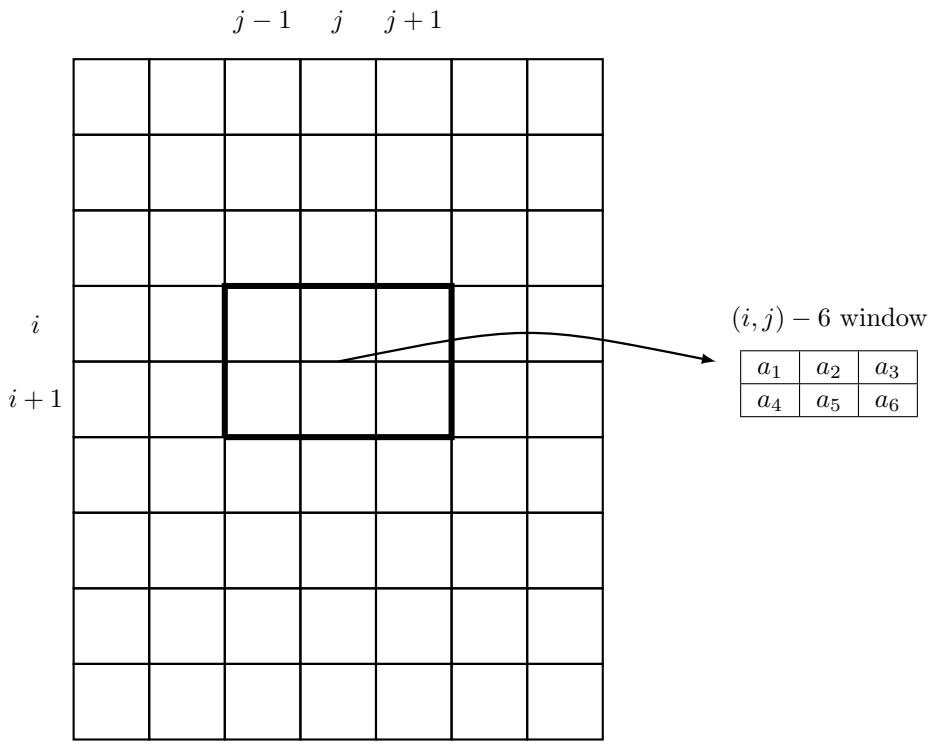
Formula φ_{accept} guarantees that an accepting configuration occurs in the tableau. The formula ensures that φ_{accept}, the symbol for the accept state, occurs in one of the cells of the tableau by stipulating that one of the corresponding variables is set to 1. Hence the formula for it is

$$\varphi_{accept} = \bigwedge_{1 \leq i,j \leq n^k} x_{i,j,q_{accept}}$$

The last part of the formula has to do with the transitions of the turing machine. Here we use the concept of window in the computation tableau

Definition 5: $(i,j) - 6$ window in a Computation Tableau

A window in the tableau is a 2×3 piece with adjacent rows and and columns.



A window is legal if it does not violate transition function of the turing machine. Determining which windows are legal can be done by case analysis.

Example 1 (Legal and Illegal 6-window)

a	q_1	b
q_2	a	c

$\longrightarrow$ Legal

a	q_1	b
a	a	q_2

$\longrightarrow$ Legal

a	b	a
a	b	q_2

$\longrightarrow$ Legal

a	q_1	b
q	a	a

$\longrightarrow$ Illegal

$\#$	b	a
$\#$	b	a

$\longrightarrow$ Illegal

Hence

$$\varphi_{move} = \bigwedge_{1 \le i,j \le n^k} ((i,j)-6 \text{ window is legal})$$

Therefore the final boolean circuit is

$$\varphi = \varphi_{start} \wedge \varphi_{cell} \wedge \varphi_{move} \wedge \varphi_{accept}$$

□

References

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. page 151–158, 1971.

[Lev73] L. A. Levin. Universal sequential search problems. *Probl. Peredachi Inf.*, 9:115–116, 1973.