CS 4510: Automata and Complexity **Fall 2019**

Lecture 19: SAT is NP-complete

November 11, 2019

Lecturer: Santosh Vempala *Scribe: Aditi Laddha*

19.1 Introduction

Definition 19.1 (NP) *There are 2 equivalent definitions of the class* NP*:*

- *A language $L \in \mathrm{NP}$ if there exists a nondeterministic Turing machine, M that decides L in n^k time, i.e., for every string x with $|x| = n$, if $x \in L$, then on input x, M has at least one computation path that accepts x in at most n^k steps and if $x \notin L$, then M rejects x on all compuation paths in n^k steps.*
- *A language $L \in \mathrm{NP}$ if there exists a polynomial time verifier for L. A verifier for L is a deterministic Turing machine M such that $L = \{x \mid M \text{ accepts } \langle x, c\rangle \text{ for a some string } c\}$. A polynomial time verifier runs in time polynomial in $|x|$.*

Definition 19.2 (Polynomial Time reductions) *A language A is said to polynomial time reducible to a language B or $A \leq_P B$ if there exists a polynomial time computable function $f : \Sigma^* \to \Sigma^*$ such that $\forall x \in \Sigma^*$,*

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

Definition 19.3 (NP-hard) *A language L is called NP-hard if $\forall L' \in \mathrm{NP}$, $L' \leq_P L$.*

Definition 19.4 (NP-complete) *A language L is* NP*-complete if*

- *L is in* NP*, and*
- *L is* NP*-hard.*

19.2 Cook-Levin Theorem

Theorem 19.5 (Cook-Levin Theorem) *SAT is* NP*-complete.*

Proof: *SAT* is in NP. Given a formula ϕ, a NTM can nondeterministically guess an assignment for ϕ and accept if the assignment satisfies ϕ in time polynomial in size of ϕ.
Consider $L \in \mathrm{NP}$. Let $M = (Q, q_0, \Sigma, \Gamma, q_A, \delta)$ be the nondeterministic Turing Machine that accepts L in n^k time. Let $C = \Sigma \cup \Gamma \cup \#$ where $\{\#\}$ is the blank symbol. We can encode the configurations of a TM on the tape by wherever the head is on the tape, write the current state to the left.

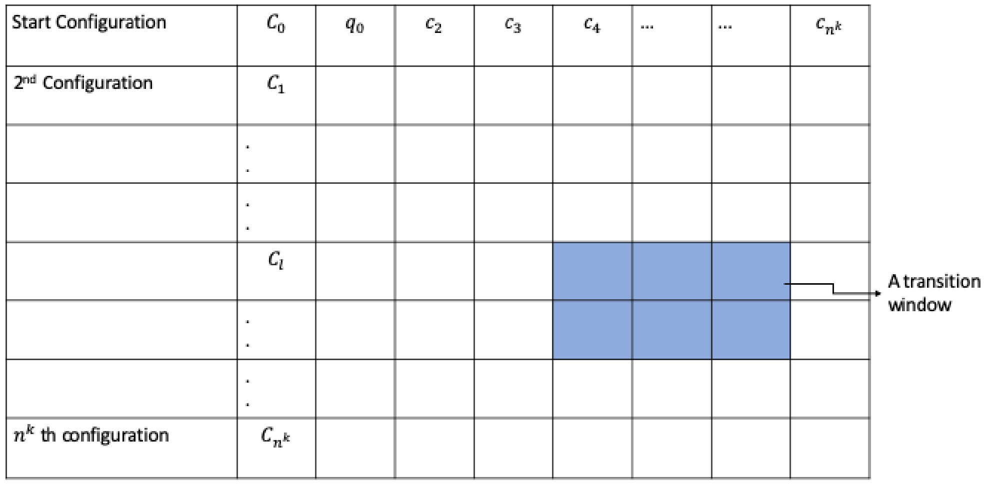

Figure 19.1: An $n^k \times n^k$ table of configurations.

This gives us a table, T of the machine execution on x. Let $T(i, j)$ be the symbol in the ith row and jth column of T. The variables of the formula are

$$X_{i,j,c} = \begin{cases} 1 & \text{if } T(i,j) = s \\ 0 & \text{otherwise} \end{cases}$$

where $i, j \in \{1, \ldots n^k\}$ and $c \in C$. We want a formula ϕ such that a satisfying assignment of ϕ corresponds to an accepting table of M. So, the formula needs to check four things:

1. Every cell is occupied by exactly one symbol. For cell i, j we need

$$\phi_{i,j} = \left(\bigvee_{s \in C} X_{i,j,c} \right) \wedge \left(\bigwedge_{s,s' \in C, s \neq s'} (\overline{X_{i,j,s}} \vee \overline{X_{i,j,s'}}) \right)$$

   We want this to be true for every cell of the table which gives the formula:

$$\phi_{cell} = \bigwedge_{1 \leq i,j \leq n^k} \phi_{i,j}$$

2. Starting configuration: The first row of the table must correspond to the start configuration of M on x, i.e., $C_0 = q_0 x_1 x_2 \ldots x_n \#\# \ldots \#$. Let c_i denote the ith symbol of C_0, then

$$\phi_{start} = X_{1,1,q_0} \wedge X_{1,2,c_2} \ldots \wedge X_{1,n^k,c_{n^k}}$$

3. M reaches q_A on x, so at least one of the cell must contain q_A:

$$\phi_{accept} = \bigvee_{1 \leq i,j \leq n^k} X_{i,j,q_A}$$

4. The transitions are valid so that each row of the table corresponds to a configuration that follows from the previous row's configuration following the transition function δ. We can check this by checking the symbols in a 6×6 window containing the state as follows:
   Right move: Consider a transtion $\delta(q, a) = (q', a', R)$. We can encode it as:

b	q	a
b	a'	q'

$$\begin{aligned}\phi_{q,a} &= X_{i,j,b} \wedge X_{i,j+1,q} \wedge X_{i,j+2,a} \Rightarrow X_{i+1,j,b} \wedge X_{i+1,j+1,a'} \wedge X_{i+1,j+2,q'} \\ &\overline{X_{i,j,b} \wedge X_{i,j+1,q} \wedge X_{i,j+2,a}} \vee (X_{i+1,j,b} \wedge X_{i+1,j+1,a'} \wedge X_{i+1,j+2,q'}) \\ &(\overline{X_{i,j,b}} \wedge \overline{X_{i,j+1,q}} \wedge \overline{X_{i,j+2,a}}) \vee (X_{i+1,j,b} \wedge X_{i+1,j+1,a'} \wedge X_{i+1,j+2,q'})\end{aligned}$$

Left move: Consider a transtion $\delta(q, a) = (q', a', L)$. We can encode it as:

b	q	a
q'	b	a'

$$\begin{aligned}\phi_{q,a} &= X_{i,j,q} \wedge X_{i,j+1,a} \wedge X_{i,j+2,b} \Rightarrow X_{i+1,j,q'} \wedge X_{i+1,j+1,b} \wedge X_{i+1,j+2,a'} \\ &\overline{X_{i,j,q} \wedge X_{i,j+1,a} \wedge X_{i,j+2,b}} \vee (X_{i+1,j,q'} \wedge X_{i+1,j+1,b} \wedge X_{i+1,j+2,a'}) \\ &(\overline{X_{i,j,q}} \wedge \overline{X_{i,j+1,a}} \wedge \overline{X_{i,j+2,b}}) \vee (X_{i+1,j,q'} \wedge X_{i+1,j+1,b} \wedge X_{i+1,j+2,a'})\end{aligned}$$

Since M is nondeterministic, $\delta(q, a)$ might be a set. Consider

$$\delta(q, a) = \{(q_i, a_i, L) : i \in L\} \cup \{(q_i, a_i, R) : i \in R\},$$

then $\phi_{(q,a)} =$

$$X_{i,j,b} \wedge X_{i,j+1,q} \wedge X_{i,j+2,a} \Rightarrow \bigvee_{l \in L} (X_{i+1,j,q_l} \wedge X_{i+1,j+1,b} \wedge X_{i+1,j+2,a_l}) \bigvee_{r \in R} (X_{i+1,j,b} \wedge X_{i+1,j+1,a_r} \wedge X_{i+1,j+2,q_r})$$

Every 6×6 window in the table must be legal. This gives the formula:

$$\phi_{transition} = \bigwedge_{1 \le i,j \le n^k} \bigwedge_{\gamma \in \delta} \phi_\gamma$$

Since we want all four conditions to be true, the overall formula is

$$\phi(x) = \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{transition}$$

and

$$x \in L \Longleftrightarrow \phi(x) \in SAT$$

If $x \in L$ with $|x| = n$, then on input x, M has at least one computation path that accepts x in at most n^k steps(and hence the machine uses at most n^k tape cells) and we can fill the table with the configurations of M from this accepting path and set the corresponding variables to *true*. If there exists a satisfying assignment for $\phi(x)$ then it must correspond to a valid sequence of configurations of M with input x. Also, these configurations must lead to the accepting state q_A and hence $x \in L$.

19.2.1 Complexity

The table has $n^k \times n^k$ cells and each cell has $|C|$ variables associated with it, where C is a constant that depends on the machine M and not on the length of the input n. So, the number of variables in ϕ is $O(n^{2k})$. We analyze the size of each formula:

- ϕ_{cell} contains a fixed formula for each cell, and the size of this formula depends on $|C|$. So, size of ϕ_{cell} is $O(n^{2k})$.

- ϕ_{start} contains a single clause with n^k variables.
- ϕ_{accept} contains one variable for each cell of the table, so its size is $O(n^{2k})$
- $\phi_{transition}$ contains a formula whose size is fixed and depends only on δ(the transition function of M) for each cell of the table. So, size of $\phi_{transition}$ is $O(n^{2k})$.

So, the total size of the formula is $O(n^{2k})$. Hence, this is a polynomial time reduction.

■

19.3 References

- Ch 7.4 NP-Completeness, "Introduction to the Theory of Computation"