Complexity Theory

(class 9 of 15)

Álvaro Moreira
alvaro.moreira@inf.ufrgs.br

Instituto de Informática
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brasil
http://www.inf.ufrgs.br

Cook-Levin Theorem - SAT is NP-Hard (I)

To prove the Cook-Levin Theorem we have to show that **every** NP language L can be reduced in polynomial time to L_{SAT}:

- we need a polynomial time transformation of any string $w \in \{0, 1\}^*$ of any NP language L to a string φ_w which represent a CNF formula
- this polynomial transformation cannot be arbitrary. It should be such that $w \in L$ iff φ_w is satisfiable, i.e the following should hold:

$$w \in L \quad \mathit{iff} \quad \varphi_w \in L_{\mathsf{SAT}}$$

Obs: Observe that the only thing known about L is that it is in NP

SAT is NP-Hard (II)

- The proof given is taken from the book *Models of Computation and Formal Languages* by R. Gregory Taylor (p. 416-424)
- A language L is in NP iff it is decided by some non-deterministic polynomial time TM (we consider a single-tape TM with infinite squares in both directions)
- The reduction takes a string w from an NP language L and produces, in polynomial time, a Boolean formula φ_w such that:
- A non-deterministic polynomial time TM accepts w iff the formula φ_w has a satisfying assignment.

SAT is NP-Hard (II)

- Therefore w is in L if and only if φ_w is satisfiable.
- Actually constructing the reduction to work in this way is a conceptually simple task. The construction however has a lot of detail
- A Boolean formula may contain the Boolean operations AND, OR, and NOT, and these operations form the basis for the circuitry used in electronic computers
- Hence the fact that we can design a Boolean formula to simulate a Turing machine isn't surprising

Preliminaries (I)

- The NDTM M decides, in polynomial time, if $w \in L$
- Let $t = p(|w|)$ (polynomial in the size of the input word w) thus, the decision about w occurs in t or fewer steps
- Hence, to accept w, M can visit at most t tape squares to the right and at most t tape squares to the left of where the head is initially positioned, i. e. at most $2t + 1$ squares
- Assume M have $m + 1$ states $q_0, \ldots q_m$ and the alphabet has $r + 1$ symbols $\alpha_0, \ldots \alpha_r$ ($\alpha_0 = B$, the blank symbol)

Preliminaries (II)

- A grid of size $t \times (2t+1)$ containing one (tape) symbol at each position represents a computation of M
- Each of the t lines of the grid represents the contents of tape at time/step t
- Line 0 of the table represents the tape's contents at time 0, line 1 represents the tape's content at time 1, and so on

Preliminaries (III)

- The formula φ_w should describe the computation of M
- It should state that:
  - M starts scanning the leftmost symbol of w on a tape that contains w and is otherwise blank;
  - all executed transitions are in accordance to M;
  - M halts after no more that t steps, scanning a 1 on an otherwise blank tape.

Sentence Letters

- τ_{ijk} for $0 \leqslant i \leqslant r$, $1 \leqslant j \leqslant 2t+1$, $0 \leqslant k \leqslant t$ - expresses that symbol α_i of the TM alphabet is at position j, k of the table (column j, line k) - i.e at time k, the symbol α_i is at the square at position j

- σ_{hjk} for $0 \leqslant h \leqslant m$, $1 \leqslant j \leqslant 2t+1$, $0 \leqslant k \leqslant t$ - expresses that the TM is at state q_h when at time k it is scanning the square at position j

The formula φ_w

The formula φ_w is a conjunction of 6 sentences, each of which in CNF:

- Sentence 1: Describes M's initial configuration
- Sentence 2: Each computation step has one state and one scanned square
- Sentence 3: At each computation step, each tape square is either blank or contains exactly one symbol
- Sentence 4: Each computation step results in a machine configuration that is either identical to its precedent or implements one valid transition of M
- Sentence 5: M halts at (or before) time t
- Sentence 6: At time t, M exhibits a valid final configuration

Sentence 1 - Initial Configuration

$$
\begin{array}{llllllll}
 & \tau_{0,1,0} & \wedge & \tau_{0,2,0} & \ldots & \wedge & \tau_{0,t,0} \\
\wedge & \tau_{i_1,t+1,0} & \wedge & \tau_{i_2,t+2,0} & \ldots & \wedge & \tau_{i_n,t+n,0} \\
\wedge & \tau_{0,t+n+1,0} & \wedge & \tau_{0,t+n+2,0} & \ldots & \wedge & \tau_{0,2t+1,0} \\
\wedge & \sigma_{0,t+1,0} & & & & &
\end{array}
$$

- 1st line express that the 1st t squares have the blank symbol
- 2nd line expresses that the following n squares have the input $w = i_1 i_2 \ldots i_n$
- 3rd line expresses that remaining squares after the input w are all blank
- 4th line expresses that at the time 0 the machine is at state q_0 and scanning symbol at square $t + 1$ (1st symbol of the input)

This sentence has $2t + 2$ conjuncts, so $O(t)$

Sentence 2 - at each step, one state, one square

- Formula $!\{P_1, \ldots P_k\}$ is the exclusive OR of sentence letters P_1, ..., P_k

- At each time/step the machine is at exactly one and is scanning exactly one square:

$$
\begin{array}{llllll}
\text{(at time } 0) & & !\{\ \sigma_{h,j,0} & | & 0 \leqslant h \leqslant m, & 0 \leqslant j \leqslant 2t+1\} \\
\text{(at time } 1) & \wedge & !\{\ \sigma_{h,j,1} & | & 0 \leqslant h \leqslant m, & 0 \leqslant j \leqslant 2t+1\} \\
\text{(at time } 2) & \wedge & !\{\ \sigma_{h,j,2} & | & 0 \leqslant h \leqslant m, & 0 \leqslant j \leqslant 2t+1\} \\
\ldots & & \ldots & & & \\
\text{(at time } t) & \wedge & !\{\ \sigma_{h,j,t} & | & 0 \leqslant h \leqslant m, & 0 \leqslant j \leqslant 2t+1\}
\end{array}
$$

Each conjunct is an exclusive OR and has length $O(t^2)$ Since there are $t+1$ conjuncts, sentence 2 has length $O(t^3)$

Sentence 3

- Expresses that each square of the grid is blank or has exactly one symbol

- Each conjunct below describes one square of the grid

$$
\begin{array}{ll}
 & !\{\tau_{i,1,0} \mid 0 \leqslant i \leqslant r\} \wedge !\{\tau_{i,2,0} \mid 0 \leqslant i \leqslant r\} \ldots \wedge !\{\tau_{i,2t+1,0} \mid 0 \leqslant i \leqslant r\} \\
\wedge & !\{\tau_{i,1,1} \mid 0 \leqslant i \leqslant r\} \wedge !\{\tau_{i,2,1} \mid 0 \leqslant i \leqslant r\} \ldots \wedge !\{\tau_{i,2t+1,1} \mid 0 \leqslant i \leqslant r\} \\
\wedge & !\{\tau_{i,1,2} \mid 0 \leqslant i \leqslant r\} \wedge !\{\tau_{i,2,2} \mid 0 \leqslant i \leqslant r\} \ldots \wedge !\{\tau_{i,2t+1,2} \mid 0 \leqslant i \leqslant r\} \\
\ldots & \ldots \\
\wedge & !\{\tau_{i,1,t} \mid 0 \leqslant i \leqslant r\} \wedge !\{\tau_{i,2,t} \mid 0 \leqslant i \leqslant r\} \ldots \wedge !\{\tau_{i,2t+1,t} \mid 0 \leqslant i \leqslant r\}
\end{array}
$$

The total length of sentence 3 is $O(t^2)$

Sentence 4

- Each computation step results in a machine configuration (row) that is either identical to its precedent or implements one valid transition of M
- This is a complex sentence and will be constructed in parts...

Sentence 4 - Preliminaries

Consider that the instructions of the TM M are the following:

- $\mathscr{W} = \{(q_k, \alpha_k;\ \alpha'_k, q'_k,) \ \mid \ k = 1 \ldots N_{\mathscr{W}}\}$
- $\mathscr{MR} = \{(q_k, \alpha_k; R, q'_k) \ \mid \ k = 1 \ldots N_{\mathscr{MR}}\}$
- $\mathscr{ML} = \{(q_k, \alpha_k; L, q'_k) \ \mid \ k = 1 \ldots N_{\mathscr{ML}}\}$

where $N_{\mathscr{W}}$, $N_{\mathscr{MR}}$, $N_{\mathscr{ML}}$ are the number of write, move right and move left instructions of the TM M, respectively

Sentence 4 - Notscan(n,p)

- 1st line of the formula below says that at step n TM M is not scanning square at position p
- 2nd line of the formula says that the symbol at position p is the same at step n and $n+1$

$$\texttt{Notscan}(n, p) \equiv$$

$$(\neg\sigma_{0,p,n} \ \wedge \ \neg\sigma_{1,p,n} \ldots \wedge \ \neg\sigma_{m,p,n})$$

$$\wedge \quad ((\tau_{0,p,n} \wedge \tau_{0,p,n+1}) \ \vee \ (\tau_{1,p,n} \wedge \tau_{1,p,n+1}) \ \vee \ \ldots \vee \ (\tau_{r,p,n} \wedge \tau_{r,p,n+1}))$$

This formula is not CNF. Its total length his constant - $O(1)$

Sentence 4 - $\mathtt{Halt}(n,p)$

- 1st line says that the head does not move its position p at the time $n+1$
- 2nd line says that the symbol in the square at position p in the time after $n+1$ does not change

$$\mathtt{Halt}(n,p) \equiv$$

$$((\sigma_{0,p,n} \wedge \sigma_{0,p,n+1}) \ \vee \ (\sigma_{1,p,n} \wedge \sigma_{1,p,n+1}) \ \vee \ \ldots \vee \ (\sigma_{m,p,n} \wedge \sigma_{m,p,n+1}))$$

$$\wedge \quad ((\tau_{0,p,n} \wedge \tau_{0,p,n+1}) \ \vee \ (\tau_{1,p,n} \wedge \tau_{1,p,n+1}) \ \vee \ \ldots \vee \ (\tau_{r,p,n} \wedge \tau_{r,p,n+1}))$$

This formula is not CNF. Its total length his constant - $O(1)$

Sentence 4 - MoveRigth(n,p)

- 1st line says that p is not the last square
- 2nd line says that after a move right instruction the head moves to the right one position
- 3rd line says the symbol at position p does not change at time $n+1$ in the case of a move right

$$
\begin{aligned}
&\texttt{MoveRight}(n,p) \equiv \\
&p \neq 2t+1 \\
\wedge \quad &\bigvee_{(q,\alpha;R;q')\in\mathscr{MR}} (\sigma_{q,p,n} \ \wedge \ \sigma_{q',p+1,n+1}) \\
\wedge \quad &(\tau_{i,p,n} \ \wedge \ \tau_{i,p,n+1})
\end{aligned}
$$

This formula is not CNF. Its total length his constant - $O(1)$

Sentence 4 - MoveLeft(n,p)

- 1st line says that p is not the 1st square
- 2nd line says that after a move left instruction the head moves to the left one position
- 3rd line says the symbol at position p does not change at time $n+1$ in the case of a move left

$$
\begin{aligned}
&\mathtt{MoveLeft}(n, p) \equiv \\
&p \neq 1 \\
\wedge\ &\bigvee_{(q,\alpha;L;q') \in \mathscr{ML}} (\sigma_{q,p,n} \wedge \sigma_{q',p-1,n+1}) \\
\wedge\ &(\tau_{i,p,n} \wedge \tau_{i,p,n+1})
\end{aligned}
$$

This formula is not CNF. Its total length his constant - $O(1)$

Sentence 4 - $\mathtt{Write}(n,p)$

- 1st line says the old symbol at position p does change to a new symbol at time $n+1$
- 2nd line says that after a write instruction the head does not move m

$$\mathtt{Write}(n,p) \equiv$$

$$\bigvee_{(q,\alpha_i;\alpha_i',q')\in\mathscr{W}} (\tau_{i,p,n} \wedge \tau_{i',p,n+1})$$

$$\wedge \quad (\sigma_{q,p,n} \wedge \sigma_{q',p,n+1})$$

This formula is not CNF. Its total length his constant - $O(1)$

Sentence 4

Each computation step results in a machine configuration that is either identical to its precedent or implements one valid transition of M

$$\bigwedge_{n=0\ldots t} \Big(\bigwedge_{p=0\ldots 2t+1} (\texttt{NotScan}(n,p) \vee \texttt{Halt}(n,p) \vee \texttt{MoveRight}(n,p)$$

$$\vee\ \texttt{MoveLeft}(n,p) \vee \texttt{Write}(n,p))\Big)$$

This is not a CNF formula, but there is an equivalent CNF.

Its total length is $O(t^2)$

Sentence 5

The TM M halts at (or before) time t

$$\bigwedge_{1 \leqslant p \leqslant 2t+2} (\mathtt{NotScan}(t, p) \ \vee \ \mathtt{Halt}(t, p))$$

The length of sentence 5 is $O(t)$

Sentence 6

At time t, M exhibits a valid final configuration, i.e:

- (line 1 of formula below) there is exactly a single 1 at M' tape
- (line 2) all the other squares are blank
- (line 3) that single 1 is being scanned

$$
\begin{aligned}
& !\{\tau_{1,1,t}, \tau_{1,2,t} \ldots \tau_{1,2t+1,t}\} \\
\wedge \quad & (\tau_{1,1,t} \vee \tau_{0,1,t}) \wedge (\tau_{1,2,t} \vee \tau_{0,2,t}) \wedge \ldots \wedge (\tau_{1,2t+1,t} \vee \tau_{0,2t+1,t}) \\
\wedge \quad & !\{\tau_{0,1,t}, \sigma_{0,1,t} \ldots \sigma_{m,1,t}\} \ \wedge \ !\{\tau_{0,2,t}, \sigma_{0,2,t} \ldots \sigma_{m,2,t}\} \ \wedge \ldots \\
& \quad \ldots \wedge \ !\{\tau_{0,2t+1,t}, \sigma_{0,2t+1,t} \ldots \sigma_{m,2t+1,t}\}
\end{aligned}
$$

The length of sentence 6 is $O(t^2)$

Proof of Cook-Levin Theorem (I)

- The construction of φ_w can be done by a TM in polynomial time : $O(t^3)$
- Now it remains to show that

$$w \in L \Leftrightarrow \varphi_w \in L_{SAT}$$

Proof of Cook-Levin theorem (II)

- Suppose M accepts w, i.e. $w \in L$
- Then, by the way that φ_w is constructed there is an assignment that makes φ_w true
- Hence $\varphi_w \in L_{SAT}$

Proof of Cook-Levin theorem (III)

- Suppose φ_w is satisfiable, there must an assignment of values to the variables σ and τ that makes it true
- This assignment corresponds to a computation of M
- To build this computation we may start from the situation described in sentence 1 (initial configuration, that corresponds to a blank tape with only w) and build each row of the table corresponding to φ_w (notice that at each step/row only one valid action of M is executed)
- Since there is a computation of M leading to a final configuration with 1 on the tape, M accepts w