Computability and Complexity

Lecture 11

NP-completeness
Cook-Levin Theorem (SAT is NP-complete)

given by Jiri Srba

What Is the Hardest Problem in NP?

Question:

We do not know if problems in NP have polynomial time deterministic algorithms. Can we at least point out to some problem(s) in NP that are the most difficult ones to solve?

What Is the Hardest Problem in NP?

Question:

We do not know if problems in NP have polynomial time deterministic algorithms. Can we at least point out to some problem(s) in NP that are the most difficult ones to solve?

Answer:

This is indeed possible and *SAT* is one example of such a problem. In fact there are many other problems in NP that are difficult for the class NP (*CLIQUE*, *HAMPATH*, *SUBSET-SUM*, *3SAT*, ...). We call them NP-complete problems.

What Is the Hardest Problem in NP?

Question:

We do not know if problems in NP have polynomial time deterministic algorithms. Can we at least point out to some problem(s) in NP that are the most difficult ones to solve?

Answer:

This is indeed possible and *SAT* is one example of such a problem. In fact there are many other problems in NP that are difficult for the class NP (*CLIQUE*, *HAMPATH*, *SUBSET-SUM*, *3SAT*, ...). We call them NP-complete problems.

Consequence:

- If just one NP-complete problem can be solved in P then all other problems in NP will be solvable in P (hence P=NP).
- If P $\neq$ NP then none of the NP-complete problems is solvable in deterministic polynomial time.

NP-Completeness

Definition (NP-Completeness)

A language B is NP-complete iff

1. $B \in \mathrm{NP}$ (containment in NP), and
2. for every $A \in \mathrm{NP}$ we have $A \leq_P B$ (NP-hardness).

NP-Completeness

Definition (NP-Completeness)

A language B is NP-complete iff

1. $B \in \mathrm{NP}$ (containment in NP), and
2. for every $A \in \mathrm{NP}$ we have $A \leq_P B$ (NP-hardness).

Theorem

If B is NP-complete and $B \in \mathrm{P}$,
then $\mathrm{P} = \mathrm{NP}$.

Proof: We know that if $A \leq_P B$ and $B \in \mathrm{P}$ then $A \in \mathrm{P}$.

NP-Completeness

Definition (NP-Completeness)

A language B is NP-complete iff

1. $B \in \mathrm{NP}$ (containment in NP), and
2. for every $A \in \mathrm{NP}$ we have $A \leq_P B$ (NP-hardness).

Theorem

If B is NP-complete and $B \in \mathrm{P}$,
then $\mathrm{P} = \mathrm{NP}$.

Proof: We know that if $A \leq_P B$ and $B \in \mathrm{P}$ then $A \in \mathrm{P}$.

Theorem

If B is NP-complete, $B \leq_P C$, and $C \in \mathrm{NP}$,
then C is NP-complete.

Proof: Because $\leq_P$ is transitive (see the tutorial).

Cook-Levin Theorem

Cook-Levin Theorem

The language SAT is NP-complete.

Cook-Levin Theorem

The language SAT is NP-complete.

Proof: SAT is clearly in NP. We have to show that SAT is NP-hard:

Every language $A \in \mathrm{NP}$ is poly-time reducible to SAT.

Cook-Levin Theorem

The language SAT is NP-complete.

Proof: SAT is clearly in NP. We have to show that SAT is NP-hard:

Every language $A \in \mathrm{NP}$ is poly-time reducible to SAT.

- Let us assume any given $A \in \mathrm{NP}$, so
- there is a nondeterm. decider M for A running in time $O(n^k)$.

Cook-Levin Theorem

Cook-Levin Theorem

The language SAT is NP-complete.

Proof: SAT is clearly in NP. We have to show that SAT is NP-hard:

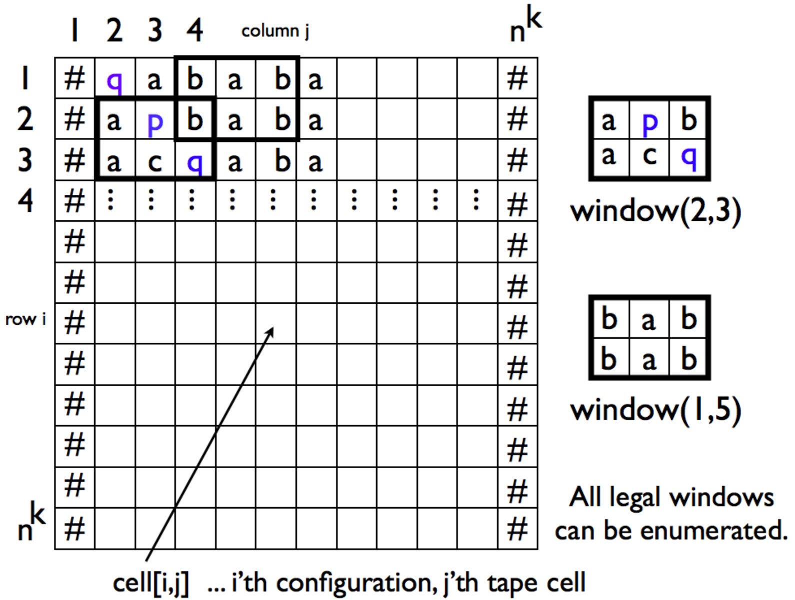
Every language $A \in \mathrm{NP}$ is poly-time reducible to SAT.

- Let us assume any given $A \in \mathrm{NP}$, so
- there is a nondeterm. decider M for A running in time $O(n^k)$.

Our aim: for any input string w construct in polynomial time a Boolean formula ϕ such that

M accepts w if and only if ϕ is satisfiable.

Table of Configurations of M Run on *ababa*



Boolean Formula Describing Table of Configurations

Assume a table of configurations when M is run on $w = w_1 \dots w_n$. We will construct a formula ϕ such that M accepts w iff $\phi \in SAT$.

$$\phi \stackrel{\text{def}}{=} \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$$

Boolean Formula Describing Table of Configurations

Assume a table of configurations when M is run on $w = w_1 \dots w_n$.
We will construct a formula ϕ such that M accepts w iff $\phi \in SAT$.

$$\phi \stackrel{\text{def}}{=} \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$$

The set of variables contains:

$$x_{i,j,s}$$

where $1 \leq i, j \leq n^k$ and $s \in C$ is a tape symbol or a control state.
Note: There are only polynomially many variables w.r.t. to n.

Boolean Formula Describing Table of Configurations

Assume a table of configurations when M is run on $w = w_1 \ldots w_n$.
We will construct a formula ϕ such that M accepts w iff $\phi \in SAT$.

$$\phi \stackrel{\text{def}}{=} \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$$

The set of variables contains:

$$x_{i,j,s}$$

where $1 \leq i, j \leq n^k$ and $s \in C$ is a tape symbol or a control state.
Note: There are only polynomially many variables w.r.t. to n.

Intuition:

Variable $x_{i,j,s}$ is true if and only if cell$[i,j]$ contains the symbol s.

Definition of ϕ_{cell}

Every cell$[i, j]$ contains exactly one symbol s.

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

Definition of ϕ_{cell}

Every cell$[i, j]$ contains exactly one symbol s.

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

Note: ϕ_{cell} is of polynomial size w.r.t. to n.

Definition of ϕ_{start}

The first row contains the initial configuration $q_0 w_1 \ldots w_n$.

$$\phi_{start} \stackrel{\text{def}}{=} x_{1,1,\#} \;\wedge\; x_{1,2,q_0} \;\wedge\; x_{1,3,w_1} \;\wedge\; x_{1,4,w_2} \;\wedge\; \ldots\; x_{1,n+2,w_n} \;\wedge$$
$$x_{1,n+3,\sqcup} \wedge \ldots x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}$$

Definition of ϕ_{start}

The first row contains the initial configuration $q_0 w_1 \dots w_n$.

$$\phi_{start} \stackrel{\text{def}}{=} x_{1,1,\#} \;\wedge\; x_{1,2,q_0} \;\wedge\; x_{1,3,w_1} \;\wedge\; x_{1,4,w_2} \;\wedge\; \dots \; x_{1,n+2,w_n} \;\wedge$$
$$x_{1,n+3,\sqcup} \wedge \dots x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}$$

Note: ϕ_{start} is of polynomial size w.r.t. to n.

Definition of ϕ_{accept}

There is an accepting configuration in the table.

$$\phi_{accept} \stackrel{\text{def}}{=} \bigvee_{1 \leq i,j \leq n^k} x_{i,j,q_{accept}}$$

Definition of ϕ_{accept}

There is an accepting configuration in the table.

$$\phi_{accept} \stackrel{\text{def}}{=} \bigvee_{1 \le i,j \le n^k} x_{i,j,q_{accept}}$$

Note: ϕ_{accept} is of polynomial size w.r.t. to n.

Definition of ϕ_{move}

Every window in the table is legal.

Let LW denote the set of all legal windows.

$$\phi_{move} \stackrel{\text{def}}{=} \bigwedge_{1 \leq i < n^k, 1 < j < n^k} \text{legal-window}(i, j)$$

Definition of ϕ_{move}

Every window in the table is legal.

Let LW denote the set of all legal windows.

$$\phi_{move} \stackrel{\text{def}}{=} \bigwedge_{1 \leq i < n^k, 1 < j < n^k} \text{legal-window}(i, j)$$

$$\text{legal-window}(i, j) \stackrel{\text{def}}{=}$$

$$\bigvee_{\frac{a_1, a_2, a_3}{a_4, a_5, a_6} \in LW} x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6}$$

Definition of ϕ_{move}

Every window in the table is legal.

Let LW denote the set of all legal windows.

$$\phi_{move} \stackrel{\text{def}}{=} \bigwedge_{1 \leq i < n^k, 1 < j < n^k} \text{legal-window}(i, j)$$

$$\text{legal-window}(i, j) \stackrel{\text{def}}{=}$$

$$\bigvee_{\frac{a_1, a_2, a_3}{a_4, a_5, a_6} \in LW} x_{i,j-1,a_1} \wedge x_{i,j,a_2} \wedge x_{i,j+1,a_3} \wedge x_{i+1,j-1,a_4} \wedge x_{i+1,j,a_5} \wedge x_{i+1,j+1,a_6}$$

Note: ϕ_{move} is of polynomial size w.r.t. to n.

Cook-Levin Theorem — Summary

- Let $A \in \mathrm{NP}$ be decided by poly-time nondeterministic TM M.
- For every $w \in \Sigma^*$ we constructed in polynomial time a formula $\phi \stackrel{\text{def}}{=} \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$.
- We argued that M accepts w if and only if ϕ is satisfiable.
- Hence SAT is NP-hard.
- Clearly SAT is in NP.
- Conclusion: SAT is NP-complete.

Cook-Levin Theorem — Summary

- Let $A \in \mathrm{NP}$ be decided by poly-time nondeterministic TM M.
- For every $w \in \Sigma^*$ we constructed in polynomial time a formula $\phi \stackrel{\mathrm{def}}{=} \phi_{cell} \wedge \phi_{start} \wedge \phi_{accept} \wedge \phi_{move}$.
- We argued that M accepts w if and only if ϕ is satisfiable.
- Hence SAT is NP-hard.
- Clearly SAT is in NP.
- Conclusion: SAT is NP-complete.

Corollary

The language *3SAT* is NP-complete.

Proof: A small modification of the proof for SAT. $\square$

Exam Questions

- Definition of NP-completeness.
- Theorems about NP-completeness.
- *SAT* and *3SAT* are NP-complete.