Computability and Complexity

Lecture 12

More NP-complete Problems

given by Jiri Srba

Summary of What We Know

Definition (Polynomial Time Reducibility)

We write $A \leq_P B$ iff there is a polynomial time computable function f such that for any input w we have $w \in A$ iff $f(w) \in B$.

Summary of What We Know

Definition (Polynomial Time Reducibility)

We write $A \leq_P B$ iff there is a polynomial time computable function f such that for any input w we have $w \in A$ iff $f(w) \in B$.

Definition (NP-Completeness)

A language B is NP-complete iff $B \in \mathrm{NP}$ (containment in NP) and for every $A \in \mathrm{NP}$ we have $A \leq_P B$ (NP-hardness).

Summary of What We Know

Definition (Polynomial Time Reducibility)

We write $A \leq_P B$ iff there is a polynomial time computable function f such that for any input w we have $w \in A$ iff $f(w) \in B$.

Definition (NP-Completeness)

A language B is NP-complete iff $B \in \text{NP}$ (containment in NP) and for every $A \in \text{NP}$ we have $A \leq_P B$ (NP-hardness).

Facts: *SAT* and *CNF-SAT* are NP-complete (last lecture).

Theorem

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

NP-Completeness of *3SAT*

Boolean Formula in cnf

$\phi = C_1 \land C_2 \land \ldots \land C_k$ where every C_i, $1 \leq i \leq k$ is a disjunction of number of literals

Boolean Formula in 3-cnf

$\phi = C_1 \land C_2 \land \ldots \land C_k$ where every C_i, $1 \leq i \leq k$ is a disjunction of exactly 3 literals

NP-Completeness of *3SAT*

Boolean Formula in cnf

$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ where every C_i, $1 \leq i \leq k$ is a disjunction of number of literals

Boolean Formula in 3-cnf

$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ where every C_i, $1 \leq i \leq k$ is a disjunction of exactly 3 literals

$\mathit{CNF\text{-}SAT} \stackrel{\mathrm{def}}{=} \{\langle\phi\rangle \mid \phi \text{ is a satisfiable Boolean formula in cnf }\}$

$\mathit{3SAT} \stackrel{\mathrm{def}}{=} \{\langle\phi\rangle \mid \phi \text{ is a satisfiable Boolean formula in 3-cnf }\}$

NP-Completeness of *3SAT*

Boolean Formula in cnf

$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ where every C_i, $1 \leq i \leq k$ is a disjunction of number of literals

Boolean Formula in 3-cnf

$\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ where every C_i, $1 \leq i \leq k$ is a disjunction of exactly 3 literals

$\mathit{CNF\text{-}SAT} \stackrel{\text{def}}{=} \{\langle\phi\rangle \mid \phi$ is a satisfiable Boolean formula in cnf $\}$

$\mathit{3SAT} \stackrel{\text{def}}{=} \{\langle\phi\rangle \mid \phi$ is a satisfiable Boolean formula in 3-cnf $\}$

Theorem

$\mathit{CNF\text{-}SAT} \leq_P \mathit{3SAT}$

Corollary

3SAT in NP-complete.

Proof: *CNF-SAT* $\leq_P$ *3SAT*

- Assume a given formula $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ in cnf.
- We construct in poly-time a formula $\phi' = C_1' \wedge C_2' \wedge \ldots \wedge C_k'$ in 3-cnf such that ϕ is satisfiable if and only if ϕ' is satisfiable.

Proof: *CNF-SAT* $\leq_P$ *3SAT*

- Assume a given formula $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ in cnf.
- We construct in poly-time a formula $\phi' = C'_1 \wedge C'_2 \wedge \ldots \wedge C'_k$ in 3-cnf such that ϕ is satisfiable if and only if ϕ' is satisfiable.
- Every clause $C_i =$

$$(\ell_1 \vee \ell_2 \vee \ldots \vee \ell_m)$$

is transformed into conjunction of clauses $C'_i =$

$$(\ell_1 \vee \ell_2 \vee z_1) \wedge (\overline{z_1} \vee \ell_3 \vee z_2) \wedge (\overline{z_2} \vee \ell_4 \vee z_3) \wedge (\overline{z_3} \vee \ell_5 \vee z_4) \wedge \ldots$$
$$\ldots (\overline{z_{m-3}} \vee \ell_{m-1} \vee \ell_m)$$

where $z_1, \ldots, z_{m-3}$ are new (fresh) variables.

- Assume a given formula $\phi = C_1 \land C_2 \land \ldots \land C_k$ in cnf.
- We construct in poly-time a formula $\phi' = C'_1 \land C'_2 \land \ldots \land C'_k$ in 3-cnf such that ϕ is satisfiable if and only if ϕ' is satisfiable.
- Every clause $C_i =$

$$(\ell_1 \lor \ell_2 \lor \ldots \lor \ell_m)$$

is transformed into conjunction of clauses $C'_i =$

$$(\ell_1 \lor \ell_2 \lor z_1) \land (\overline{z_1} \lor \ell_3 \lor z_2) \land (\overline{z_2} \lor \ell_4 \lor z_3) \land (\overline{z_3} \lor \ell_5 \lor z_4) \land \ldots$$
$$\ldots (\overline{z_{m-3}} \lor \ell_{m-1} \lor \ell_m)$$

where $z_1, \ldots, z_{m-3}$ are new (fresh) variables.

Proof: *CNF-SAT* $\leq_P$ *3SAT*

- Assume a given formula $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ in cnf.
- We construct in poly-time a formula $\phi' = C'_1 \wedge C'_2 \wedge \ldots \wedge C'_k$ in 3-cnf such that ϕ is satisfiable if and only if ϕ' is satisfiable.
- Every clause $C_i =$

$$(\ell_1 \vee \ell_2 \vee \ldots \vee \ell_m)$$

is transformed into conjunction of clauses $C'_i =$

$$(\ell_1 \vee \ell_2 \vee z_1) \wedge (\overline{z_1} \vee \ell_3 \vee z_2) \wedge (\overline{z_2} \vee \ell_4 \vee z_3) \wedge (\overline{z_3} \vee \ell_5 \vee z_4) \wedge \ldots$$
$$\ldots (\overline{z_{m-3}} \vee \ell_{m-1} \vee \ell_m)$$

where $z_1, \ldots, z_{m-3}$ are new (fresh) variables.

Proof: *CNF-SAT* $\leq_P$ *3SAT*

- Assume a given formula $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ in cnf.
- We construct in poly-time a formula $\phi' = C_1' \wedge C_2' \wedge \ldots \wedge C_k'$ in 3-cnf such that ϕ is satisfiable if and only if ϕ' is satisfiable.
- Every clause $C_i =$

$$(\ell_1 \vee \ell_2 \vee \ldots \vee \ell_m)$$

is transformed into conjunction of clauses $C_i' =$

$$(\ell_1 \vee \ell_2 \vee z_1) \wedge (\overline{z_1} \vee \ell_3 \vee z_2) \wedge (\overline{z_2} \vee \ell_4 \vee z_3) \wedge (\overline{z_3} \vee \ell_5 \vee z_4) \wedge \ldots$$
$$\ldots (\overline{z_{m-3}} \vee \ell_{m-1} \vee \ell_m)$$

where $z_1, \ldots, z_{m-3}$ are new (fresh) variables.

Proof: *CNF-SAT* $\leq_P$ *3SAT*

- Assume a given formula $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$ in cnf.
- We construct in poly-time a formula $\phi' = C_1' \wedge C_2' \wedge \ldots \wedge C_k'$ in 3-cnf such that ϕ is satisfiable if and only if ϕ' is satisfiable.
- Every clause $C_i =$

$$(\ell_1 \vee \ell_2 \vee \ldots \vee \ell_m)$$

is transformed into conjunction of clauses $C_i' =$

$$(\ell_1 \vee \ell_2 \vee z_1) \wedge (\overline{z_1} \vee \ell_3 \vee z_2) \wedge (\overline{z_2} \vee \ell_4 \vee z_3) \wedge (\overline{z_3} \vee \ell_5 \vee z_4) \wedge \ldots$$
$$\ldots (\overline{z_{m-3}} \vee \ell_{m-1} \vee \ell_m)$$

where $z_1, \ldots, z_{m-3}$ are new (fresh) variables.

- Clearly, C_i is satisfiable iff C_i' is satisfiable, the formula ϕ' is in 3-cnf (if fewer variables than 3 in a clause then repeat some literal), and the reduction works in polynomial time. $\square$

NP-Completeness of *CLIQUE*

Theorem

CLIQUE is NP-complete.

Proof: We already know (from previous lectures) that

- *CLIQUE* is in NP, and
- $\mathit{3SAT} \leq_P \mathit{CLIQUE}$.

Because *3SAT* is NP-complete, we conclude that *CLIQUE* is NP-complete too. □

NP-Completeness of *VERTEX-COVER*

Vertex-Cover Problem:

Given an undirected graph G and a number k, is there a subset of nodes of size k s.t. every edge touches at least one of these nodes?

We call such a subset a k-node vertex cover.

Definition of the Language *VERTEX-COVER*

$$\mathit{VERTEX\text{-}COVER} \stackrel{\text{def}}{=} \{\langle G, k\rangle \mid \ G \text{ is a graph with } k\text{-vertex cover}\}$$

NP-Completeness of *VERTEX-COVER*

Vertex-Cover Problem:

Given an undirected graph G and a number k, is there a subset of nodes of size k s.t. every edge touches at least one of these nodes?

We call such a subset a k-node vertex cover.

Definition of the Language *VERTEX-COVER*

$$VERTEX\text{-}COVER \stackrel{\text{def}}{=} \{\langle G, k\rangle \mid G \text{ is a graph with } k\text{-vertex cover}\}$$

Clearly, *VERTEX-COVER* is in NP.

Theorem

$3SAT \leq_P VERTEX\text{-}COVER$

Corollary

VERTEX-COVER is NP-complete.

Proof: *3SAT* $\leq_P$ *VERTEX-COVER*

- Let ϕ be a 3-cnf formula with m variables and p clauses.
- We construct in poly-time an instance $\langle G, k \rangle$ of *VERTEX-COVER* where $k = m + 2p$ and G is given by:

  - For every variable x in ϕ add two nodes labelled with x and $\overline{x}$ and connect them by an edge (variable gadget).
  - For every clause $(\ell_1 \vee \ell_2 \vee \ell_3)$ in ϕ add three nodes labelled with ℓ_1, ℓ_2 and ℓ_3 and connect them by 3 edges so that they form a triangle (clause gadget).
  - Add an edge between any two identically labelled nodes, one from a variable gadget and one from a clause gadget.

- Note that the reduction works in polynomial time and that ϕ is satisfiable iff G has a k-vertex cover. $\square$

NP-Completeness of *HAMPATH*

Theorem

$3SAT \leq_P HAMPATH$

Corollary

HAMPATH is NP-complete.

NP-Completeness of *HAMPATH*

Theorem

3SAT $\leq_P$ *HAMPATH*

Corollary

HAMPATH is NP-complete.

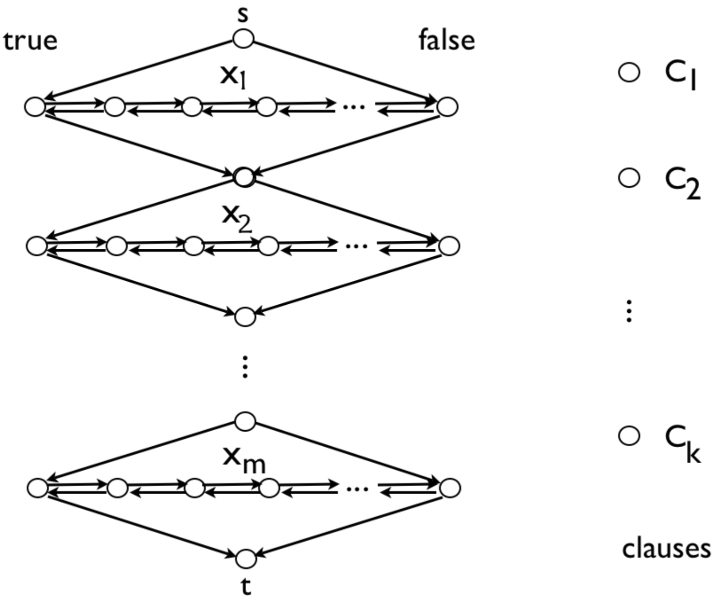
Proof (*3SAT* $\leq_P$ *HAMPATH*): For a given 3-cnf formula

$$\phi = \underbrace{(a_1 \vee b_1 \vee c_1)}_{C_1} \wedge \underbrace{(a_2 \vee b_2 \vee c_2)}_{C_2} \wedge \ldots \wedge \underbrace{(a_k \vee b_k \vee c_k)}_{C_k}$$

over the variables $x_1, x_2, \ldots, x_m$ construct in poly-time a digraph G and nodes s and t such that

ϕ is satisfiable if and only if G has a Hamiltonian path from s to t.

Proof: $3SAT \leq_P HAMPATH$



NP-Completeness of *UHAMPATH*

Definition

$UHAMPATH \stackrel{\text{def}}{=} \{\langle G, s, t\rangle \mid$
G is undirected graph with a Hamiltonian path from s to t $\}$

Theorem

UHAMPATH is NP-complete.

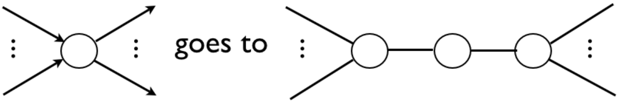
Definition

$UHAMPATH \stackrel{\text{def}}{=} \{\langle G, s, t\rangle \mid$
G is undirected graph with a Hamiltonian path from s to t $\}$

Theorem

UHAMPATH is NP-complete.

Proof: By poly-time reduction from *HAMPATH*. In the reduction from a directed graph to an undirected one, we replace every node with an undirected path of length 2:



NP-Completeness of *SUBSET-SUM*

$SUBSET\text{-}SUM \stackrel{\text{def}}{=} \{\langle S, t\rangle \mid$
$S = \{x_1, \ldots, x_k\} \subseteq \mathbb{N}$ is a multiset, $t \in \mathbb{N}$,
and there is a multiset $X \subseteq S$ s.t. $\sum X = t$ $\}$

Theorem

SUBSET-SUM is NP-complete.

NP-Completeness of *SUBSET-SUM*

SUBSET-SUM $\stackrel{\text{def}}{=} \{\langle S, t\rangle \mid$
$S = \{x_1, \ldots, x_k\} \subseteq \mathbb{N}$ is a multiset, $t \in \mathbb{N}$,
and there is a multiset $X \subseteq S$ s.t. $\sum X = t$ $\}$

Theorem

SUBSET-SUM is NP-complete.

Proof: By poly-time reduction from *3SAT*. For a given 3-cnf formula

$$\phi = \underbrace{(a_1 \vee b_1 \vee c_1)}_{C_1} \wedge \underbrace{(a_2 \vee b_2 \vee c_2)}_{C_2} \wedge \ldots \wedge \underbrace{(a_k \vee b_k \vee c_k)}_{C_k}$$

over the variables x_1, x_2, ..., x_m construct in poly-time a set of numbers S and a number t such that

ϕ is satisfiable iff from S we can select numbers that add up to t.

Proof: *3SAT* $\leq_P$ *SUBSET-SUM*

							C_1	C_2	$\dots$	C_k
x_1	1	0	0	0	$\dots$	0	1	0	$\dots$	0
$\overline{x_1}$	1	0	0	0	$\dots$	0	0	0	$\dots$	1
x_2		1	0	0	$\dots$	0	0	0	$\dots$	1
$\overline{x_2}$		1	0	0	$\dots$	0	1	0	$\dots$	0
x_3			1	0	$\dots$	0	0	0	$\dots$	0
$\overline{x_3}$			1	0	$\dots$	0	1	0	$\dots$	0
$\vdots$										
x_m					$\dots$	1	0	0	$\dots$	0
$\overline{x_m}$					$\dots$	1	0	1	$\dots$	0
							1	0	$\dots$	0
							1	0	$\dots$	0
								1	$\dots$	0
								1	$\dots$	0
										$\vdots$
										1
										1
t	1	1	1	1	$\dots$	1	3	3	$\dots$	3

Summary

Cook-Levin Theorem: *SAT* is NP-complete.

- Because poly-time reducibility ($\leq_P$) is transitive, all languages below are NP-hard.
- All languages below belong to NP, so they are NP-complete.