NP-completeness

CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Fall 2019

Chinese University of Hong Kong

Polynomial-time reductions

What we say

"INDEPENDENT-SET is at least as hard as CLIQUE"

What does that mean?

We mean

> If CLIQUE cannot be decided by a polynomial-time Turing machine, then neither does INDEPENDENT-SET

> If INDEPENDENT-SET can be decided by a polynomial-time Turing machine, then so does CLIQUE

Similar to the reductions we saw in the past 4-5 lectures, but with the additional restriction of polynomial-time

CLIQUE $= \{\langle G, k \rangle \mid G$ is a graph having a clique of k vertices$\}$

INDEPENDENT-SET $= \{\langle G, k \rangle \mid G$ is a graph having

an independent set of k vertices$\}$

Theorem

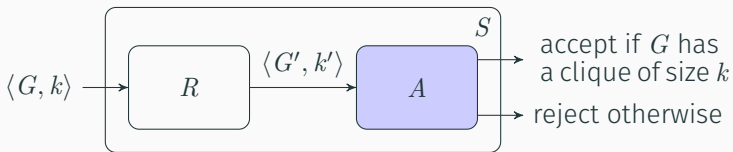If INDEPENDENT-SET has a polynomial-time Turing machine, so does CLIQUE

Polynomial-time reductions

If INDEPENDENT-SET has a polynomial-time Turing machine, so does CLIQUE

Proof

Suppose INDEPENDENT-SET is decided by a poly-time TM A
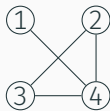
We want to build a TM S that uses A to solve CLIQUE

We look for a polynomial-time Turing machine R that turns the question

"Does G have a clique of size k?"

into

"Does G' have an independent set (IS) of size k'?"

Graph G Graph G'
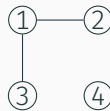
clique of size k $\overset{k=k'}{\longleftrightarrow}$ IS of size k'
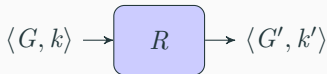
Reducing CLIQUE to INDEPENDENT-SET

On input $\langle G, k \rangle$
  Construct G' by flipping all edges
of G
  Set $k' = k$
  Output $\langle G', k' \rangle$

$$\langle G, k \rangle \longrightarrow \boxed{R} \longrightarrow \langle G', k' \rangle$$

Cliques in G $\longleftrightarrow$ Independent sets in G'

- If G has a clique of size k
  then G' has an independent set of size k

- If G does not have a clique of size k
  then G' does not have an independent set of size k

Reduction recap

We showed that

> If INDEPENDENT-SET is decidable by a polynomial-time Turing machine, so is CLIQUE

by converting any Turing machine for INDEPENDENT-SET into one for CLIQUE

To do this, we came up with a reduction that transforms instances of CLIQUE into ones of INDEPENDENT-SET

Polynomial-time reductions

Language L polynomial-time reduces to L' if

there exists a polynomial-time Turing machine R that takes an instance x of L into an instance y of L' such that
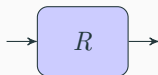
$x \in L$ if and only if $y \in L'$

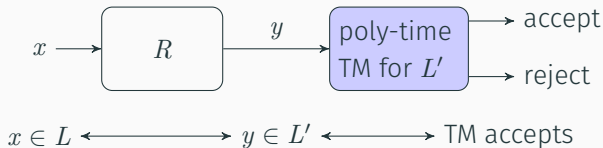| Clique	IS
L	L'
$x = \langle G, k \rangle$	$y = \langle G', k' \rangle$
$x \in L$	$y \in L'$
G has a clique of size k	G' has an IS of size k

$\rightarrow$ R $\rightarrow$

L reduces to L' means L is no harder than L'

If we can solve L', then we can also solve L

Therefore

If L polynomial-time reduces to L' and $L' \in \mathsf{P}$, then $L \in \mathsf{P}$

$x \in L \longleftrightarrow y \in L' \longleftrightarrow$ TM accepts

Pay attention to the direction of reduction

"A is no harder than B" and "B is no harder than A"

have completely different meanings

It is possible that L reduces to L' and L' reduces to L

That means L and L' are as hard as each other

For example, IS and Clique reduce to each other

A boolean formula is an expression made up of variables, ANDs, ORs, and negations, like

$$\varphi = (x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge (\overline{x}_1)$$

Task: Assign TRUE/FALSE values to variables so that the formula evaluates to true

e.g. $x_1 = \mathsf{F}$ $x_2 = \mathsf{F}$ $x_3 = \mathsf{T}$ $x_4 = \mathsf{T}$

Given a formula, decide whether such an assignment exist

SAT $= \{\langle \varphi \rangle \mid \varphi$ is a satisfiable Boolean formula$\}$

3SAT $= \{\langle \varphi \rangle \mid \varphi$ is a satisfiable Boolean formula

conjunctive normal form with 3 literals per clause$\}$

literal: x_i or $\overline{x}_i$

Conjuctive Normal Form (CNF): AND of ORs of literals

3CNF: CNF with 3 literals per clause (repetitions allowed)

$$(\underbrace{\overline{x}_1}_{\text{literal}} \vee x_2 \vee \overline{x}_2) \wedge \underbrace{(\overline{x}_2 \vee x_3 \vee x_4)}_{\text{clause}}$$

$$\varphi = (x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge (\overline{x}_1)$$

Finding a solution:
Try all possible assignments

FFFF	FTFF	TFFF	TTFF
FFFT	FTFT	TFFT	TTFT
FFTF	FTTF	TFTF	TTTF
FFTT	FTTT	TFTT	TTTT

For n variables, there are 2^n possible assignments
Takes exponential time

Verifying a solution:
substitute
$x_1 = \mathsf{F}$ $x_2 = \mathsf{F}$
$x_3 = \mathsf{T}$ $x_4 = \mathsf{T}$
evaluating the formula
$\varphi = (\mathsf{F} \vee \mathsf{T}) \wedge (\mathsf{F} \vee \mathsf{F} \vee \mathsf{T}) \wedge (\mathsf{T})$
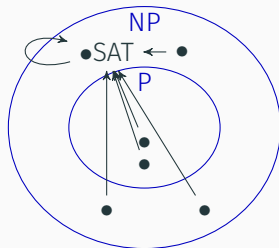can be done in linear time

Every $L \in$ NP polynomial-time reduces to SAT

SAT $= \{\langle \varphi \rangle \mid \varphi$ is a satisfiable Boolean formula$\}$

e.g. $\varphi = (x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3 \vee x_4) \wedge (\overline{x}_1)$

Every problem in NP is no harder than SAT

But SAT itself is in NP, so SAT must be the "hardest problem" in NP
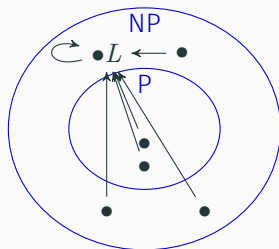
If SAT $\in$ P, then P $=$ NP

A language L is NP-hard if:

For every N in NP, N polynomial-time reduces to L

A language L is NP-complete if L is in NP and L is NP-hard

Cook–Levin theorem

SAT is NP-complete
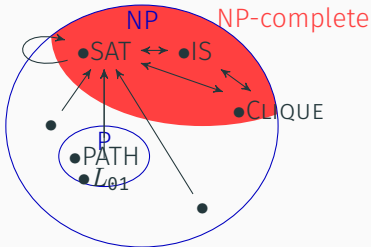
$A \to B$: A polynomial-time reduces to B

In practice, most NP problems are either in P (easy) or NP-complete (probably hard)

Interpretation of Cook–Levin theorem

Optimistic:

If we manage to solve SAT, then we can also solve CLIQUE and many other

Pessimistic:

Since we believe P $\neq$ NP, it is unlikely that we will ever have a fast algorithm for SAT
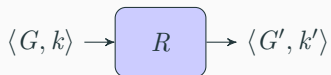
We saw a few examples of NP-complete problems, but there are many more

Surprisingly, most computational problems are either in P or NP-complete

By now thousands of problems have been identified as NP-complete

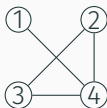$$\langle G, k \rangle \longrightarrow \boxed{R} \longrightarrow \langle G', k' \rangle$$

G has an IS of size k $\longleftrightarrow$ G' has a VC of size k'

Example

Independent sets:
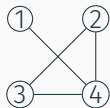
$\emptyset, \{1\}, \{2\}, \{3\}, \{4\},$
$\{1, 2\}, \{1, 3\}$

vertex covers:

$\{2, 4\}, \{3, 4\},$
$\{1, 2, 3\}, \{1, 2, 4\},$
$\{1, 3, 4\}, \{2, 3, 4\},$
$\{1, 2, 3, 4\}$

Claim

S is an independent set if and
only if $\overline{S}$ is a vertex cover

Proof:

S is an independent set

$\Updownarrow$

no edge has both endpoints in S

$\Updownarrow$

every edge has an endpoint in $\overline{S}$

$\Updownarrow$

$\overline{S}$ is a vertex cover

IS	VC
$\emptyset$	$\{1, 2, 3, 4\}$
$\{1\}$	$\{2, 3, 4\}$
$\{2\}$	$\{1, 3, 4\}$
$\{3\}$	$\{1, 2, 4\}$
$\{4\}$	$\{1, 2, 3\}$
$\{1, 2\}$	$\{3, 4\}$
$\{1, 3\}$	$\{2, 4\}$

$$\langle G, k \rangle \longrightarrow \boxed{R} \longrightarrow \langle G', k' \rangle$$
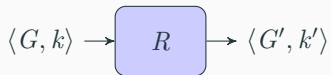
R: On input $\langle G, k \rangle$

Output $\langle G, n - k \rangle$

G has an IS of size k $\longleftrightarrow$ G has a VC of size $n - k$

Overall sequence of reductions:

$$\mathsf{SAT} \to \mathsf{3SAT} \to \mathsf{CLIQUE} \xrightarrow{\checkmark} \mathsf{IS} \xrightarrow{\checkmark} \mathsf{VC}$$

$3\text{SAT} = \{\varphi \mid \varphi \text{ is a satisfiable Boolean formula in 3CNF}\}$

$\text{Clique} = \{\langle G, k \rangle \mid G \text{ is a graph having a clique of } k \text{ vertices}\}$
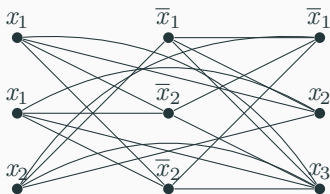
3CNF formula $\varphi \longrightarrow \boxed{R} \longrightarrow \langle G, k \rangle$
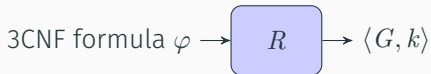
φ is satisfiable $\quad \longleftrightarrow \quad G$ has a clique of size k

Example:

$$\varphi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x}_1 \vee \overline{x}_2 \vee \overline{x}_2) \wedge (\overline{x}_1 \vee x_2 \vee x_3)$$

One vertex for each literal occurrence

One edge for each consistent pair (non-opposite literals)
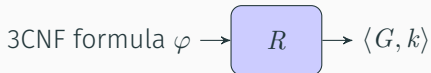
$$\text{3CNF formula } \varphi \longrightarrow \boxed{R} \longrightarrow \langle G, k \rangle$$

R: On input φ, where φ is a 3CNF formula with m clauses

    Construct the following graph G:

        G has $3m$ vertices, divided into m groups

            One for each literal occurrence in φ

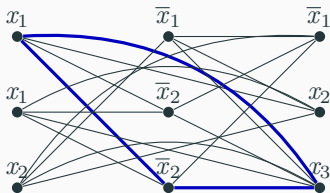   If vertices u and v are in different groups and consistent

            Add an edge (u, v)

    Output $\langle G, m \rangle$

$$\text{3CNF formula } \varphi \longrightarrow \boxed{R} \longrightarrow \langle G, k \rangle$$

φ is satisfiable $\longleftrightarrow$ G has a clique of size m
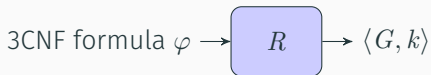

$$\varphi = (\underset{\text{T}}{x_1} \vee \underset{\text{T}}{x_1} \vee \underset{\text{F}}{x_2}) \wedge (\underset{\text{F}}{\overline{x}_1} \vee \underset{\text{T}}{\overline{x}_2} \vee \underset{\text{T}}{\overline{x}_2}) \wedge (\underset{\text{F}}{\overline{x}_1} \vee \underset{\text{F}}{x_2} \vee \underset{\text{T}}{x_3})$$

$$\text{3CNF formula } \varphi \longrightarrow \boxed{R} \longrightarrow \langle G, k \rangle$$

Every satisfying assignment of φ gives a clique of size m in G

Conversely, every clique of size m in G gives a satisfying assignment of φ

Overall sequence of reductions:

$$\text{SAT} \rightarrow \text{3SAT} \xrightarrow{\checkmark} \text{CLIQUE} \xrightarrow{\checkmark} \text{IS} \xrightarrow{\checkmark} \text{VC}$$

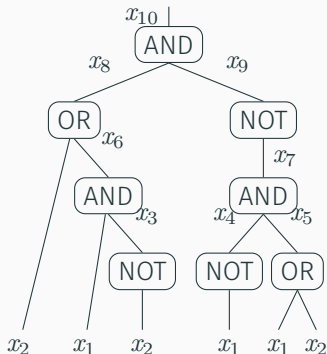$$SAT = \{\varphi \mid \varphi \text{ is a satisfiable Boolean formula}\}$$

e.g. $((x_1 \vee x_2) \wedge \overline{(x_1 \vee x_2)}) \vee \overline{((x_1 \vee (x_2 \wedge x_3)) \wedge \overline{x}_3)}$

$$3SAT = \{\varphi' \mid \varphi' \text{ is a satisfiable 3CNF formula}\}$$

e.g. $(x_1 \vee x_2 \vee x_2) \wedge (x_2 \vee x_3 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee \overline{x}_5)$

Example: $\quad \varphi = (x_2 \vee (x_1 \wedge \overline{x}_2)) \wedge \overline{(\overline{x}_1 \wedge (x_1 \vee x_2))}$

Tree representation of φ
Add extra variable to φ' for
each wire in the tree

Example: $\quad \varphi = (x_2 \vee (x_1 \wedge \overline{x}_2)) \wedge \overline{(\overline{x}_1 \wedge (x_1 \vee x_2))}$
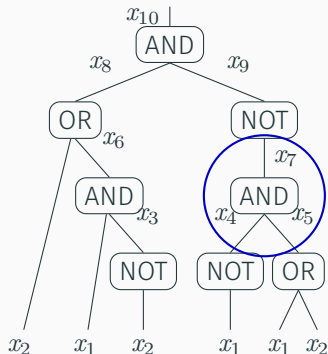
Tree representation of φ
Add extra variable to φ' for
each wire in the tree

Add clauses to φ' for each gate

$x_4 x_5 x_7$			$x_7 = x_4 \wedge x_5$
T	T	T	T
T	T	F	F
T	F	T	F
T	F	F	T
F	T	T	F
F	T	F	T
F	F	T	F
F	F	F	T

Clauses added:
$(\overline{x}_4 \vee \overline{x}_5 \vee x_7) \wedge (\overline{x}_4 \vee x_5 \vee \overline{x}_7)$
$(x_4 \vee \overline{x}_5 \vee \overline{x}_7) \wedge (x_4 \vee x_5 \vee \overline{x}_7)$

Boolean formula $\varphi \longrightarrow \boxed{R} \longrightarrow$ 3CNF formula φ'
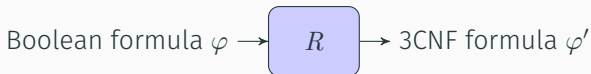
R: On input $\langle \varphi \rangle$, where φ is a Boolean formula

    Construct and output the following 3CNF formula φ'

    φ' has extra variable $x_{n+1}, \ldots, x_{n+t}$

        one for each gate G_j in φ

    For each gate G_j, construct the forumla φ_j

        forcing the output of G_j to be correct given its inputs

    Set $\varphi' = \varphi_{n+1} \wedge \cdots \wedge \varphi_{n+t} \wedge \underbrace{(x_{n+t} \vee x_{n+t} \vee x_{n+t})}_{\text{requires output of } \varphi \text{ to be TRUE}}$

Boolean formula $\varphi \longrightarrow$ $\boxed{R}$ $\longrightarrow$ 3CNF formula φ'
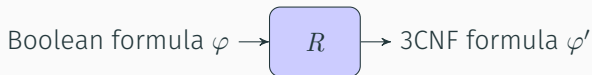
φ satisfiable $\longleftrightarrow$ φ' satisfiable

Every satisfying assignment of φ extends uniquely to a satisfying assignment of φ'

In the other direction, in every satisfying assignment of φ', the $x_1, \ldots, x_n$ part satisfies φ