

NP-completeness

CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Fall 2022

Chinese University of Hong Kong

Polynomial-time reductions

When 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

Polynomial-time reductions

CLIQUE = $\{\langle G, k \rangle \mid \text{Graph } G \text{ has a clique of } k \text{ vertices}\}$

INDEPENDENT-SET = $\{\langle G, k \rangle \mid \text{Graph } G \text{ has having}$
an independent set of k vertices}

Theorem

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

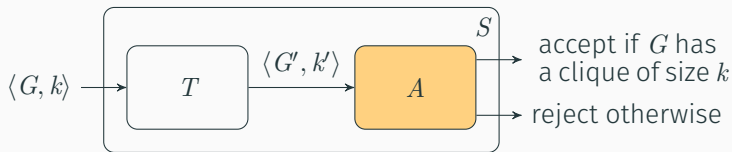
Polynomial-time reductions

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



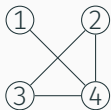
Reducing CLIQUE to INDEPENDENT-SET

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

“Does G have a clique of size k ?”

into

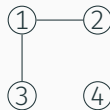
“Does G' have an independent set (IS) of size k' ?”



Graph G

clique of size k

flip all edges
→



Graph G'

IS of size k'

$k=k'$
↔

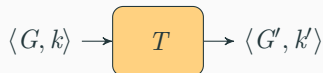
Reducing CLIQUE to INDEPENDENT-SET

On input $\langle G, k \rangle$

Construct G' by flipping all edges of G

Set $k' = k$

Output $\langle G', k' \rangle$



Cliques in $G \iff$ 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 T that takes an instance x of L into an instance y of L' such that

$$x \in L \quad \text{if and only if} \quad y \in L'$$

CLIQUE

IS

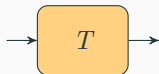
L

L'

$x = \langle G, k \rangle$

$x \in L$

G has a clique of size k



$y = \langle G', k' \rangle$

$y \in L'$

G' has an IS of size k

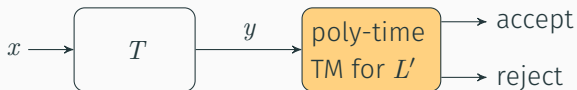
The meaning of reductions

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 $L' \in P$, then $L \in P$



Direction of reduction

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

Boolean formula satisfiability

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

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

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

e.g. $x_1 \mapsto F$ $x_2 \mapsto F$ $x_3 \mapsto T$ $x_4 \mapsto T$

Given a formula, decide whether such an assignment exist

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

3SAT = $\{\langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula}$
 conjunctive normal form with 3 literals per clause}

literal: x_i or \bar{x}_i

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

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

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

3SAT is in NP

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

Finding a solution:

Try all possible assignments

FFFF FTFF TFFF TTF

FFFT FTFT TFFT TTFT

FFTF FTTF TFTF TTF

FFTT FTTT TFTT TTTT

For n variables, there are 2^n
possible assignments

Takes **exponential time**

Verifying a solution:

substitute

$x_1 \mapsto F$ $x_2 \mapsto F$

$x_3 \mapsto T$ $x_4 \mapsto T$

evaluating the formula

$\phi = (F \vee T) \wedge (F \vee F \vee T) \wedge (T)$

can be done in **linear time**

Cook–Levin theorem

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

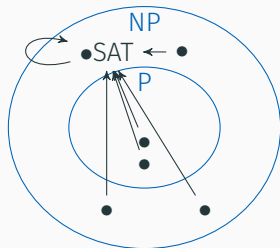
$\text{SAT} = \{ \langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula} \}$

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

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 $\text{SAT} \in \text{P}$, then $\text{P} = \text{NP}$



NP-completeness

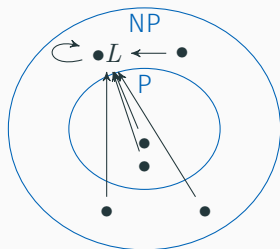
A language L is **NP-hard** if:

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

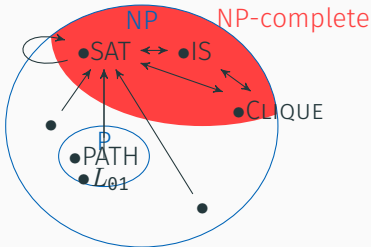
A language L is **NP-complete** if L is in NP and L is NP-hard

Cook–Levin theorem

SAT is NP-complete



Our (conjectured) picture of NP



$A \rightarrow B$: 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

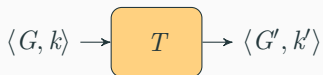
Ubiquity of NP-complete problems

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

Reducing IS to VC

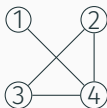


G has an IS of size $k \iff G'$ has a VC of size k'

Example

Independent sets:

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



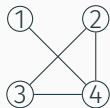
vertex covers:

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

Reducing IS to VC

Claim

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



Proof:

S is an independent set



no edge has both endpoints in S



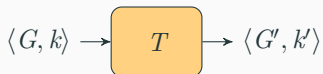
every edge has an endpoint in \bar{S}



\bar{S} is a vertex cover

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

Reducing IS to VC



T : On input $\langle G, k \rangle$

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

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

Overall sequence of reductions:

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

Reducing 3SAT to CLIQUE

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

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

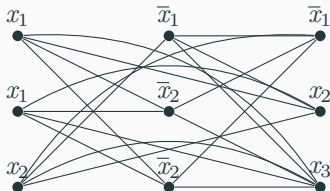


ϕ is satisfiable $\iff G$ has a clique of size k

Reducing 3SAT to CLIQUE

Example:

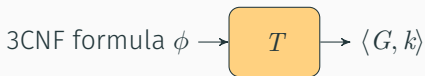
$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$



One vertex for each **literal occurrence**

One edge for each **consistent pair** across different groups
(not opposite literals of the same variable)

Reducing 3SAT to CLIQUE



T: 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

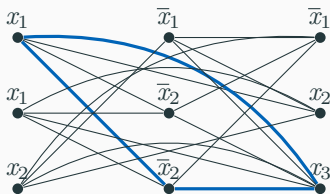
Add an edge (u, v)

Output $\langle G, m \rangle$

Reducing 3SAT to CLIQUE



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



$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee x_2 \vee x_3)$$

$\quad \quad \quad \begin{matrix} \text{T} & \text{T} & \text{F} \\ \text{F} & \text{T} & \text{T} \\ \text{F} & \text{F} & \text{T} \end{matrix}$

Reducing 3SAT to CLIQUE: Summary



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:

SAT \rightarrow 3SAT $\checkmark \rightarrow$ CLIQUE $\checkmark \rightarrow$ IS $\checkmark \rightarrow$ VC

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

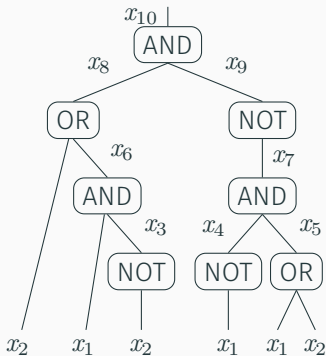
e.g. $((x_1 \vee x_2) \wedge \overline{(x_1 \vee x_2)}) \vee \overline{((x_1 \vee (x_2 \wedge x_3)) \wedge \bar{x}_3)}$

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

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

Reducing SAT to 3SAT

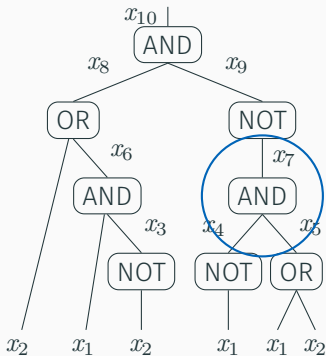
Example: $\phi = (x_2 \vee (x_1 \wedge \bar{x}_2)) \wedge \overline{(\bar{x}_1 \wedge (x_1 \vee x_2))}$



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

Reducing SAT to 3SAT

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



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

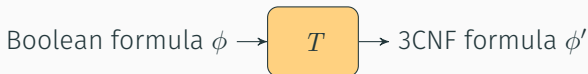
Add clauses to ϕ' for each gate

$x_4 x_5 x_7$	$x_7 = x_4 \wedge x_5$
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:

$$\begin{aligned} &(\bar{x}_4 \vee \bar{x}_5 \vee x_7) \wedge (\bar{x}_4 \vee x_5 \vee \bar{x}_7) \\ &(x_4 \vee \bar{x}_5 \vee \bar{x}_7) \wedge (x_4 \vee x_5 \vee \bar{x}_7) \end{aligned}$$

Reducing SAT to 3SAT



T: On input $\langle \phi \rangle$, where ϕ is a Boolean formula

Construct and output the following 3CNF formula ϕ'

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

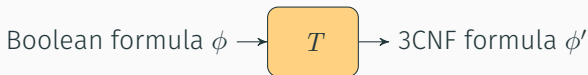
one for each gate G_j in ϕ

For each gate G_j , construct the formula ϕ_j

forcing the output of G_j to be correct given its inputs

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

Reducing SAT to 3SAT



$$\phi \text{ satisfiable} \iff \phi' \text{ satisfiable}$$

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, \dots, x_n part satisfies ϕ