NP-completeness

CSCI 3130 Formal Languages and Automata Theory

Siu On CHAN

Chinese University of Hong Kong

Fall 2017
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
Polynomial-time reductions

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

\[ \text{INDEPENDENT-SET} = \{\langle G, k \rangle \mid G \text{ is a graph having an independent set of } k \text{ vertices}\} \]

Theorem

If \text{INDEPENDENT-SET} has a polynomial-time Turing machine, so does \text{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

Given $\langle G, k \rangle$, $R$ runs $A$ on $\langle G', k' \rangle$ and $S$ accepts if $G$ has a clique of size $k$. Otherwise, $S$ rejects.
Reducing CLIQUE to INDEPENDENT-SET

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 $G$

clique of size $k$

Graph $G'$

IS of size $k'$

flip all edges
Reducing CLIQUE to INDEPENDENT-SET

On input \( \langle G, k \rangle \)

- Construct \( G' \) by flipping all edges of \( G \)
- Set \( k' = k \)
- Output \( \langle G', k' \rangle \)

\[ \langle G, k \rangle \xrightarrow{R} \langle G', k' \rangle \]

Cliquess in \( G \) \quad \leftrightarrow \quad \text{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 \( k \)
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 $y \in L'$

<table>
<thead>
<tr>
<th>CLIQUE</th>
<th>IS</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L$</td>
<td>$L'$</td>
</tr>
</tbody>
</table>

$x = \langle G, k \rangle$

$x \in L$

$G$ 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 $L$ reduces to $L'$ and $L' \in P$, then $L \in P$

\[ x \xrightarrow{R} y \]  
\[ x \in L \]  
\[ y \in L' \]  
\[ \text{TM accepts} \]
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

\[ \varphi = (x_1 \lor \overline{x}_2) \land (x_2 \lor \overline{x}_3 \lor x_4) \land (\overline{x}_1) \]

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

e.g. \( x_1 = F \quad x_2 = F \quad x_3 = T \quad x_4 = T \)

Given a formula, decide whether such an assignment exist
3SAT

\[ \text{SAT} = \{ \langle \varphi \rangle \mid \varphi \text{ is a satisfiable Boolean formula} \} \]

\[ \text{3SAT} = \{ \langle \varphi \rangle \mid \varphi \text{ is a satisfiable Boolean formulaconjunctive normal form with 3 literals per clause} \} \]

literal: \( x_i \) or \( \overline{x}_i \)

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

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

\[
(\overline{x}_1 \lor x_2 \lor \overline{x}_2) \land (\overline{x}_2 \lor x_3 \lor x_4)
\]
3SAT is in NP

\[ \varphi = (x_1 \lor \overline{x}_2) \land (x_2 \lor \overline{x}_3 \lor x_4) \land (\overline{x}_1) \]

Finding a solution:
Try all possible assignments
- FFFF
  - FTFF
  - TFFF
  - TTFF
- FFFT
  - FTFT
  - TFFT
  - TTFT
- FFTF
  - FTTF
  - TFTF
  - TTTF
- FTTT
  - FTTT
  - TFTT
  - TTTT

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

Verifying a solution:
substitute
- \( x_1 = F \)
- \( x_2 = F \)
- \( x_3 = T \)
- \( x_4 = T \)
evaluating the formula
\[ \varphi = (F \lor T) \land (F \lor F \lor T) \land (T) \]
can be done in linear time
Cook–Levin theorem

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

\[
\text{SAT} = \{ \langle \varphi \rangle \mid \varphi \text{ is a satisfiable Boolean formula} \}
\]
e.g. $\varphi = (x_1 \lor \overline{x}_2) \land (x_2 \lor \overline{x}_3 \lor x_4) \land (\overline{x}_1)$

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 $P = \text{NP}$
NP-completeness

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

For every $N$ in NP, $N$ 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 picture of NP

\[ A \rightarrow B: A \text{ reduces to } B \]

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 ≠ 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

\[ \langle G, k \rangle \rightarrow R \rightarrow \langle G', k' \rangle \]

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

**Example**

Independent sets:

\( \emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{1, 3\} \)

```
1 2
\( \begin{array}{c}
1 \\
3 \\
4 \\
\end{array} \)
```

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 $\overline{S}$ is a vertex cover

Proof:

$S$ is an independent set

\[ \iff \]

no edge has both endpoints in $S$

\[ \iff \]

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

\[ \iff \]

$\overline{S}$ is a vertex cover

<table>
<thead>
<tr>
<th>IS</th>
<th>VC</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset$</td>
<td>${1, 2, 3, 4}$</td>
</tr>
<tr>
<td>${1}$</td>
<td>${2, 3, 4}$</td>
</tr>
<tr>
<td>${2}$</td>
<td>${1, 3, 4}$</td>
</tr>
<tr>
<td>${3}$</td>
<td>${1, 2, 4}$</td>
</tr>
<tr>
<td>${4}$</td>
<td>${1, 2, 3}$</td>
</tr>
<tr>
<td>${1, 2}$</td>
<td>${3, 4}$</td>
</tr>
<tr>
<td>${1, 3}$</td>
<td>${2, 4}$</td>
</tr>
</tbody>
</table>
Reducing IS to VC

\[ \langle G, k \rangle \rightarrow R \rightarrow \langle G', k' \rangle \]

\( R \): 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 \( \rightarrow \) IS \( \rightarrow \) VC
Reducing 3SAT to CLIQUE

\[ 3\text{SAT} = \{ \varphi \mid \varphi \text{ is a satisfiable Boolean formula in 3CNF} \} \]
\[ \text{CLIQUE} = \{ \langle G, k \rangle \mid G \text{ is a graph having a clique of } k \text{ vertices} \} \]

3CNF formula \( \varphi \) \( \overset{R}{\rightarrow} \) \( \langle G, k \rangle \)

\( \varphi \) is satisfiable \( \iff \) \( G \) has a clique of size \( k \)
Reducing 3SAT to CLIQUE

Example:

\[ \varphi = (x_1 \lor x_1 \lor x_2) \land (\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_2) \land (\overline{x}_1 \lor x_2 \lor x_3) \]

One vertex for each literal occurrence

One edge for each consistent pair (non-opposite literals)
Reducing 3SAT to CLIQUE

\[ R: \text{On input } \varphi, \text{where } \varphi \text{ is a 3CNF formula with } m \text{ clauses} \]

\textbf{Construct} the following graph } G: 

- \( G \) has \( 3m \) vertices, divided into \( m \) groups
- One for each literal occurrence in \( \varphi \)
- If vertices \( u \) and \( v \) are in different groups and consistent
  - Add an edge \((u, v)\)

\textbf{Output } \langle G, m \rangle
Reducing 3SAT to CLIQUE

3CNF formula $\varphi \rightarrow R \rightarrow \langle G, k \rangle$

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

$\varphi = (x_1 \lor x_1 \lor x_2) \land (\overline{x}_1 \lor \overline{x}_2 \lor \overline{x}_2) \land (\overline{x}_1 \lor x_2 \lor x_3)$
Reducing 3SAT to CLIQUE: Summary

3CNF formula $\varphi \rightarrow R \rightarrow \langle G, k \rangle$

Every satisfying assignment of $\varphi$ gives a clique of size $m$ in $G$

Conversely, every clique of size $m$ in $G$ gives a satisfying assignment of $\varphi$

Overall sequence of reductions:

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

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

e.g. \quad (x_1 \lor x_2) \land \overline{(x_1 \lor x_2)} \lor ((x_1 \lor (x_2 \land x_3)) \land \overline{x_3})

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

e.g. \quad (x_1 \lor x_2 \lor x_3) \land (x_2 \lor x_3 \lor \overline{x_4}) \land (x_2 \lor \overline{x_3} \lor \overline{x_5})
Reducing SAT to 3SAT

Example: \( \varphi = (x_2 \lor (x_1 \land \overline{x}_2)) \land (\overline{x}_1 \land (x_1 \lor x_2)) \)

Tree representation of \( \varphi \)

Add extra variable to \( \varphi' \) for each wire in the tree
Reducing SAT to 3SAT

Example: \( \varphi = (x_2 \lor (x_1 \land \overline{x_2})) \land (\overline{x_1} \land (x_1 \lor x_2)) \)

Tree representation of \( \varphi \)

Add extra variable to \( \varphi' \) for each wire in the tree

Add clauses to \( \varphi' \) for each gate

<table>
<thead>
<tr>
<th>( x_4x_5x_7 )</th>
<th>( x_7 = x_4 \land x_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T T T</td>
<td>T</td>
</tr>
<tr>
<td>T T F</td>
<td>F</td>
</tr>
<tr>
<td>T F T</td>
<td>F</td>
</tr>
<tr>
<td>T F F</td>
<td>T</td>
</tr>
<tr>
<td>F T T</td>
<td>F</td>
</tr>
<tr>
<td>F T F</td>
<td>T</td>
</tr>
<tr>
<td>F F T</td>
<td>F</td>
</tr>
<tr>
<td>F F F</td>
<td>T</td>
</tr>
</tbody>
</table>

Clauses added:

\[
(x_4 \lor \overline{x_5} \lor x_7) \land (x_4 \lor x_5 \lor \overline{x_7})
\]

\[
(x_4 \lor \overline{x_5} \lor \overline{x_7}) \land (x_4 \lor x_5 \lor \overline{x_7})
\]
Reducing SAT to 3SAT

Boolean formula $\varphi \xrightarrow{R} 3$CNF formula $\varphi'$

$R$: On input $\langle \varphi \rangle$, where $\varphi$ is a Boolean formula

Construct and output the following 3CNF formula $\varphi'$

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

one for each gate $G_j$ in $\varphi$

For each gate $G_j$, construct the formula $\varphi_j$

forcing the output of $G_j$ to be correct given its inputs

Set $\varphi' = \varphi_{n+1} \land \cdots \land \varphi_{n+t} \land (x_{n+t} \lor x_{n+t} \lor x_{n+t})$

requires output of $\varphi$ to be TRUE
Reducing SAT to 3SAT

Boolean formula $\varphi \xrightarrow{R} 3$CNF formula $\varphi'$

$\varphi$ satisfiable $\iff \varphi'$ satisfiable

Every satisfying assignment of $\varphi$ extends uniquely to a satisfying assignment of $\varphi'$

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