COSC 341
Theory of Computing
Lecture 20
Reductions from SAT (3-SATand INDEPENDENT-SET)

Stephen Cranefield
stephen.cranefield@otago.ac.nz

Keywords: k-SAT, 3-SAT

Satisfiability with bounded length clauses

As seen in the Cook-Levin theorem, a SAT instance can have very large clauses. That makes it hard to see how we could reduce SAT to other problems to show they are $\mathbf{NP}$-complete.

It turns out that restricting ourselves to three literals in each clause still gives us an $\mathbf{NP}$-complete problem (3-SAT).

k-SAT

Instance: A formula in CNF over a set of variables V with exactly k distinct literals in each clause.
Problem: Does the formula have a satisfying assignment?

Why "exactly k" rather than "at most k"? It doesn't matter.

Suppose $k = 3$ and we want to write a clause with two variables, e.g. $a \vee \neg b$. How can we extend this to three literals without changing the meaning?

We can just introduce a new variable (not to be used anywhere else), e.g. c. We can replace $a \vee \neg b$ with:

$$a \vee \neg b \vee c$$
$$a \vee \neg b \vee \neg c$$

What is the complexity of k-SAT?

Theorem

For $k \geq 3$, k-SAT is $\mathbf{NP}$-complete. For $k < 3$ it is in $\mathbf{P}$.

This is easy to see for 1-SAT. A set of 1-variable clauses is unsatisfiable if and only if some variable and its negation both occur.

Satisfiable	**Unsatisfiable**
a	a
b	b
c	c
a	$\neg a$
b	b

This is easily resolved in polynomial time.

Proof for the case $k = 2$

We can reduce an instance of 2-SAT with n variables to one using $n-1$ variables in polynomial time. Repeat until there is only one variable (a simple case to solve).

Let C_I denote the clauses in I. Consider the set:

$$P = \{\text{`}z\text{'} : \text{`}x_n \vee z\text{'} \in C_I \text{ and } z \text{ is } x_i \text{ or } \neg x_i \text{ for } i < n\}$$

(P means "positive for x_n)

Suppose P contains both x_i and $\neg x_i$ for some i. Then any satisfying assignment must have $x_n = \mathtt{t}$. We can remove x_n from each clause in P, giving us a set P' of $(n-1)$-variable clauses. Our next iteration starts with $(C_I \setminus P) \cup P'$.

Otherwise, try a similar approach with:

$$N = \{\text{`}z\text{'} : \text{`}\neg x_n \vee z\text{'} \in C_I \text{ and } z \text{ is } x_i \text{ or } \neg x_i \text{ for } i < n\}$$

If that fails, . . .

Proof for the case $k = 2$ (continued)

If the previous two checks fail, Notes 18 shows that for I to be satisfiable, this $(n-1)$-variable formula must be too:

$$\left(\left(\bigwedge_{w \in N} w\right) \vee \left(\bigwedge_{z \in P} z\right)\right) \vee \left(\bigwedge_{\text{clauses in } C_I \text{ not involving } x_n} C\right)$$

This is not in CNF, but the first two parts can be converted to a conjunction of $|N||P|$ clauses each of size 2 in the remaining variables, and the remaining has size no larger than $|I|$.

Keep removing variables until we have only one variable left, which gives us only two truth assignments to check.

Proof for the case $k = 3$ (3-SAT is $\mathbf{NP}$-complete)

We reduce SAT to 3-SAT by replacing each 'big' clause:

$$l_1 \vee l_2 \vee \cdots \vee l_n$$

(where $n \geq 4$) with $n-2$ clauses of size 3 that include new variables z_3 to z_{n-1}:

$$\begin{gathered}
l_1 \vee l_2 \vee z_3 \\
\neg z_3 \vee l_3 \vee z_4 \\
\neg z_4 \vee l_4 \vee z_5 \\
\cdots \\
\neg z_{n-2} \vee l_{n-2} \vee z_{n-1} \\
\neg z_{n-1} \vee l_{n-1} \vee l_n .
\end{gathered}$$

Proof for the case $k = 3$ (3-SAT is $\mathbf{NP}$-complete)

We reduce SAT to 3-SAT by replacing each 'big' clause:

$$l_1 \vee l_2 \vee \cdots \vee l_n$$

(where $n \geq 4$) with $n - 2$ clauses of size 3 that include new variables z_3 to z_{n-1}:

$$\begin{gathered} l_1 \vee l_2 \vee z_3 \\ \neg z_3 \vee l_3 \vee z_4 \\ \neg z_4 \vee l_4 \vee z_5 \\ \cdots \\ \neg z_{n-2} \vee l_{n-2} \vee z_{n-1} \\ \neg z_{n-1} \vee l_{n-1} \vee l_n. \end{gathered}$$

<u>If the original clause is unsatisfiable</u> (all l_i are false) that forces z_3 to z_{n-1} to be true, but then the last clause will be false, so the 3-SAT instance is unsatisfiable.

Proof for the case $k = 3$ (3-SAT is $\mathbf{NP}$-complete)

We reduce SAT to 3-SAT by replacing each 'big' clause:

$$l_1 \vee l_2 \vee \cdots \vee l_n$$

(where $n \geq 4$) with $n-2$ clauses of size 3 that include new variables z_3 to z_{n-1}:

$$\begin{gathered} l_1 \vee l_2 \vee z_3 \\ \neg z_3 \vee l_3 \vee z_4 \\ \neg z_4 \vee l_4 \vee z_5 \\ \cdots \\ \neg z_{n-2} \vee l_{n-2} \vee z_{n-1} \\ \neg z_{n-1} \vee l_{n-1} \vee l_n. \end{gathered}$$

If the original clause is satisfiable, at least one l_i is true. There are three cases for showing the 3-SAT instance is satisfiable (see Notes 18). One case: $3 \leq i \leq n-2$, e.g. $i = 4$. Set $z_j = \mathsf{t}$ for $j \leq i$ and $z_j = \mathsf{f}$ for $j > i$. This satisfies all the clauses.

Proof for the case $k > 3$

We can reduce 3-SAT to 4-SAT, 4-SAT to 5-SAT and so on. Just duplicate a literal in each clause to produce the larger instance.

Some reductions from 3-SAT

Now we know that 3-SAT is $\mathbf{NP}$-complete, it turns out to be quite handy as a source of reductions to other problems.

Three examples in Notes 18:

- 3-SAT to INDEPENDENT-SET (covered today).
- 3-SAT to 3-COLOURING (next lecture).
- 3-SAT to HAMILTON-CYCLE (next lecture).

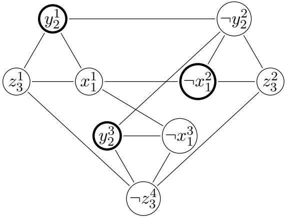
3-SAT to INDEPENDENT-SET

Reduction needed (in polynomial time):

INPUT: A set of clauses each containing exactly three literals
OUTPUT: A graph G and a parameter k such that the clauses are satisfiable if and only if G has a k-element independent set.

- We can choose k to be the number of clauses in the 3-SAT instance.
- For each clause $l_1 \vee l_2 \vee l_3$, include a triangle in G with node labels $l^{\text{clause\_num}}_{\text{lit\_num}}$, giving a total of $3k$ vertices. Variables in no clauses will not appear in the graph.
- Then add an edge between each pair of vertices representing contradictory literals (i.e., a variable and its negation). For any satisfying assignment, e.g. $(x, y, z) = (\mathtt{f}, \mathtt{t}, \mathtt{t})$ below, choosing one satisfied literal from each triangle gives a k-element independent set.

$$\begin{array}{c} x \vee y \vee z \\ \neg x \vee \neg y \vee z \\ \neg x \vee y \vee \neg z \end{array} \quad \Rightarrow$$



3-SAT to INDEPENDENT-SET (continued)

- Conversely, if the graph has a k-element independent set, it must include exactly one vertex from each triangle. This must be a valid truth assignment as we can't have both a variable and its negation in an independent set (they are connected by edges). Any unassigned variables can be given a arbitrary value. This gives us a truth assignment with one true literal per clause, so the 3-SAT instance is satisfiable.