CSCI 1590
Intro to Computational Complexity
NP-Complete Languages

John E. Savage

Brown University

February 6, 2008

1. Definition of **NP**-Complete Languages
2. NAESAT is **NP**-complete
3. 0-1 INTEGER PROGRAMMING is **NP**-complete
4. INDEPENDENT SET is **NP**-complete
5. CLIQUE is **NP**-complete

Definition of **NP**-complete Languages

Definition

A language L_2 is **NP**-complete if a) L_2 is in **NP** and b) for every language L_1 in **NP** there is polynomial-time reduction from it to L_2 (L_2 is **NP-hard**.)

$f : \Sigma_1^* \mapsto \Sigma_2^*$ is a **reduction** from L_1 to L_2 if $x \in L_1$ if and only if $f(x) \in L_2$. f is a **polynomial-time reduction** if it can also be computed by a DTM in polynomial time in the length of its input.

A language L_1 is in **NP** if there is an NTM M_1 that accepts every $\mathbf{x} \in L_1$ in $p(n)$ steps where $n = |\mathbf{x}|$ and $p(n)$ is a polynomial and does not accept any $\mathbf{x} \notin L_1$.

Three **NP**-Complete Languages

CIRCUIT SAT

Instance: A circuit description with n input variables $\{x_1, x_2, \ldots, x_n\}$ for some integer n and a designated output gate.

Answer: "Yes" if there is an assignment of values to the variables such that the output of the circuit has value 1.

SAT

Instance: Literals $X = \{x_1, \overline{x}_1, x_2, \overline{x}_2, \ldots, x_n, \overline{x}_n\}$, and clauses $C = (c_1, c_2, \ldots, c_m)$ where each clause c_i is a subset of X.

Answer: "Yes" if for some assignment of Booleans to variables in $\{x_1, x_2, \ldots, x_n\}$, at least one literal in each clause has value 1.

3-SAT

Instance: SAT instance which has at most three literals/clause.

Answer: "Yes" if for some assignment of Booleans to variables in $\{x_1, x_2, \ldots, x_n\}$, at least one literal in each clause has value 1.

Method for Showing a Language L_2 is **NP**-Complete

- Show that the language L_2 is in **NP**.
- Give **P**-time reduction from **NP**-complete language L_1 to L_2.

Proof that NAESAT is **NP**-complete

NAESAT

Instance: An instance of 3-SAT.

Answer: "Yes" if each clause is satisfiable when not all literals have the same value.

Theorem

NAESAT *is* **NP**-*complete.*

Proof

Use the reduction from CIRCUIT SAT to 3-SAT. However, first convert the circuit from AND, OR, and NOT to NAND. Replace a step computing $f_{\text{NAND}}(x, y, z) = (z = \text{NAND}(x, y))$ by its CNF, which is $(x \vee y \vee z) \wedge (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$.
Combining the first and second (first and third) disjuncts gives $(x \vee z)$ $((y \vee z))$ and $f_{\text{NAND}}(x, y, z) = (x \vee z) \wedge (y \vee z) \wedge (\bar{x} \vee \bar{y} \vee \bar{z})$.

Proof that NAESAT is **NP**-complete

Proof (cont.)

In polynomial time produce clauses C by reducing an SLP having NANDs to a set of clauses equivalent to $f_{\text{NAND}}(x, y, z)$. When $f_{\text{NAND}}(x, y, z)$ is True, the only clauses in which literals don't have two different values in all cases are $(x \vee z)$ and $(y \vee z)$. To these clauses add literal w not found in other clauses, giving clauses C'.

If clauses C are satisfiable, clauses C' are satisfiable with not all literals equal. To show this, let w be False.

If the clauses C' are satisfiable, they are satisfiable when variables are complemented. Thus, for some assignment w is False. This assignment satisfies C. Thus, C is satisfiable if and only if C' is satisfiable.

0-1 INTEGER PROGRAMMING is **NP**-complete

0-1 INTEGER PROGRAMMING
Instance: A set of linear inequalities over Boolean variables $x_1, x_2, \ldots x_n$ with rational coefficients.
Answer: "Yes" if there is an assignment of values in $\{0, 1\}$ that satisifies the inequalities.

Theorem

0-1 INTEGER PROGRAMMING *is* **NP**-*complete.*

Proof.

Clearly 0-1 INTEGER PROGRAMMING is in **NP**. We reduce 3-SAT to it. For each i add inequalities $0 \leq x_i \leq 1$. This ensures the integer values are 0 or 1. Second, express each clause as an inequality. E.g., express $(x_1 \vee \bar{x_2} \vee x_3)$ as $x_1 + (1 - x_2) + x_3 \geq 1$. □

INDEPENDENT SET is **NP**-complete

INDEPENDENT SET
Instance: A graph $G = (V, E)$ and an integer k.
Answer: "Yes" if there is a set of k vertices of G such that there is no edge in E between them.

Theorem

INDEPENDENT SET *is* **NP***-complete*

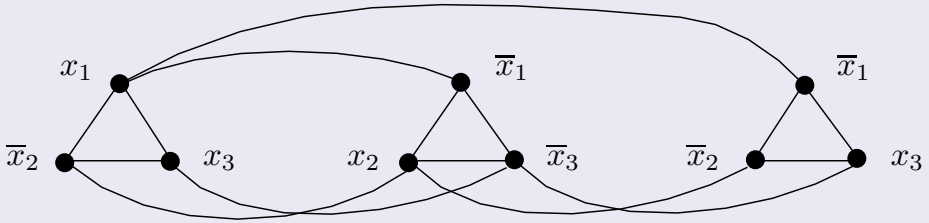
Proof

It is clearly in **NP**. We show it is **NP**-hard by reducing 3-SAT to it. Assume without loss of generality that each clause has exactly three literals. (Continued)

INDEPENDENT SET is **NP**-complete

Proof (cont.)

Construct instance $G = (V, E)$ of INDEPENDENT SET from instance of 3-SAT with k clauses. (See Fig. 10.) G has one triangle for each clause and vertices have names of literals in a clause. G has an edge between variable labels and their complements.



For a "Yes" instance of 3-SAT, pick one True literal from each clause, identifying k vertices, one per triangle without edges between them. This is a "Yes" instance of INDEPENDENT SET. Conversely, a "Yes" instance of INDEPENDENT SET on G has k vertices, one per triangle. No two vertices have label of variable and complement. The literals associated with these independent vertices are assigned value 1, causing each clause to be satisfied.

CLIQUE is **NP**-complete

CLIQUE
Instance: An undirected graph $G = (V, E)$ and an integer k.
Answer: "Yes" if there is a set of k pairwise adjacent vertices.

Theorem

CLIQUE *is* **NP***-complete.*

Proof.

Clearly CLIQUE is in **NP**. To show that it is **NP**-hard, reduce INDEPENDENT SET to it. It is easy to show that a graph $G = (V, E)$ has an independent set of size k if and only if its complementary graph $\bar{G} = (V, \bar{E})$, where $\bar{E} = V \times V - E$, has a k-clique. □