Completeness[a]

- As reducibility is transitive, problems can be ordered with respect to their difficulty.
- Is there a *maximal* element?
- It is not altogether obvious that there should be a maximal element.
- Many infinite structures (such as integers and reals) do not have maximal elements.
- Hence it may surprise you that most of the complexity classes that we have seen so far have maximal elements.

[a]Cook (1971).

Completeness (concluded)

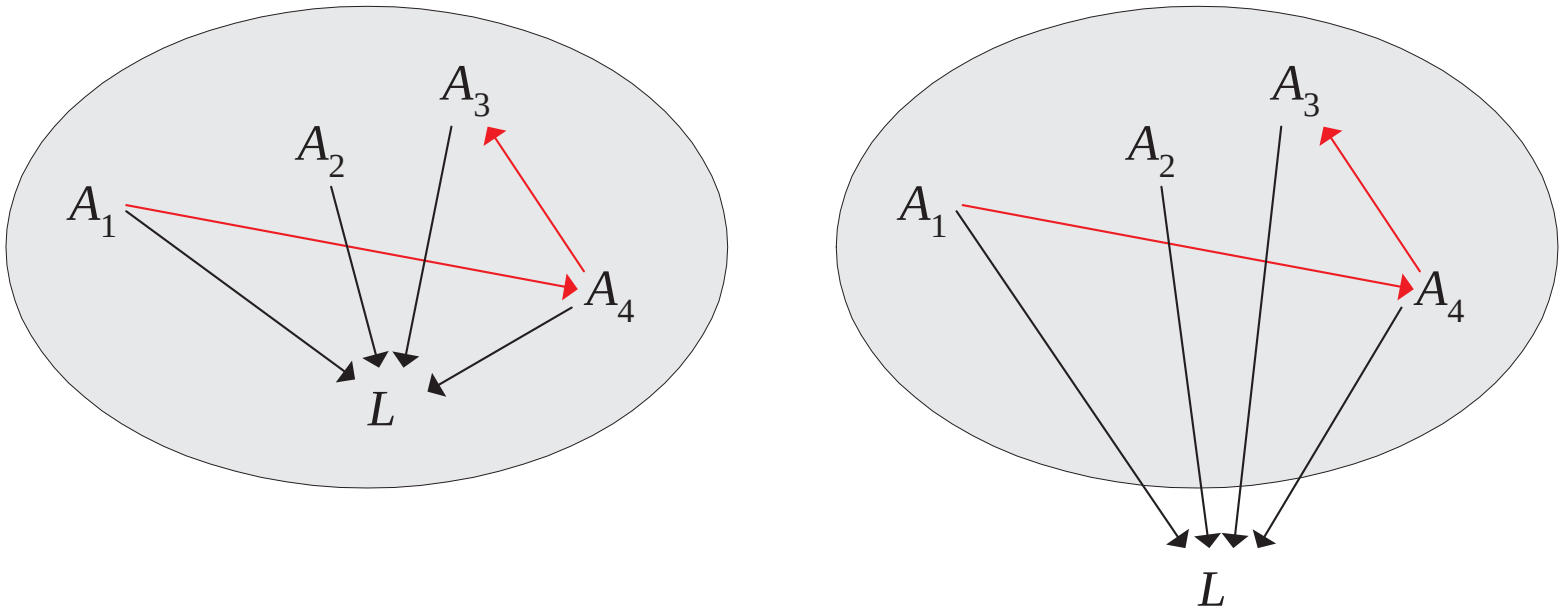
- Let $\mathcal{C}$ be a complexity class and $L \in \mathcal{C}$.
- L is **$\mathcal{C}$-complete** if every $L' \in \mathcal{C}$ can be reduced to L.
  - Most complexity classes we have seen so far have complete problems!
- Complete problems capture the difficulty of a class because they are the hardest.

Hardness

- Let $\mathcal{C}$ be a complexity class.
- L is **$\mathcal{C}$-hard** if every $L' \in \mathcal{C}$ can be reduced to L.
- It is not required that $L \in \mathcal{C}$.
- If L is $\mathcal{C}$-hard, then by definition, every $\mathcal{C}$-complete problem can be reduced to L.[a]

[a] Contributed by Mr. Ming-Feng Tsai (D92922003) on October 15, 2003.

Illustration of Completeness and Hardness



Closedness under Reduction

- A class $\mathcal{C}$ is **closed under reductions** if whenever L is reducible to L' and $L' \in \mathcal{C}$, then $L \in \mathcal{C}$.
- P, NP, and EXP are all closed under reductions.

Complete Problems and Complexity Classes

Proposition 10 *Let $\mathcal{C}'$ and $\mathcal{C}$ be two complexity classes such that $\mathcal{C}' \subseteq \mathcal{C}$. Assume $\mathcal{C}'$ is closed under reductions and L is a complete problem for $\mathcal{C}$. Then $\mathcal{C} = \mathcal{C}'$ if and only if $L \in \mathcal{C}'$.*

- Suppose $L \in \mathcal{C}'$ first.
- Every language $A \in \mathcal{C}$ reduces to $L \in \mathcal{C}'$.
- Because $\mathcal{C}'$ is closed under reductions, $A \in \mathcal{C}'$.
- Hence $\mathcal{C} \subseteq \mathcal{C}'$.
- As $\mathcal{C}' \subseteq \mathcal{C}$, we conclude that $\mathcal{C} = \mathcal{C}'$.

The Proof (concluded)

- On the other hand, suppose $\mathcal{C} = \mathcal{C}'$.
- As L is $\mathcal{C}$-complete, $L \in \mathcal{C}$.
- Thus, trivially, $L \in \mathcal{C}'$.

Two Immediate Corollaries

Proposition 10 implies that

- P = NP if and only if an NP-complete problem in P.
- L = P if and only if a P-complete problem is in L.

Table of Computation

- Let $M = (K, \Sigma, \delta, s)$ be a single-string polynomial-time deterministic TM deciding L.
- Its computation on input x can be thought of as a $|\,x\,|^k \times |\,x\,|^k$ table, where $|\,x\,|^k$ is the time bound (recall that it is an upper bound).
  - It is a sequence of configurations.
- Rows correspond to time steps 0 to $|\,x\,|^k - 1$.
- Columns are positions in the string of M.
- The (i, j)th table entry represents the contents of position j of the string *after* i steps of computation.

Some Conventions To Simplify the Table

- M halts after at most $|\,x\,|^k - 2$ steps.
  - The string length hence never exceeds $|\,x\,|^k$.
- Assume a large enough k to make it true for $|\,x\,| \geq 2$.
- Pad the table with $\sqcup$s so that each row has length $|\,x\,|^k$.
  - The computation will never reach the right end of the table for lack of time.
- If the cursor scans the jth position at time i when M is at state q and the symbol is σ, then the (i, j)th entry is a *new* symbol σ_q.

Some Conventions To Simplify the Table (continued)

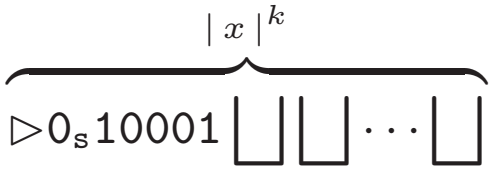
- If q is "yes" or "no," simply use "yes" or "no" instead of σ_q.
- Modify M so that the cursor starts not at $\triangleright$ but at the first symbol of the input.
- The cursor never visits the leftmost $\triangleright$ by telescoping two moves of M each time the cursor is about to move to the leftmost $\triangleright$.
- So the first symbol in every row is a $\triangleright$ and not a $\triangleright_q$.

Some Conventions To Simplify the Table (concluded)

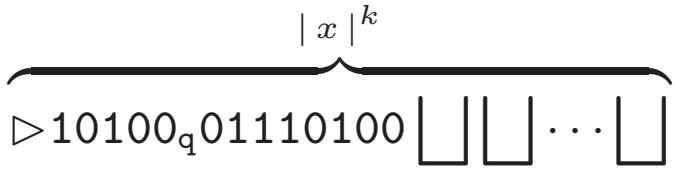
- If M has halted before its time bound of $|\,x\,|^k$, so that "yes" or "no" appears at a row before the last, then all subsequent rows will be identical to that row.
- M accepts x if and only if the $(|\,x\,|^k - 1, j)$th entry is "yes" for some j.

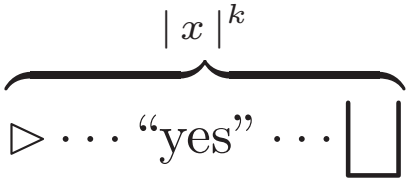
Comments

- Each row is essentially a configuration.
- If the input $x = 010001$, then the first row is



- A typical row may be



- The last rows must look like 

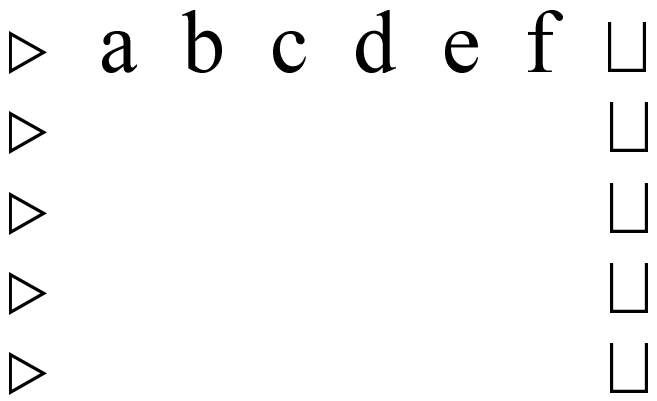
A P-Complete Problem

Theorem 11 (Ladner (1975)) CIRCUIT VALUE *is P-complete.*

- It is easy to see that CIRCUIT VALUE $\in$ P.
- For any $L \in$ P, we will construct a reduction R from L to CIRCUIT VALUE.
- Given any input x, $R(x)$ is a variable-free circuit such that $x \in L$ if and only if $R(x)$ evaluates to true.
- Let M decide L in time n^k.
- Let T be the computation table of M on x.

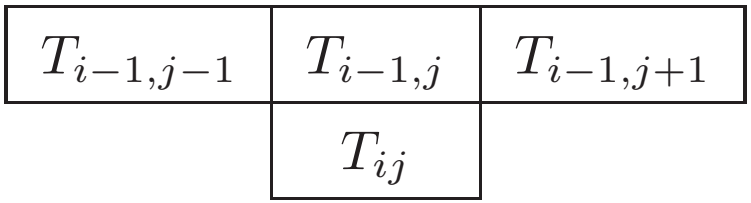
The Proof (continued)

- When $i = 0$, or $j = 0$, or $j = |\,x\,|^k - 1$, then the value of T_{ij} is known.
  - The jth symbol of x or $\sqcup$, a $\triangleright$, and a $\sqcup$, respectively.
  - Three out of four of T's borders are known.



The Proof (continued)

- Consider *other* entries T_{ij}.
- T_{ij} depends on only $T_{i-1,j-1}$, $T_{i-1,j}$, and $T_{i-1,j+1}$.



- Let Γ denote the set of all symbols that can appear on the table: $\Gamma = \Sigma \cup \{\sigma_q : \sigma \in \Sigma, q \in K\}$.
- Encode each symbol of Γ as an m-bit number, where

$$m = \lceil \log_2 |\, \Gamma \,| \rceil$$

(**state assignment** in circuit design).

The Proof (continued)

- Let binary string $S_{ij1}S_{ij2}\cdots S_{ijm}$ encode T_{ij}.
- We may treat them interchangeably without ambiguity.
- The computation table is now a table of binary entries $S_{ij\ell}$, where

$$0 \le i \le n^k - 1,$$
$$0 \le j \le n^k - 1,$$
$$1 \le \ell \le m.$$

The Proof (continued)

- Each bit $S_{ij\ell}$ depends on only $3m$ other bits:

$$\begin{array}{lllll} T_{i-1,j-1}: & S_{i-1,j-1,1} & S_{i-1,j-1,2} & \cdots & S_{i-1,j-1,m} \\ T_{i-1,j}: & S_{i-1,j,1} & S_{i-1,j,2} & \cdots & S_{i-1,j,m} \\ T_{i-1,j+1}: & S_{i-1,j+1,1} & S_{i-1,j+1,2} & \cdots & S_{i-1,j+1,m} \end{array}$$

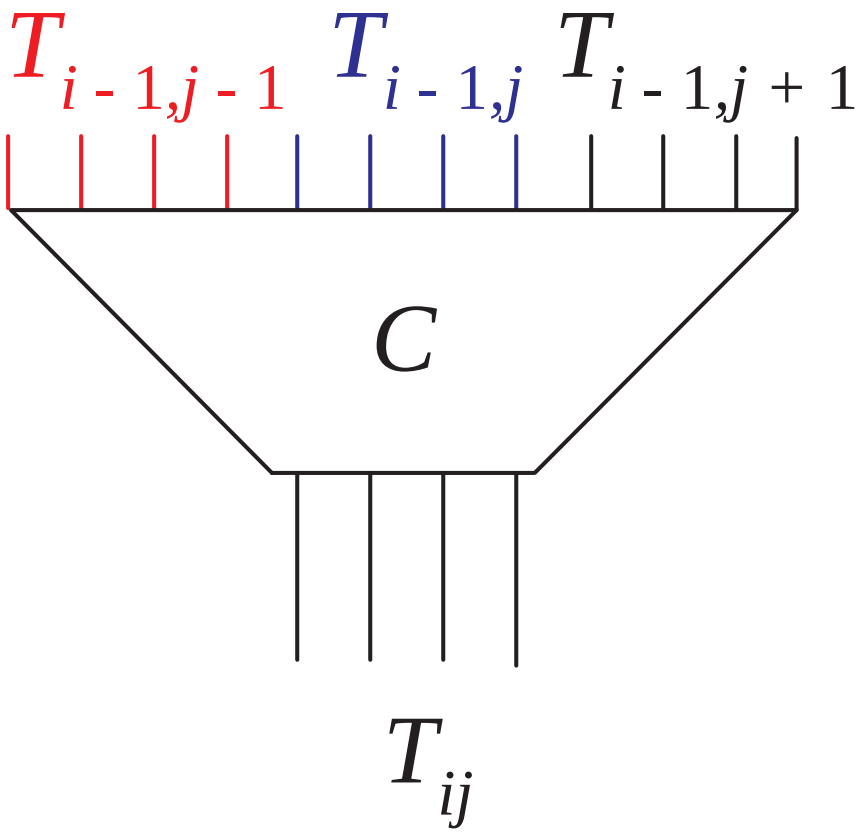
- So there are m boolean functions $F_1, F_2, \ldots, F_m$ with $3m$ inputs each such that for all $i, j > 0$,

$$\begin{aligned} S_{ij\ell} = F_\ell(&S_{i-1,j-1,1}, S_{i-1,j-1,2}, \ldots, S_{i-1,j-1,m}, \\ &S_{i-1,j,1}, S_{i-1,j,2}, \ldots, S_{i-1,j,m}, \\ &S_{i-1,j+1,1}, S_{i-1,j+1,2}, \ldots, S_{i-1,j+1,m}). \end{aligned}$$

The Proof (continued)

- These F_i's depend on only M's specification, not on x.
- Their sizes are fixed.
- These boolean functions can be turned into boolean circuits.
- Compose these m circuits in parallel to obtain circuit C with $3m$-bit inputs and m-bit outputs.
  - Schematically, $C(T_{i-1,j-1}, T_{i-1,j}, T_{i-1,j+1}) = T_{ij}$.
  - C is like an ASIC (application-specific IC) chip.

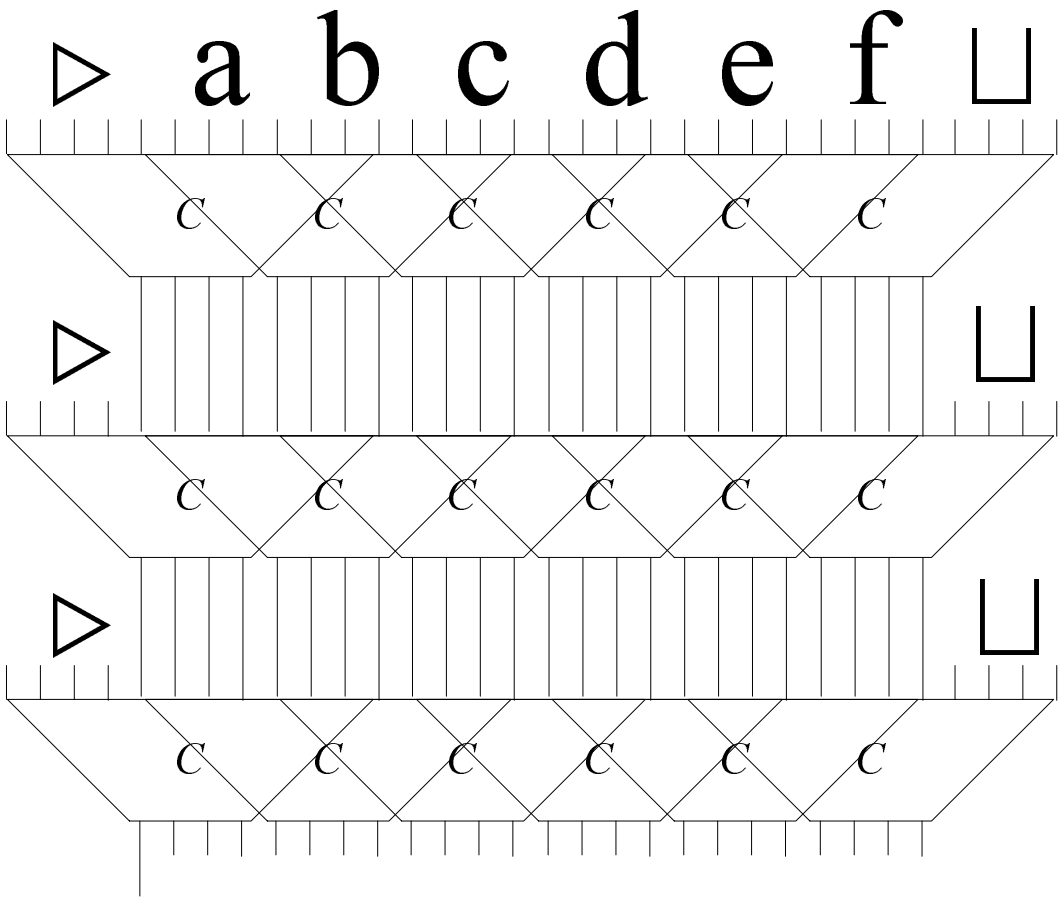
Circuit C



The Proof (concluded)

- A copy of circuit C is placed at each entry of the table.
  - Exceptions are the top row and the two extreme columns.
- $R(x)$ consists of $(|\,x\,|^k - 1)(|\,x\,|^k - 2)$ copies of circuit C.
- Without loss of generality, assume the output "yes"/"no" (coded as 1/0) appear at position $(|\,x\,|^k - 1, 1)$.

The Computation Tableau and $R(x)$



Cook's Theorem: the First NP-Complete Problem

Theorem 12 (Cook (1971)) SAT *is NP-complete.*

- SAT $\in$ NP (p. 51).
- CIRCUIT SAT reduces to SAT (p. 116).
- Now we only need to show that all languages in NP can be reduced to CIRCUIT SAT.

The Proof (continued)

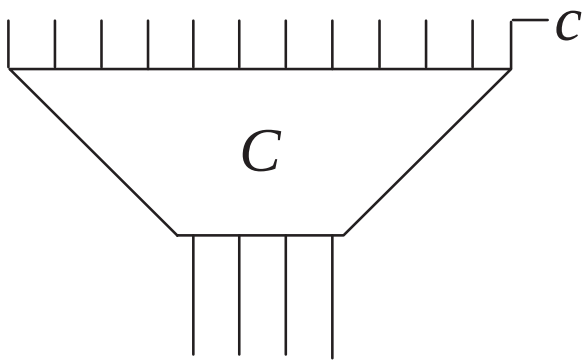
- Let single-string NTM M decide $L \in \text{NP}$ in time n^k.
- Assume M has exactly *two* nondeterministic choices at each step: choices 0 and 1.
- For each input x, we construct circuit $R(x)$ such that $x \in L$ if and only if $R(x)$ is satisfiable.
- A sequence of nondeterministic choices is a bit string

$$B = (c_1, c_2, \ldots, c_{|x|^k - 1}) \in \{0,1\}^{|x|^k - 1}.$$

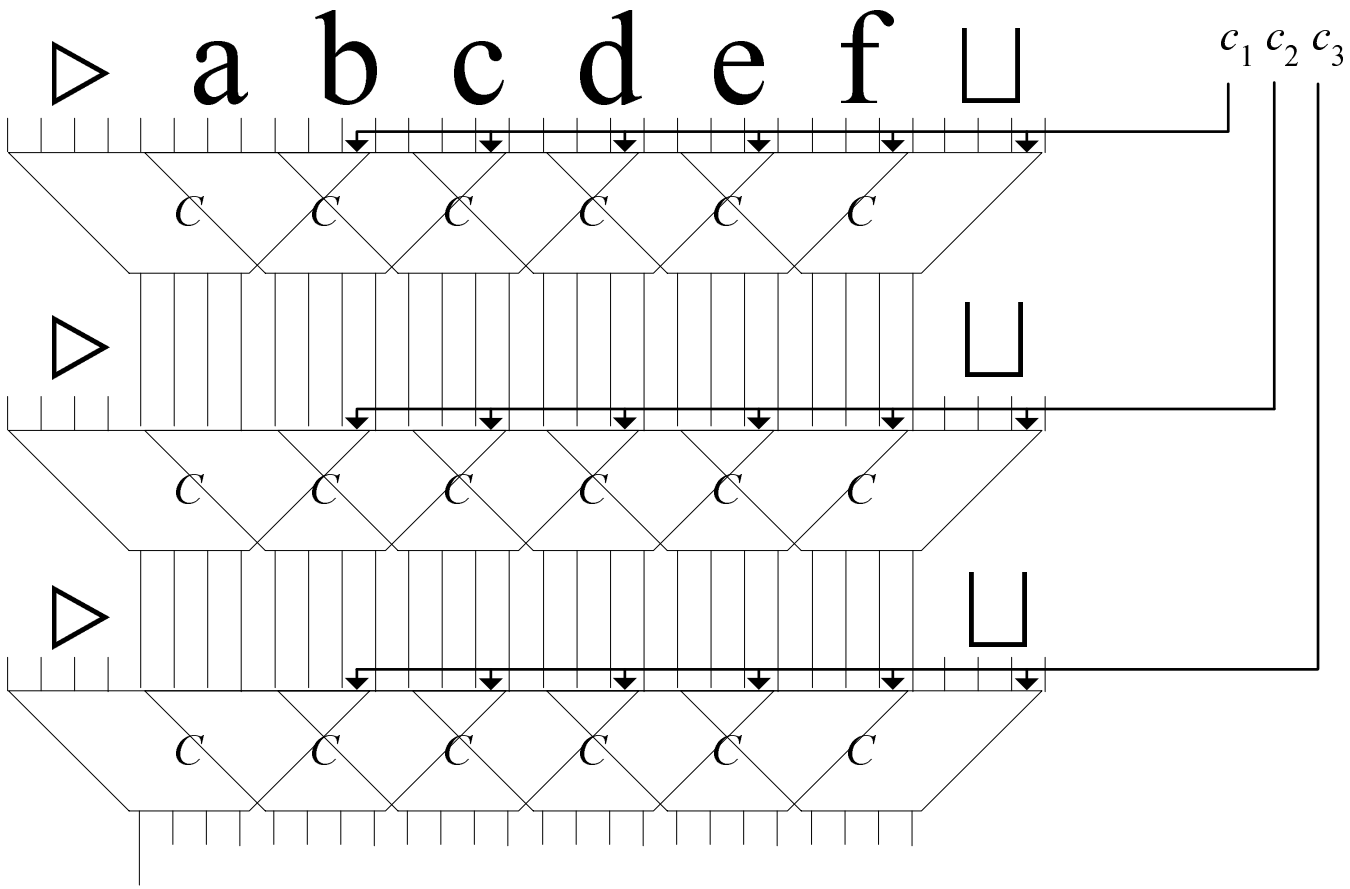
- Once B is fixed, the computation is *deterministic*.

The Proof (continued)

- Each choice of B results in a deterministic polynomial-time computation, hence a table like the one on p. 141.
- Each circuit C at time i has an extra binary input c corresponding to the nondeterministic choice: $C(T_{i-1,j-1}, T_{i-1,j}, T_{i-1,j+1}, c) = T_{ij}$.



The Computation Tableau for NTMs and $R(x)$



The Proof (concluded)

- The overall circuit $R(x)$ (on p. 145) is satisfiable if there is a truth assignment B such that the computation table accepts.
- This happens if and only if M accepts x, i.e., $x \in L$.

NP-Complete Problems

Wir müssen wissen, wir werden wissen.
(We must know, we shall know.)
— David Hilbert (1900)

3SAT

- k-SAT, where $k \in \mathbb{Z}^+$, is the special case of SAT.
- The formula is in CNF and all clauses have *exactly* k literals (repetition of literals is allowed).
- For example,

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_1 \vee \neg x_2) \wedge (x_1 \vee \neg x_2 \vee \neg x_3).$$

3SAT Is NP-Complete

- Recall Cook's Theorem (p. 142) and the reduction of CIRCUIT SAT to SAT (p. 116).
- The resulting CNF has at most 3 literals for each clause.
  - This shows that 3SAT where each clause has at most 3 literals is NP-complete.
- Finally, duplicate one literal once or twice to make it a 3SAT formula.

Another Variant of 3SAT

Proposition 13 3SAT *is NP-complete for expressions in which each variable is restricted to appear at most three times, and each literal at most twice. (*3SAT *here requires only that each clause has* at most *3 literals.)*

- Consider a general 3SAT expression in which x appears k times.
- Replace the first occurrence of x by x_1, the second by x_2, and so on, where $x_1, x_2, \ldots, x_k$ are k *new* variables.

The Proof (concluded)

- Add $(\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_3) \wedge \cdots \wedge (\neg x_k \vee x_1)$ to the expression.
  - This is logically equivalent to
    $x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_k \Rightarrow x_1$.
  - Note that each clause above has fewer than 3 literals.
- The resulting equivalent expression satisfies the condition for x.

An Example

- Suppose we are given the following 3SAT expression

$$\cdots (\neg x \vee w \vee g) \wedge \cdots \wedge (x \vee y \vee z) \cdots .$$

- The transformed expression is

$$\cdots (\neg x_1 \vee w \vee g) \wedge \cdots \wedge (x_2 \vee y \vee z) \cdots (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee x_1).$$

  - Variable x_1 appears thrice.
  - Literal x_1 appears once.
  - Literal $\neg x_1$ appears twice.

NAESAT

- The NAESAT (for "not-all-equal" SAT) is like 3SAT.
- But we require additionally that there be a satisfying truth assignment under which no clauses have the three literals equal in truth value.
  - Each clause must have one literal assigned true and one literal assigned false.

NAESAT Is NP-Complete[a]

- Recall the reduction of CIRCUIT SAT to SAT on p. 116.
- It produced a CNF ϕ in which each clause has at most 3 literals.
- Add the same variable z to all clauses with fewer than 3 literals to make it a 3SAT formula.
- Goal: The new formula $\phi(z)$ is NAE-satisfiable if and only if the original circuit is satisfiable.

[a]Karp (1972).

The Proof (continued)

- Suppose T NAE-satisfies $\phi(z)$.
  - $\bar{T}$ also NAE-satisfies $\phi(z)$.
  - Under T or $\bar{T}$, variable z takes the value false.
  - This truth assignment must still satisfy all clauses of ϕ.
  - So it satisfies the original circuit.

The Proof (concluded)

- Suppose there is a truth assignment that satisfies the circuit.
  - Then there is a truth assignment T that satisfies every clause of ϕ.
  - Extend T by adding $T(z) = \texttt{false}$ to obtain T'.
  - T' satisfies $\phi(z)$.
  - So in no clauses are all three literals false under T'.
  - Under T', in no clauses are all three literals true.
    - Review the detailed construction on p. 117 and p. 118.

Undirected Graphs

- An **undirected graph** $G = (V, E)$ has a finite set of nodes, V, and a set of *undirected* edges, E.
- It is like a directed graph except that the edges have no directions and there are no self-loops.
- We use $[\, i, j \,]$ to denote the fact that there is an edge between node i and node j.

Independent Sets

- Let $G = (V, E)$ be an undirected graph.
- $I \subseteq V$.
- I is **independent** if whenever $i, j \in I$, there is no edge between i and j.
- The INDEPENDENT SET problem: Given an undirected graph and a goal K, is there an independent set of size K?
  - Many applications.

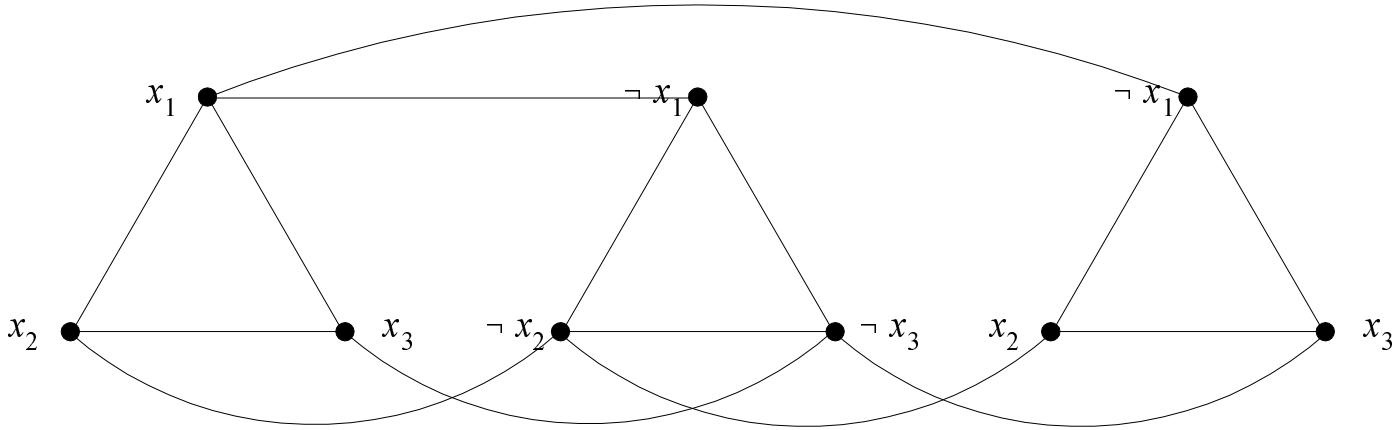
INDEPENDENT SET Is NP-Complete

- This problem is in NP: Guess a set of nodes and verify that it is independent and meets the count.
- If a graph contains a triangle, any independent set can contain at most one node of the triangle.
- We consider graphs whose nodes can be partitioned in m disjoint triangles.
  - If the special case is hard, the original problem must be at least as hard.

Reduction from 3SAT to INDEPENDENT SET

- Let ϕ be an instance of 3SAT with m clauses.
- We will construct graph G (with constraints as said) with $K = m$ such that ϕ is satisfiable if and only if G has an independent set of size K.
- There is a triangle for each clause with the literals as the nodes.
- Add additional edges between x and $\neg x$ for every variable x.

A Sample Construction



$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3).$$

The Proof (continued)

- Suppose G has an independent set I of size $K = m$.
  - An independent set can contain at most m nodes, one from each triangle.
  - An independent set of size m exists if and only if it contains exactly one node from each triangle.
  - Truth assignment T assigns true to those literals in I.
  - T is consistent because contradictory literals are connected by an edge, hence not both in I.
  - T satisfies ϕ because it has a node from every triangle, thus satisfying every clause.

The Proof (concluded)

- Suppose a satisfying truth assignment T exists for ϕ.
  - Collect one node from each triangle whose literal is true under T.
  - The choice is arbitrary if there is more than one true literal.
  - This set of m nodes must be independent by construction.
    - Literals x and $\neg x$ cannot be both assigned true.

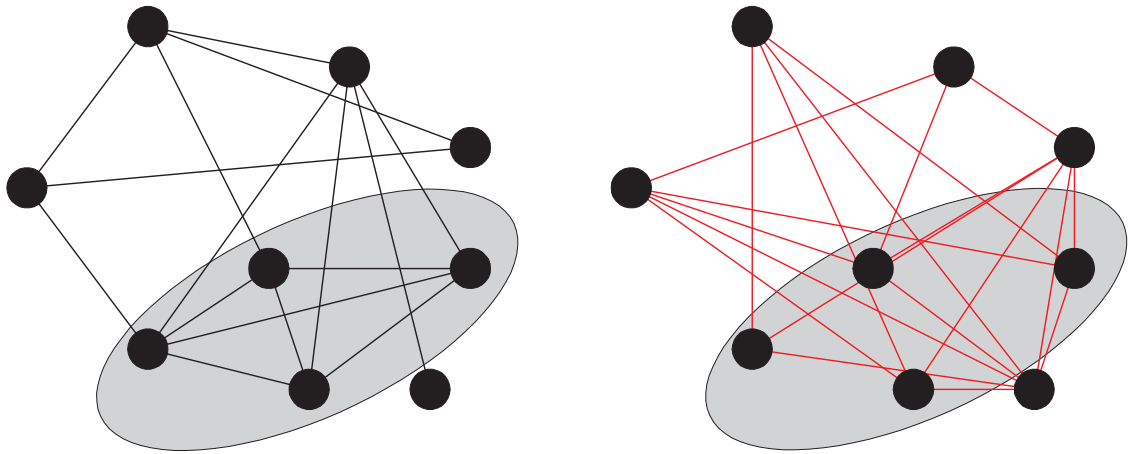
CLIQUE

- We are given an undirected graph G and a goal K.
- CLIQUE asks if there is a set C with K nodes such that whenever $i, j \in C$, there is an edge between i and j.

CLIQUE Is NP-Complete

Corollary 14 CLIQUE *is NP-complete.*

- Let $\bar{G}$ be the **complement** of G, where $[x, y] \in \bar{G}$ if and only if $[x, y] \notin G$.
- I is an independent set in $G \Leftrightarrow I$ is a clique in $\bar{G}$.



NODE COVER

- We are given an undirected graph G and a goal K.
- NODE COVER asks if there is a set C with K or fewer nodes such that each edge of G has at least one of its endpoints in C.

NODE COVER Is NP-Complete

Corollary 15 NODE COVER *is NP-complete.*

- I is an independent set of $G = (V, E)$ if and only if $V - I$ is a node cover of G.