Completeness[a]

- As reducibility is transitive, problems can be ordered with respect to their difficulty.
- Is there a *maximal* element (the so-called *hardest* problem)?
- It is not obvious that there should be a maximal element.
  - Many infinite structures (such as integers and real numbers) do not have maximal elements.
- Surprisingly, most of the complexity classes that we have seen so far have maximal elements!

[a] Post (1944); Cook (1971); Levin (1973).

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 of the complexity classes we have seen so far have complete problems!
- Complete problems capture the difficulty of a class because they are the hardest problems in the class.[a]

---

[a] See also p. 164.

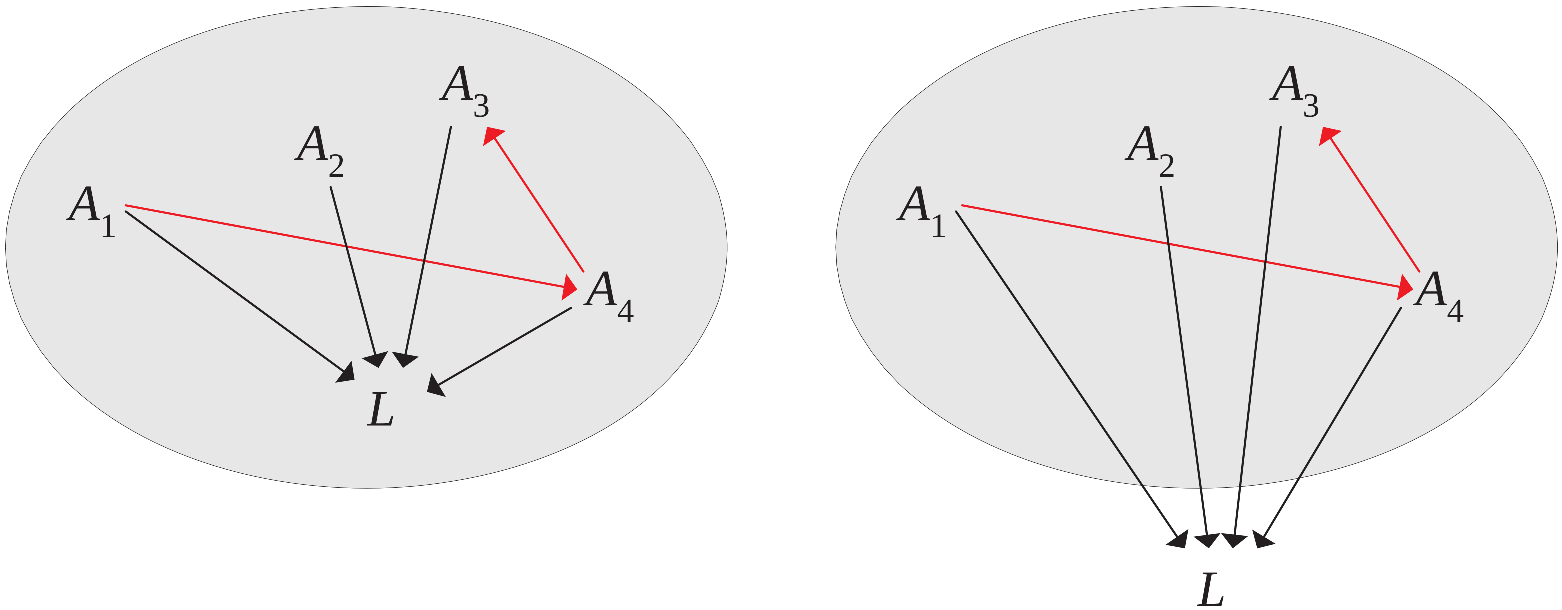
Hardness

- 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 Reductions

- A class $\mathcal{C}$ is **closed under reductions** if whenever L is reducible to L' and $L' \in \mathcal{C}$, then $L \in \mathcal{C}$.
- It is easy to show that P, NP, coNP, L, NL, PSPACE, and EXP are all closed under reductions.
- E is not closed under reductions.[a]

[a] Balcázar, Díaz, & Gabarró (1988).

Complete Problems and Complexity Classes

Proposition 29 *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 $\mathcal{C}$-complete. 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 Important Corollaries

Proposition 29 implies the following.

Corollary 30 *$P = NP$ if and only if an NP-complete problem is in P.*

Corollary 31 *$L = P$ if and only if a P-complete problem is in L.*

Complete Problems and Complexity Classes, Again

Proposition 32 *Let $\mathcal{C}'$ and $\mathcal{C}$ be two complexity classes closed under reductions. If L is complete for both $\mathcal{C}$ and $\mathcal{C}'$, then $\mathcal{C} = \mathcal{C}'$.*

- All languages $A \in \mathcal{C}$ reduce to $L \in \mathcal{C}$ and $L \in \mathcal{C}'$.
- Since $\mathcal{C}'$ is closed under reductions, $A \in \mathcal{C}'$.
- Hence $\mathcal{C} \subseteq \mathcal{C}'$.
- The proof for $\mathcal{C}' \subseteq \mathcal{C}$ is symmetric.

Complete Problems and Complexity Classes, Again (concluded)

Proposition 33 *Let $\mathcal{C}$ be a complexity class. If L is $\mathcal{C}$-complete and L is reducible to $L' \in \mathcal{C}$, then L' is also $\mathcal{C}$-complete.*

- Every language $A \in \mathcal{C}$ reduces to L.
- By Proposition 28 (p. 291), A reduces to 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.
  - It is essentially 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.[a]
- 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.

[a] $|x|^k - 3$ may be safer.

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 $\rhd$ but at the first symbol of the input.
- The cursor never visits the leftmost $\rhd$ by telescoping two moves of M each time the cursor is about to move to the leftmost $\rhd$.
- So the first symbol in every row is a $\rhd$ and not a $\rhd_q$.

Some Conventions To Simplify the Table (concluded)

- M will halt before the last row is reached.
- All subsequent rows will be identical to the row where M halts.
- M accepts x if and only if the $(|\,x\,|^k - 1, j)$th entry is "yes" for some position j.

Comments

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

$$\overbrace{\rhd \mathtt{0}_{\mathtt{s}}\mathtt{10001} \sqcup \sqcup \cdots \sqcup}^{|\,x\,|^k}$$

- A typical row looks like

$$\overbrace{\rhd \mathtt{10100}_{\mathtt{q}}\mathtt{01110100} \sqcup \sqcup \cdots \sqcup}^{|\,x\,|^k}$$

Comments (concluded)

- The last rows must look like

$$\overbrace{\rhd \cdots \text{"yes"} \cdots \sqcup}^{|\,x\,|^k} \quad \text{or} \quad \overbrace{\rhd \cdots \text{"no"} \cdots \sqcup}^{|\,x\,|^k}$$

- Three out of the table's 4 borders are known:

$$\begin{array}{cccccccc}
\rhd & a & b & c & d & e & f & \sqcup \\
\rhd & & & & & & & \sqcup \\
\rhd & & & & & & & \sqcup \\
\rhd & & & & & & & \sqcup \\
\rhd & & & & & & & \sqcup \\
 & & & \vdots & & & &
\end{array}$$

A P-Complete Problem

Theorem 34 (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)

- Recall that three out of T's 4 borders are known.
- So when $i = 0$, or $j = 0$, or $j = |\, x \,|^k - 1$, the value of T_{ij} is known.
  - The jth symbol of x or $\sqcup$, a $\rhd$, or a $\sqcup$, respectively.
- Consider *other* entries T_{ij}.

The Proof (continued)

- T_{ij} depends on *only* $T_{i-1,j-1}$, $T_{i-1,j}$, and $T_{i-1,j+1}$:

$T_{i-1,j-1}$	$T_{i-1,j}$	$T_{i-1,j+1}$
	T_{ij}	

- T_{ij} does *not* depend on any other entries!
- T_{ij} does *not* depend on i, j, or x either (given $T_{i-1,j-1}$, $T_{i-1,j}$, and $T_{i-1,j+1}$).
- The dependency is thus "local."

The Proof (continued)

- 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,[a] where

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

---

[a] Called **state assignment** in circuit design.

The Proof (continued)

- Let the m-bit 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

$$
\begin{aligned}
0 &\leq i \leq n^k - 1, \\
0 &\leq j \leq n^k - 1, \\
1 &\leq \ell \leq m.
\end{aligned}
$$

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 truth values for the $3m$ bits determine $S_{ij\ell}$.

The Proof (continued)

- This means there is a boolean function F_ℓ with $3m$ inputs such that

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

for all $i, j > 0$ and $1 \leq \ell \leq m$.

The Proof (continued)

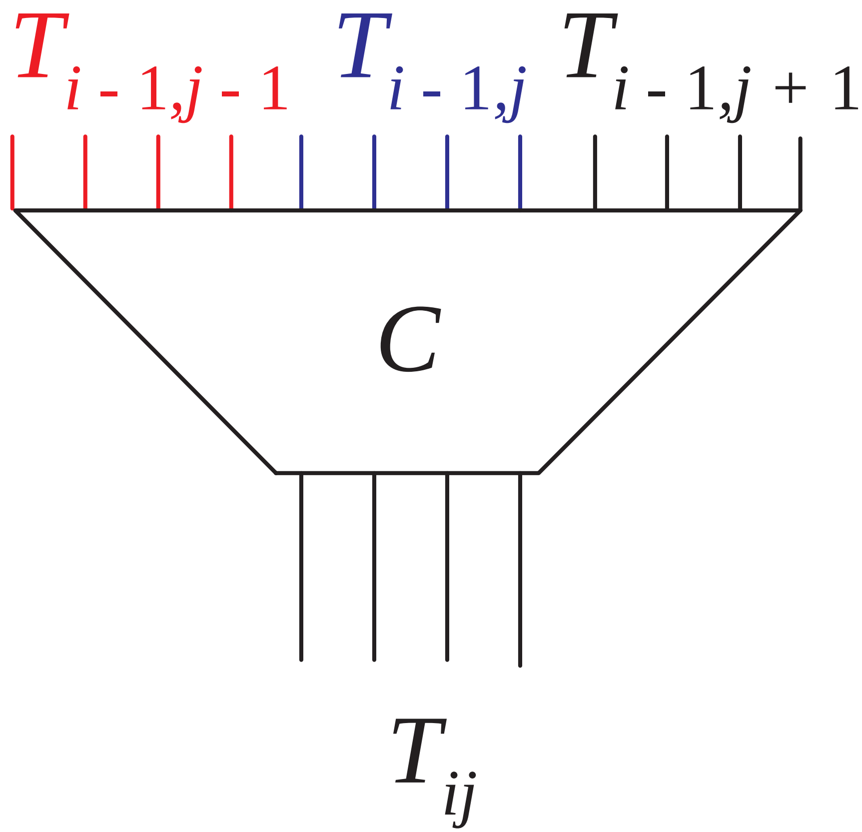
- These F_ℓ's depend only on M's specification, not on x, i, or j.
- Their sizes are constant.[a]
- These boolean functions can be turned into boolean circuits (see p. 209).
- Compose these m circuits in parallel to obtain circuit C with $3m$-bit inputs and m-bit outputs.
  - Schematically, $C(T_{i-1,j-1}, T_{i-1,j}, T_{i-1,j+1}) = T_{ij}$.[b]

---

[a] It means independence of the input x.

[b] C is like an ASIC (application-specific IC) chip.

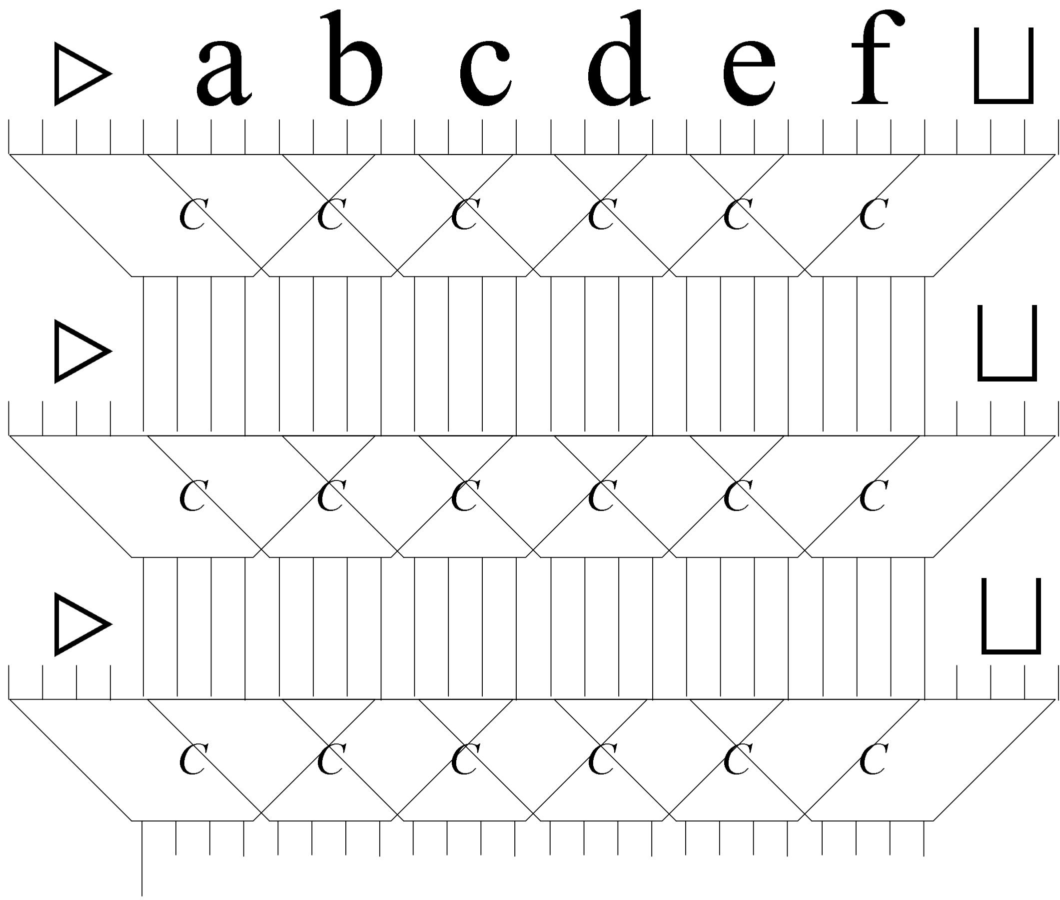
Circuit C



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 column borders.
- $R(x)$ consists of $(|\, x\,|^k - 1)(|\, x\,|^k - 2)$ copies of circuit C.
- Without loss of generality, assume the output "yes"/"no" appear at position $(|\, x\,|^k - 1, 1)$.
- Encode "yes" as 1 and "no" as 0.

The Computation Tableau and $R(x)$



A Corollary

The construction in the above proof yields the following, more general result.

Corollary 35 *If $L \in TIME(T(n))$, then a circuit with $O(T^2(n))$ gates can decide L.*

MONOTONE CIRCUIT VALUE

- A **monotone** boolean circuit's output cannot change from true to false when one input changes from false to true.
- Monotone boolean circuits are hence less expressive than general circuits.
  - They can compute only *monotone* boolean functions.
- Monotone circuits do not contain $\neg$ gates (prove it).
- MONOTONE CIRCUIT VALUE is CIRCUIT VALUE applied to monotone circuits.

MONOTONE CIRCUIT VALUE Is P-Complete

Despite their limitations, MONOTONE CIRCUIT VALUE is as hard as CIRCUIT VALUE.

Corollary 36 (Goldschlager, 1977) MONOTONE CIRCUIT VALUE *is P-complete.*

- Given any general circuit, "move the $\neg$'s downwards" using de Morgan's laws[a] to yield a monotone circuit with the same output.

Theorem 37 (Goldschlager, 1977) PLANAR MONOTONE CIRCUIT VALUE *is P-complete.*

---

[a] How? Need to make sure no exponential blowup.

MAXIMUM FLOW Is P-Complete

Theorem 38 (Goldschlager, Shaw, & Staples, 1982) MAXIMUM FLOW *is P-complete.*

Cook's Theorem: the First NP-Complete Problem

Theorem 39 (Cook, 1971) SAT *is NP-complete.*

- SAT $\in$ NP (p. 121).
- CIRCUIT SAT reduces to SAT (p. 285).
- By Proposition 33 (p. 301), it remains to show that all languages in NP can be reduced to CIRCUIT SAT.[a]

---

[a] As a bonus, this also shows CIRCUIT SAT is NP-complete.

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.
- Equivalently, for each input x, $M(x) =$ "yes" for some computation path if and only if $R(x)$ is satisfiable.
- How to come up with a polynomial-sized $R(x)$ when there are exponentially many computation paths?

The Proof (continued)

- A straightforward proof is to construct a variable-free circuit $R_i(x)$ for the ith computation path.[a]
- Then add a small circuit to output 1 if and only if there is an $R_i(x)$ that outputs a "yes."
- Clearly, the resulting circuit outputs 1 if and only if M accepts x.
- But, it is too large because there are exponentially many computation paths.
- Need to do better.

[a] The circuit for Theorem 34 (p. 308) will do.

The Proof (continued)

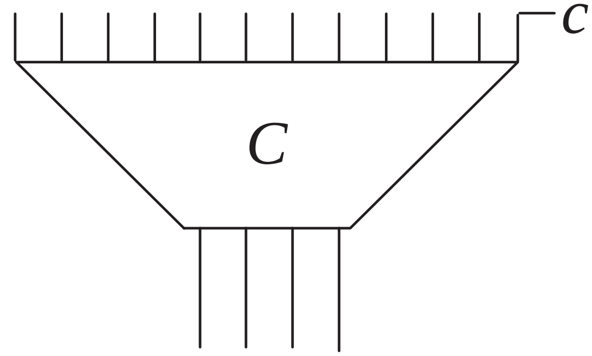
- A sequence of nondeterministic choices is a bit string

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

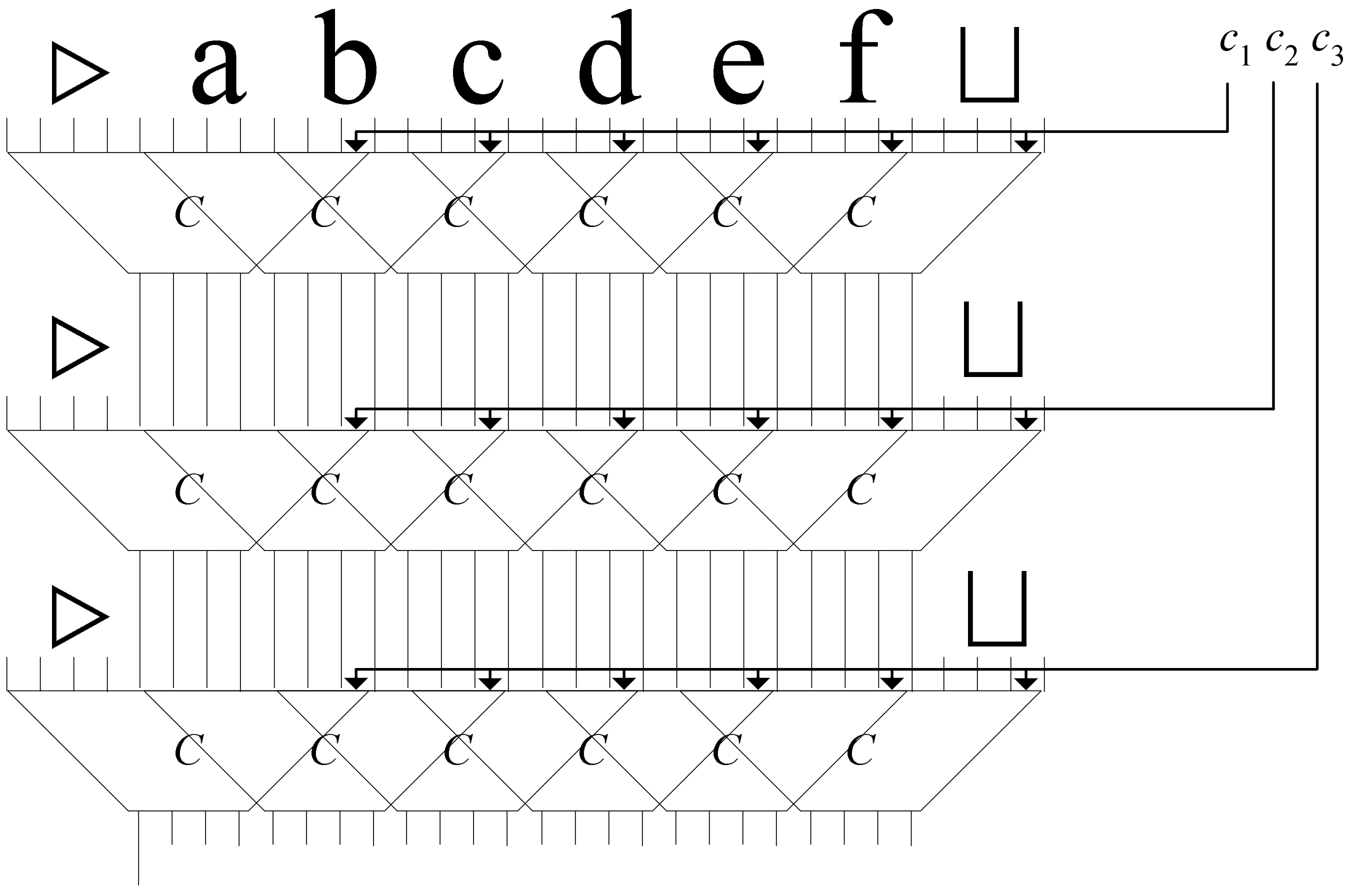
- Once B is given, the computation is *deterministic*.
- Each choice of B results in a deterministic polynomial-time computation.
- Each circuit C at time i has an *extra* binary input c corresponding to the nondeterministic choice:

$$C(T_{i-1,j-1}, T_{i-1,j}, T_{i-1,j+1}, c) = T_{ij}.$$

The Proof (continued)



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

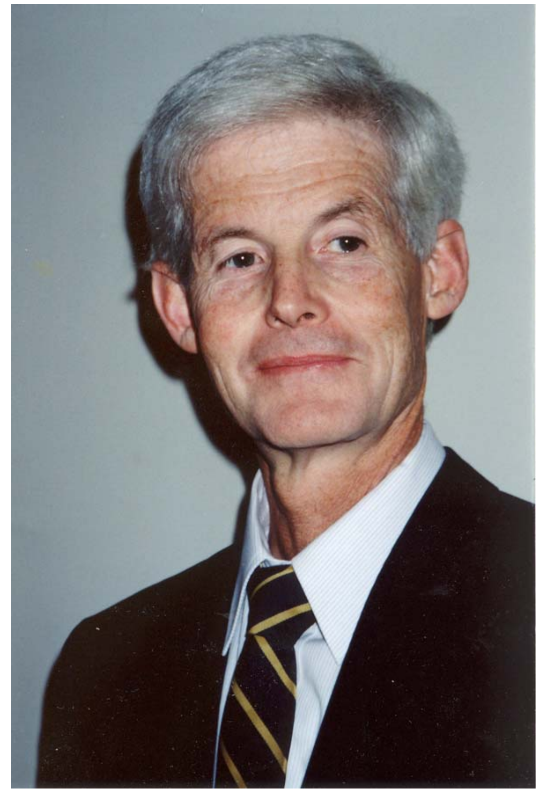


The Proof (concluded)

- Note that $c_1, c_2, \ldots, c_{|\,x\,|^k-1}$ constitute the variables of $R(x)$.
  - Some call them the choice gates of the circuit.
- The overall circuit $R(x)$ (on p. 328) is satisfiable if and only if there is a truth assignment B such that the computation table accepts.
- This happens if and only if M accepts x, i.e., $x \in L$.

Stephen Arthur Cook[a] (1939–)

Richard Karp, "It is to our everlasting shame that we were unable to persuade the math department [of UC-Berkeley] to give him tenure."



[a]Turing Award (1982). See http://conservancy.umn.edu/handle/107226 for an interview in 2002.

A Corollary

The construction in the above proof yields the following, more general result.

Corollary 40 *If $L \in NTIME(T(n))$, then a nondeterministic circuit with $O(T^2(n))$ gates can decide L.*

NP-Complete Problems

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

I predict that scientists will one day adopt a new principle: "NP-complete problems are hard." That is, solving those problems efficiently is impossible on any device that could be built in the real world, whatever the final laws of physics turn out to be.
— Scott Aaronson (2008)

Two Notions

- Let $R \subseteq \Sigma^* \times \Sigma^*$ be a binary relation on strings.
- R is called **polynomially decidable** if
$$\{\, x; y : (x, y) \in R \,\}$$
is in P.
- R is said to be **polynomially balanced** if $(x, y) \in R$ implies $|\, y \,| \leq |\, x \,|^k$ for some $k \geq 1$.

An Alternative Characterization of NP

Proposition 41 (Edmonds, 1965) *Let $L \subseteq \Sigma^*$ be a language. Then $L \in NP$ if and only if there is a polynomially decidable and polynomially balanced relation R such that*

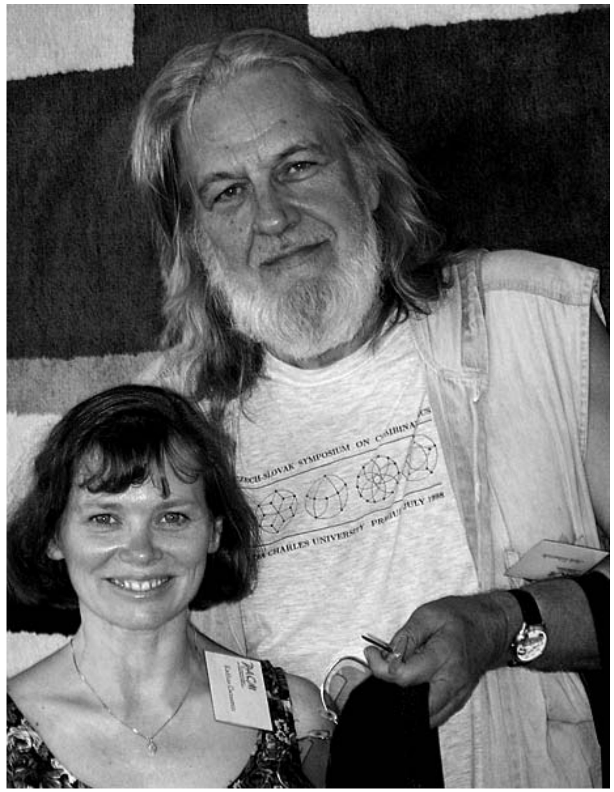
$$L = \{ x : \exists y \, (x, y) \in R \}.$$

- Suppose such an R exists.
- L can be decided by this NTM:
  - On input x, the NTM guesses a y of length $\leq |x|^k$.
  - It then tests if $(x, y) \in R$ in polynomial time.
  - It returns "yes" if the test is positive.

The Proof (concluded)

- Now suppose $L \in \text{NP}$.
- NTM N decides L in time $|\,x\,|^k$.
- Define R as follows: $(x, y) \in R$ if and only if y is the encoding of an accepting computation of N on input x.
- R is polynomially balanced as N runs in polynomial time.
- R is polynomially decidable because it can be efficiently verified by consulting N's transition function.
- Finally $L = \{\, x : (x, y) \in R \text{ for some } y \,\}$ because N decides L.

Jack Edmonds (1934–)



Comments

- Any "yes" instance x of an NP problem has at least one **succinct certificate** or **polynomial witness** y.
- "No" instances have none.
- Certificates are short and easy to verify.
  - An alleged satisfying truth assignment for SAT; an alleged Hamiltonian path for HAMILTONIAN PATH.
- Certificates may be hard to generate,[a] but verification must be easy.
- NP is thus the class of *easy-to-verify*[b] problems.

---

[a] Unless P equals NP.
[b] That is, in polynomial time.

Comments (concluded)

- The degree k is not an input.
- How to find the k needed by the NTM is of no concern.[a]
- We only need to prove there *exists* an NTM that accepts L in nondeterministic polynomial time.

---

[a] Contributed by Mr. Kai-Yuan Hou (`B99201038`, `R03922014`) on November 3, 2015.

You Have an NP-Complete Problem (for Your Thesis)

- From Propositions 29 (p. 297) and 32 (p. 300), it is the least likely to be in P.
- Your options are:
  - Approximations.
  - Special cases.
  - Average performance.
  - Randomized algorithms.
  - Exponential-time algorithms that work well in practice.
  - "Heuristics" (and pray that it works *for your thesis*).

I thought NP-completeness was an interesting idea:
I didn't quite realize its potential impact.
— Stephen Cook (1998)

I was indeed surprised by Karp's work
since I did not expect so many
wonderful problems were NP-complete.
— Leonid Levin (1998)

Correct Use of Reduction in Proving NP-Completeness

- Recall that L_1 reduces to L_2 if there is an efficient function R such that for all inputs x (p. 270),

$$x \in L_1 \text{ if and only if } R(x) \in L_2.$$

- When L_1 is known to be NP-complete and when $L_2 \in \text{NP}$, then L_2 is NP-complete.[a]
- A common mistake is to focus on solving $x \in L_1$ or solving $R(x) \in L_2$.
- The correct way is to, given a certificate for $x \in L_1$ (a satisfying truth assignment, e.g.), construct a certificate for $R(x) \in L_2$ (a Hamiltonian path, e.g.), and vice versa.

---

[a] Proposition 33 (p. 301).

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,

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

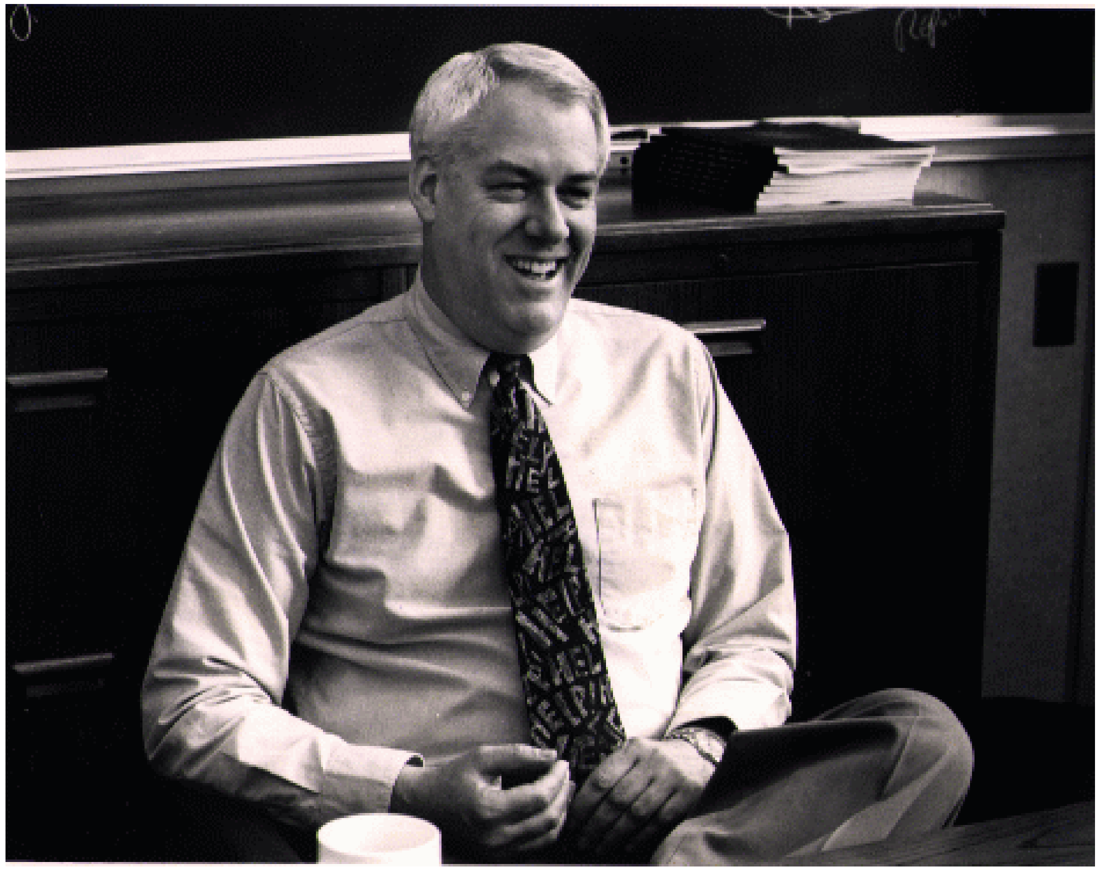
3SAT Is NP-Complete[a]

- Recall Cook's Theorem (p. 323) and the reduction of CIRCUIT SAT to SAT (p. 285).
- The resulting CNF has at most 3 literals for each clause.
  - This accidentally 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.
  - So

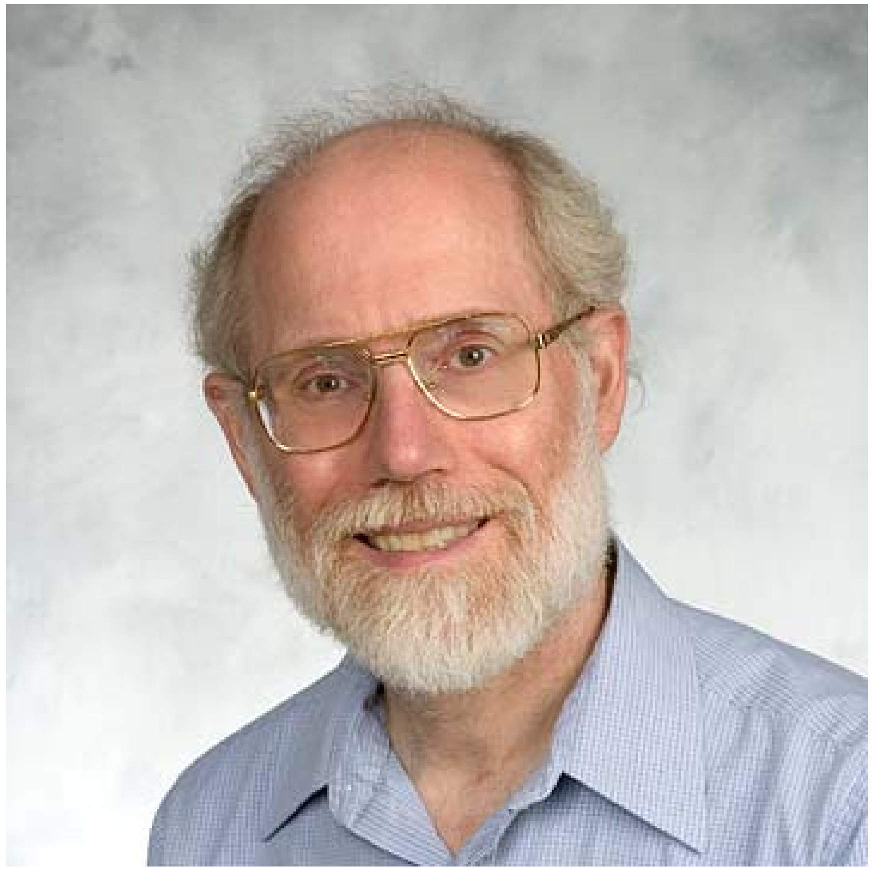
$$x_1 \vee x_2 \quad \text{becomes} \quad x_1 \vee x_2 \vee x_2.$$

[a]Garey, Johnson, & Stockmeyer (1976).

Michael R. Garey (1945–)



David S. Johnson (1945–)



Larry Stockmeyer (1948–2004)

The Satisfiability of Random 3SAT Expressions

- Consider a random 3SAT expressions ϕ with n variables and cn clauses.
- Each *clause* is chosen independently and uniformly from the set of all possible clauses.
- Intuitively, the larger the c, the less likely ϕ is satisfiable as more constraints are added.
- Indeed, there is a c_n such that for $c < c_n(1 - \epsilon)$, ϕ is satisfiable almost surely, and for $c > c_n(1 + \epsilon)$, ϕ is unsatisfiable almost surely.[a]

[a] Friedgut & Bourgain (1999). As of 2006, $3.52 < c_n < 4.596$.

Another Variant of 3SAT

Proposition 42 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.)*

The Proof (continued)

- 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.
  - $x_1, x_2, \ldots, x_k$ are k *new* variables.

The Proof (concluded)

- Add $(\neg x_1 \lor x_2) \land (\neg x_2 \lor x_3) \land \cdots \land (\neg x_k \lor x_1)$ to the expression.
  - It is logically equivalent to
$$x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_k \Rightarrow x_1.$$
  - So $x_1, x_2, \ldots, x_k$ must assume an identical truth value for the whole expression to be satisfied.
- Note that each clause $\neg x_i \lor x_j$ above has only 2 literals.
- The resulting equivalent expression satisfies the conditions for x.

An Example

- Suppose we are given the following 3SAT expression

$$\cdots (\neg x \lor w \lor g) \land \cdots \land (x \lor y \lor z) \cdots .$$

- The transformed expression is

$$\cdots (\neg \boxed{x_1} \lor w \lor g) \land \cdots \land (\boxed{x_2} \lor y \lor z) \cdots (\boxed{\neg x_1} \lor \boxed{x_2}) \land (\boxed{\neg x_2} \lor \boxed{x_1}).$$

  - Variable x_1 appears 3 times.
  - Literal x_1 appears once.
  - Literal $\neg x_1$ appears 2 times.

2SAT Is in NL $\subseteq$ P

- Let ϕ be an instance of 2SAT: Each clause has 2 literals.
- NL is a subset of P (p. 248).
- By Eq. (3) on p. 262, coNL equals NL.
- We need to show only that recognizing *unsatisfiable* 2SAT expressions is in NL.
- See the textbook for the complete proof.

Generalized 2SAT: MAX2SAT

- Consider a 2SAT formula.
- Let $K \in \mathbb{N}$.
- MAX2SAT asks whether there is a truth assignment that satisfies at least K of the clauses.
  - MAX2SAT becomes 2SAT when K equals the number of clauses.

Generalized 2SAT: MAX2SAT (concluded)

- MAX2SAT can be used to solve the related optimization problem.
  - With binary search, one can nail the maximum number of satisfiable clauses of 2SAT formulas.
- MAX2SAT $\in$ NP: Guess a truth assignment and verify the count.
- We now reduce 3SAT to MAX2SAT.

MAX2SAT Is NP-Complete[a]

- Consider the following 10 clauses:

$$(x) \wedge (y) \wedge (z) \wedge (w)$$
$$(\neg x \vee \neg y) \wedge (\neg y \vee \neg z) \wedge (\neg z \vee \neg x)$$
$$(x \vee \neg w) \wedge (y \vee \neg w) \wedge (z \vee \neg w)$$

- Let the 2SAT formula $r(x, y, z, w)$ represent the conjunction of these clauses.
- The clauses are symmetric with respect to x, y, and z.
- How many clauses can we satisfy?

[a]Garey, Johnson, & Stockmeyer (1976).

The Proof (continued)

All of x, y, z are true: By setting w to true, we satisfy $4 + 0 + 3 = 7$ clauses, whereas by setting w to false, we satisfy only $3 + 0 + 3 = 6$ clauses.

Two of x, y, z are true: By setting w to true, we satisfy $3 + 2 + 2 = 7$ clauses, whereas by setting w to false, we satisfy $2 + 2 + 3 = 7$ clauses.

The Proof (continued)

One of x, y, z is true: By setting w to false, we satisfy $1 + 3 + 3 = 7$ clauses, whereas by setting w to true, we satisfy only $2 + 3 + 1 = 6$ clauses.

None of x, y, z is true: By setting w to false, we satisfy $0 + 3 + 3 = 6$ clauses, whereas by setting w to true, we satisfy only $1 + 3 + 0 = 4$ clauses.

The Proof (continued)

- A truth assignment that satisfies $x \vee y \vee z$ can be *extended* to satisfy 7 of the 10 clauses of $r(x, y, z, w)$, *and no more*.
- A truth assignment that does *not* satisfy $x \vee y \vee z$ can be extended to satisfy only 6 of them, *and no more*.
- The reduction from 3SAT ϕ to 2SAT $R(\phi)$:
  - For each clause $C_i = (\alpha \vee \beta \vee \gamma)$ of ϕ, add $r(\alpha, \beta, \gamma, w_i)$ to $R(\phi)$.
- If ϕ has m clauses, then $R(\phi)$ has $10m$ clauses.

The Proof (continued)

- Finally, set $K = 7m$.
- So the reduction transforms ϕ to $(R(\phi), 7m)$.
- We now show that K clauses of $R(\phi)$ can be satisfied if and only if ϕ is satisfiable.

The Proof (continued)

- Suppose $K = 7m$ clauses of $R(\phi)$ can be satisfied.
  - 7 clauses of each $r(\alpha, \beta, \gamma, w_i)$ must be satisfied because it can have at most 7 clauses satisfied.[a]
  - Hence each clause $C_i = (\alpha \vee \beta \vee \gamma)$ of ϕ is satisfied by the same truth assignment.
  - So ϕ is satisfied.

[a]If 70% of the world population are male and if at most 70% of each country's population are male, then each country must have exactly 70% male population.

The Proof (concluded)

- Suppose ϕ is satisfiable.
  - Let T satisfy all clauses of ϕ.
  - Each $r(\alpha, \beta, \gamma, w_i)$ can set its w_i appropriately to have 7 clauses satisfied.
  - So $K = 7m$ clauses are satisfied.