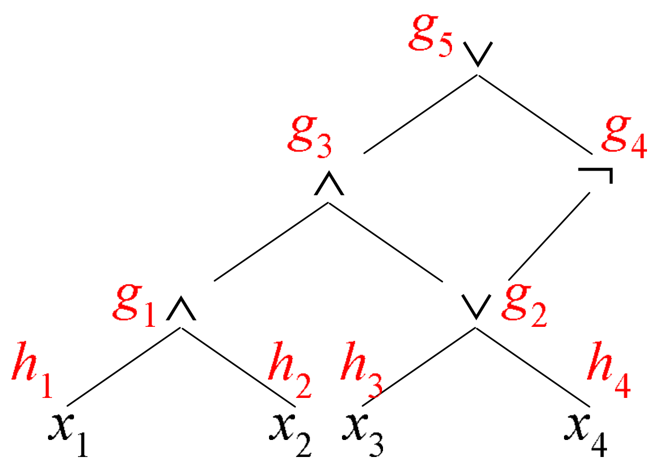
An Example



$$
\begin{aligned}
& (h_1 \Leftrightarrow x_1) \wedge (h_2 \Leftrightarrow x_2) \wedge (h_3 \Leftrightarrow x_3) \wedge (h_4 \Leftrightarrow x_4) \\
\wedge \quad & [\, g_1 \Leftrightarrow (h_1 \wedge h_2) \,] \wedge [\, g_2 \Leftrightarrow (h_3 \vee h_4) \,] \\
\wedge \quad & [\, g_3 \Leftrightarrow (g_1 \wedge g_2) \,] \wedge (g_4 \Leftrightarrow \neg g_2) \\
\wedge \quad & [\, g_5 \Leftrightarrow (g_3 \vee g_4) \,] \wedge g_5 .
\end{aligned}
$$

An Example (concluded)

- In general, the result is a CNF.
- The CNF has size proportional to the circuit's number of gates.
- The CNF adds new variables to the circuit's original input variables.
- Had we used the idea on p. 192 for the reduction, the resulting formula may have an exponential length because of the copying.[a]

[a]Contributed by Mr. Ching-Hua Yu (`D00921025`) on October 16, 2012.

Composition of Reductions

Proposition 26 *If R_{12} is a reduction from L_1 to L_2 and R_{23} is a reduction from L_2 to L_3, then the composition $R_{12} \circ R_{23}$ is a reduction from L_1 to L_3.*

- So reducibility is transitive.

Completeness[a]

- As reducibility is transitive, problems can be ordered with respect to their difficulty.
- Is there a *maximal* element (the *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.
- Hence it may surprise you that most of the complexity classes that we have seen so far have maximal elements.

[a]Cook (1971) and 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 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.

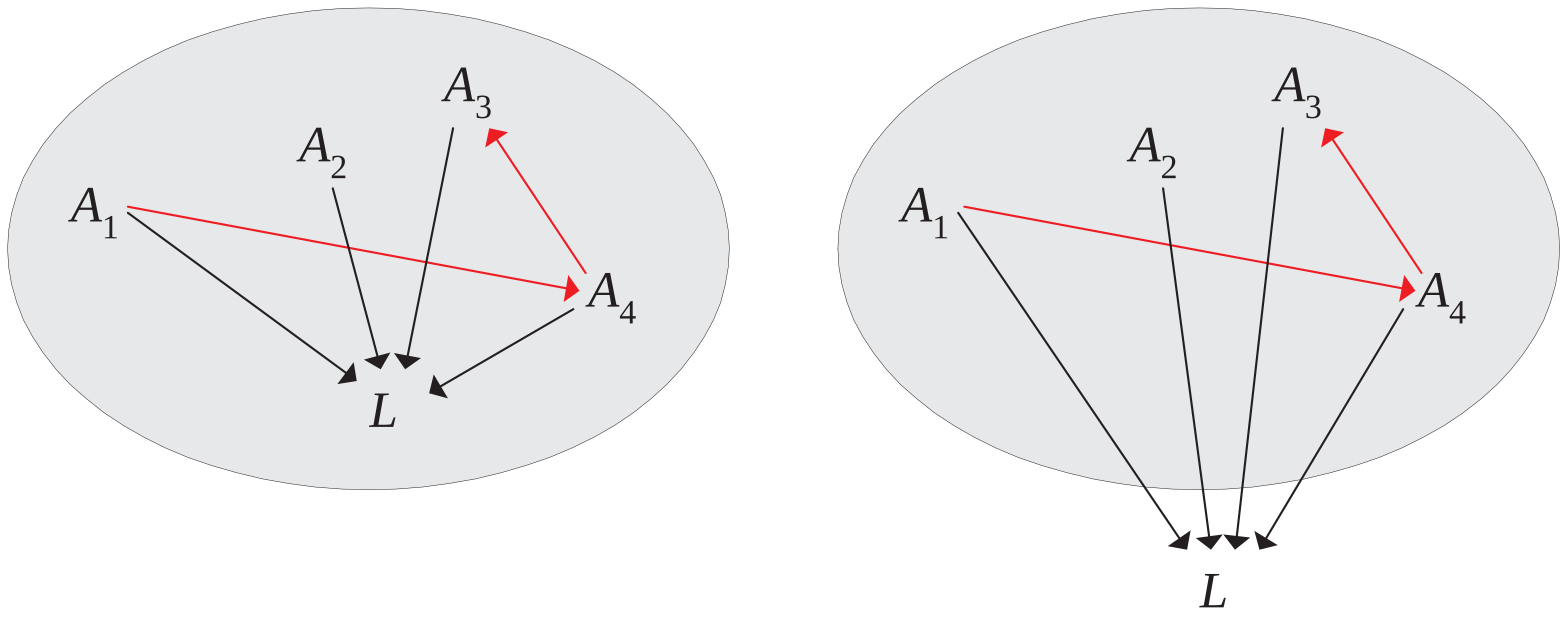
Hardness

- 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.

Complete Problems and Complexity Classes

Proposition 27 *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 27 implies the following.

Corollary 28 *$P = NP$ if and only if an NP-complete problem in P.*

Corollary 29 *$L = P$ if and only if a P-complete problem is in L.*

Complete Problems and Complexity Classes

Proposition 30 *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 $\mathcal{L} \in \mathcal{C}$ reduce to $L \in \mathcal{C}$ and $L \in \mathcal{C}'$.
- Since $\mathcal{C}'$ is closed under reductions, $\mathcal{L} \in \mathcal{C}'$.
- Hence $\mathcal{C} \subseteq \mathcal{C}'$.
- The proof for $\mathcal{C}' \subseteq \mathcal{C}$ is symmetric.

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.
- 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)

- Suppose 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 position j.

Comments

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

$$\overbrace{\triangleright \mathtt{0_s10001} \sqcup \sqcup \cdots \sqcup}^{|\,x\,|^k}$$

- A typical row may look like

$$\overbrace{\triangleright \mathtt{10100_q01110100} \sqcup \sqcup \cdots \sqcup}^{|\,x\,|^k}$$

Comments (concluded)

- The last rows must look like

$$\overbrace{\rhd \cdots \text{“yes”} \cdots \bigsqcup}^{|\,x\,|^k} \quad \text{or} \quad \overbrace{\rhd \cdots \text{“no”} \cdots \bigsqcup}^{|\,x\,|^k}$$

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

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

A P-Complete Problem

Theorem 31 (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 $\rhd$, and a $\sqcup$, respectively.
  - Recall that three out of T's 4 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 $T_{i-1,j+1}$:[a]

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

- 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[b]

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

[a]The dependency is "local."

[b]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:

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

- 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

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

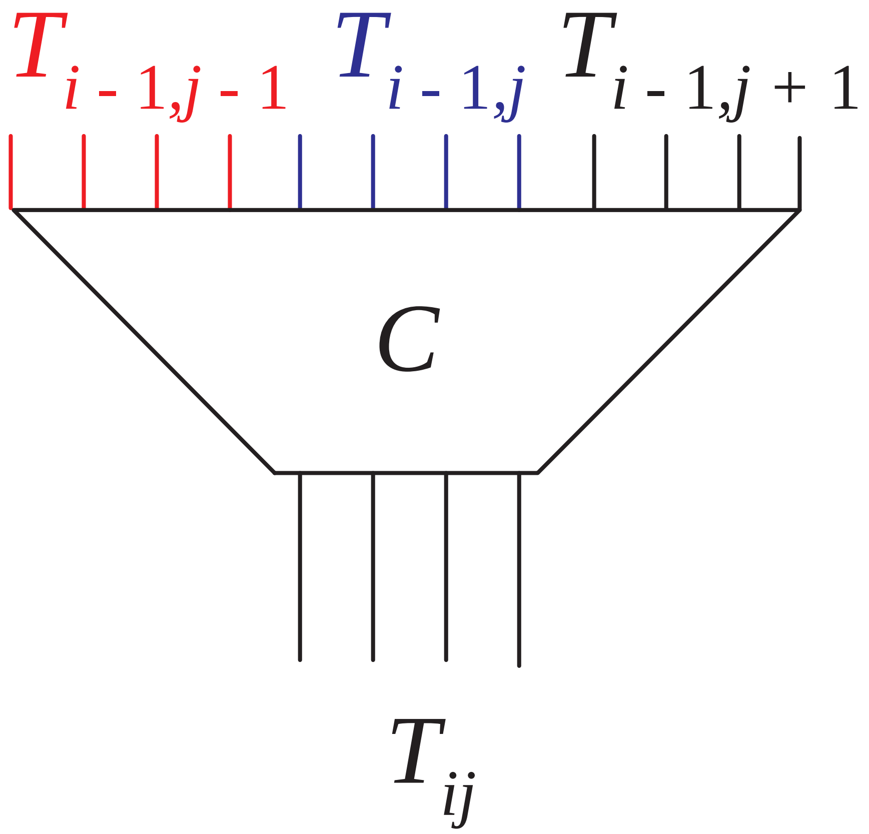
where for all $i, j > 0$ and $1 \leq \ell \leq m$.

The Proof (continued)

- These F_ℓ's depend only on M's specification, not on x.
- Their sizes are constant.
- These boolean functions can be turned into boolean circuits (see p. 191).
- Compose these m circuits in parallel to obtain circuit C with $3m$-bit inputs and m-bit outputs.
  - Schematically, $C(T_{i-1,j-1}, T_{i-1,j}, T_{i-1,j+1}) = T_{ij}$.[a]

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

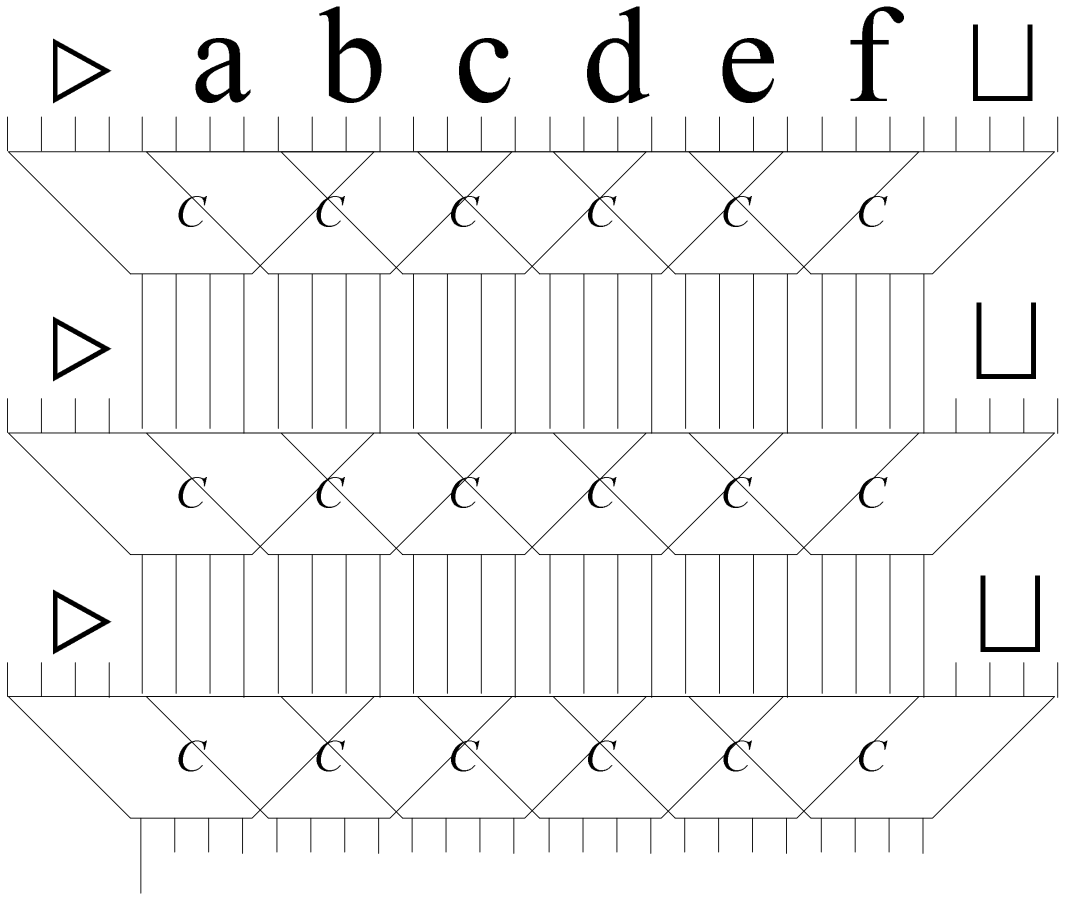
Circuit C



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 $R(x)$



$\triangleright$ a b c d e f $\sqcup$
C C C C C C
$\triangleright$ $\sqcup$
C C C C C C
$\triangleright$ $\sqcup$
C C C C C C

A Corollary

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

Corollary 32 *If* $L \in \mathit{TIME}(T(n))$, *then a circuit with* $O(T^2(n))$ *gates can decide if* $x \in L$ *for* $|\, x \,| = n$.

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 33 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.

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

Cook's Theorem: the First NP-Complete Problem

Theorem 34 (Cook (1971)) SAT *is NP-complete.*

- SAT $\in$ NP (p. 104).
- CIRCUIT SAT reduces to SAT (p. 261).
- Now we only need 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.

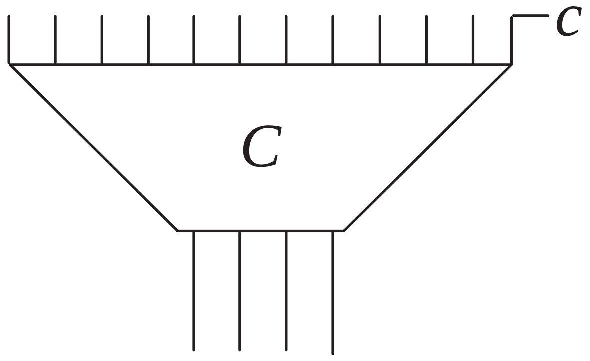
[a]The circuit for Theorem 31 (p. 282) will do.

The Proof (continued)

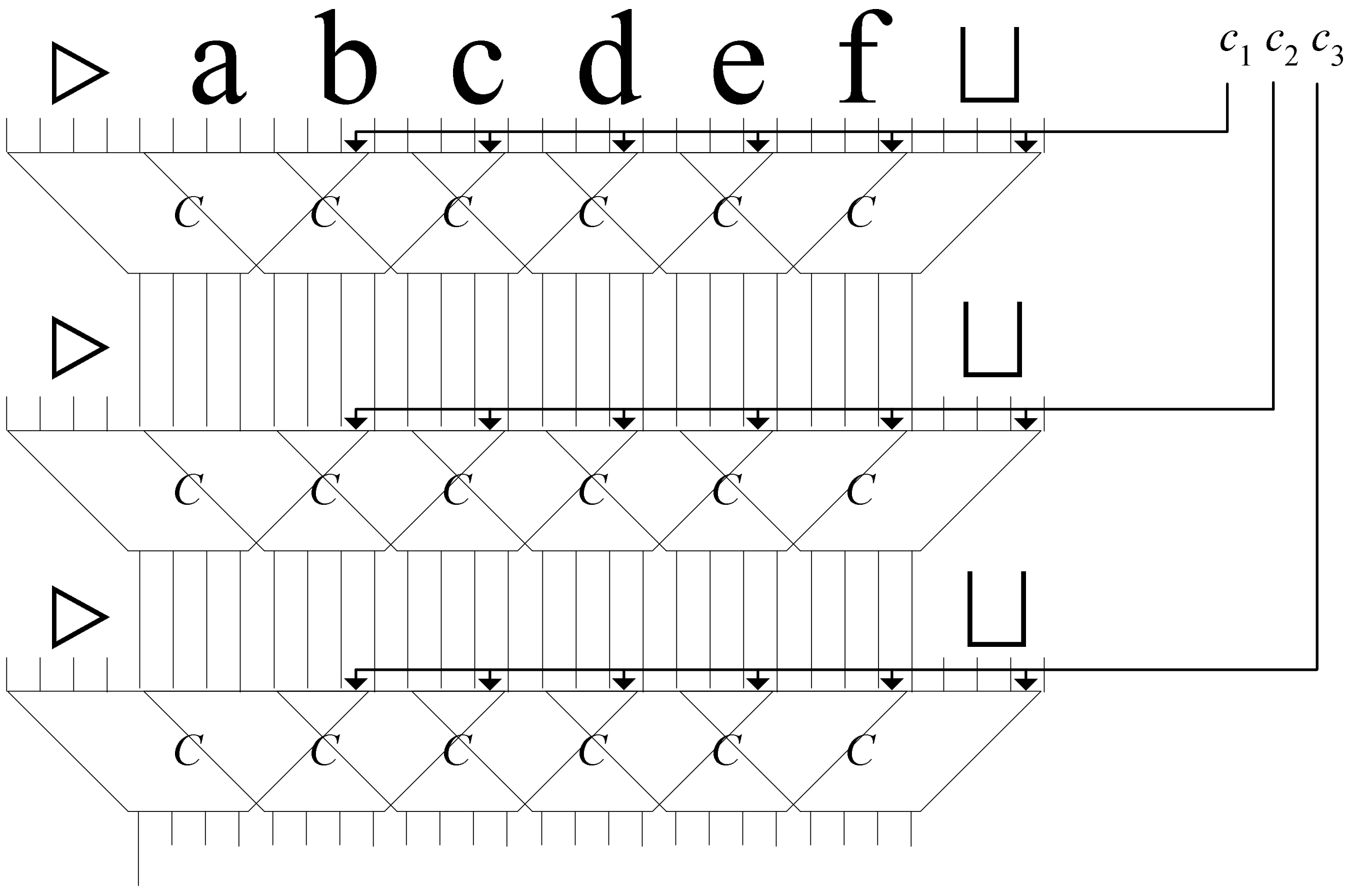
- A sequence of nondeterministic choices is a bit string

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

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


c
C

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

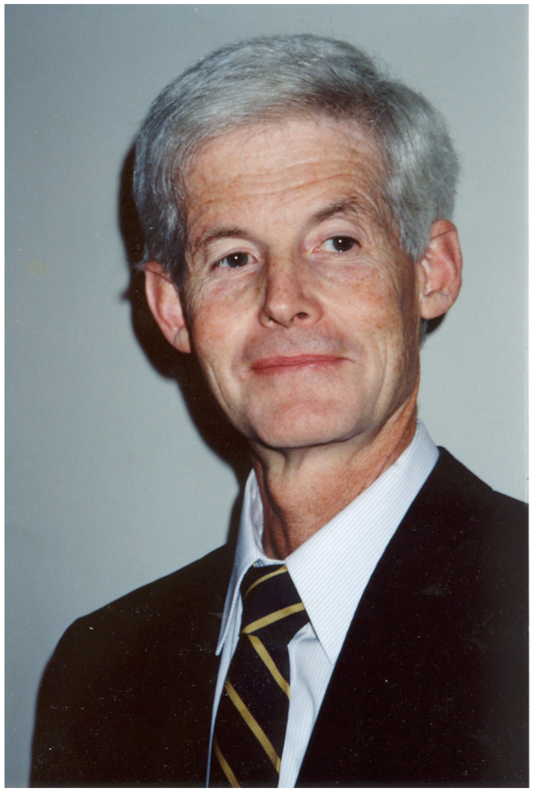


The Proof (concluded)

- Note that $c_1, c_2, \ldots, c_{|\, x\, |^k - 1}$ constitute the variables of $R(x)$.
- The overall circuit $R(x)$ (on p. 300) 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.

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 35 (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*

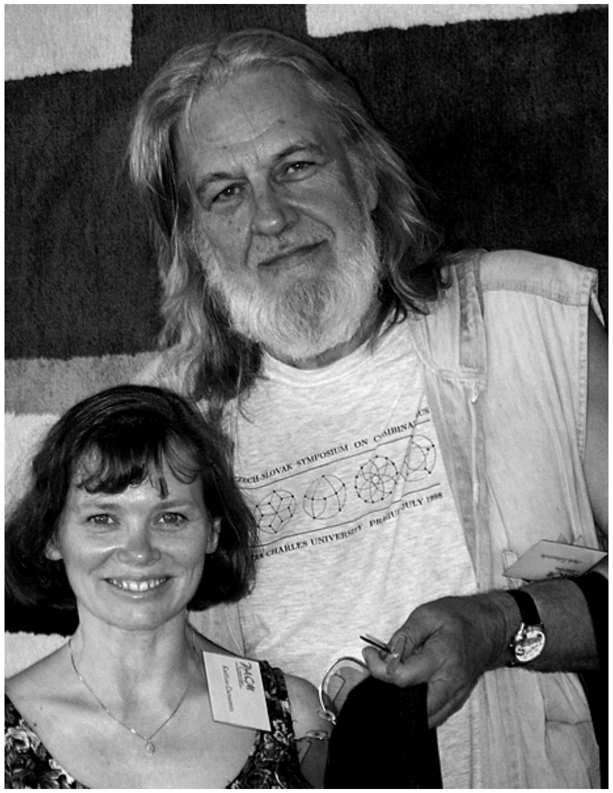
$$L = \{x : \exists y \, (x, y) \in R\}.$$

- 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 $L \in \mathrm{NP}$.
- 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 is polynomially bounded.
- 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–)


CZECH-SLOVAK SYMPOSIUM ON COMBINATORICS
CHARLES UNIVERSITY PRAGUE JULY 1998

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, etc.
- Certificates may be hard to generate,[a] but verification must be easy.
- NP is the class of *easy-to-verify* (i.e., in P) problems.

[a] Unless P equals NP.

Levin Reduction

- The reduction R in Cook's theorem (p. 295) is such that
  - Each satisfying truth assignment for circuit $R(x)$ corresponds to an accepting computation path for $M(x)$.
- It actually yields an efficient way to transform a certificate for x to a satisfying assignment for $R(x)$, and vice versa.
- A reduction with this property is called a **Levin reduction**.[a]

[a]Levin is the co-inventor of NP-completeness, in 1973.

Leonid Levin (1948–)

Leonid Levin (1998), "Mathematicians often think that historical evidence is that NP is exponential. Historical evidence is quite strongly in the other direction."