More Undecidability

- $H^* = \{M : M \text{ halts on all inputs}\}$.
  - Given the question "$M; x \in H$?" we construct the following machine:[a]

$$M_x(y) : M(x).$$

  - M_x halts on all inputs if and only if M halts on x.
  - In other words, $M_x \in H^*$ if and only if $M; x \in H$.
  - So if H^* were recursive, H would be recursive, a contradiction.

---

[a]Simplified by Mr. Chih-Hung Hsieh (D95922003) on October 5, 2006. M_x ignores its input y; x is part of M_x's code but not M_x's input.

More Undecidability (concluded)

- $\{M; x : \text{there is a } y \text{ such that } M(x) = y\}$.
- $\{M; x : \text{the computation } M \text{ on input } x \text{ uses all states of } M\}$.
- $\{M; x; y : M(x) = y\}$.

Complements of Recursive Languages

Lemma 9 *If L is recursive, then so is $\bar{L}$.*

- Let L be decided by M (which is deterministic).
- Swap the "yes" state and the "no" state of M.
- The new machine decides $\bar{L}$.

Recursive and Recursively Enumerable Languages

Lemma 10 *L is recursive if and only if both L and $\bar{L}$ are recursively enumerable.*

- Suppose both L and $\bar{L}$ are recursively enumerable, accepted by M and $\bar{M}$, respectively.
- Simulate M and $\bar{M}$ in an *interleaved* fashion.
- If M accepts, then $x \in L$ and M' halts on state "yes."
- If $\bar{M}$ accepts, then $x \notin L$ and M' halts on state "no."

A Very Useful Corollary and Its Consequences

Corollary 11 *L is recursively enumerable but not recursive, then $\bar{L}$ is not recursively enumerable.*

- Suppose $\bar{L}$ is recursively enumerable.
- Then both L and $\bar{L}$ are recursively enumerable.
- By Lemma 10 (p. 131), L is recursive, a contradiction.

Corollary 12 *$\bar{H}$ is not recursively enumerable.*

R, RE, and coRE

RE: The set of all recursively enumerable languages.

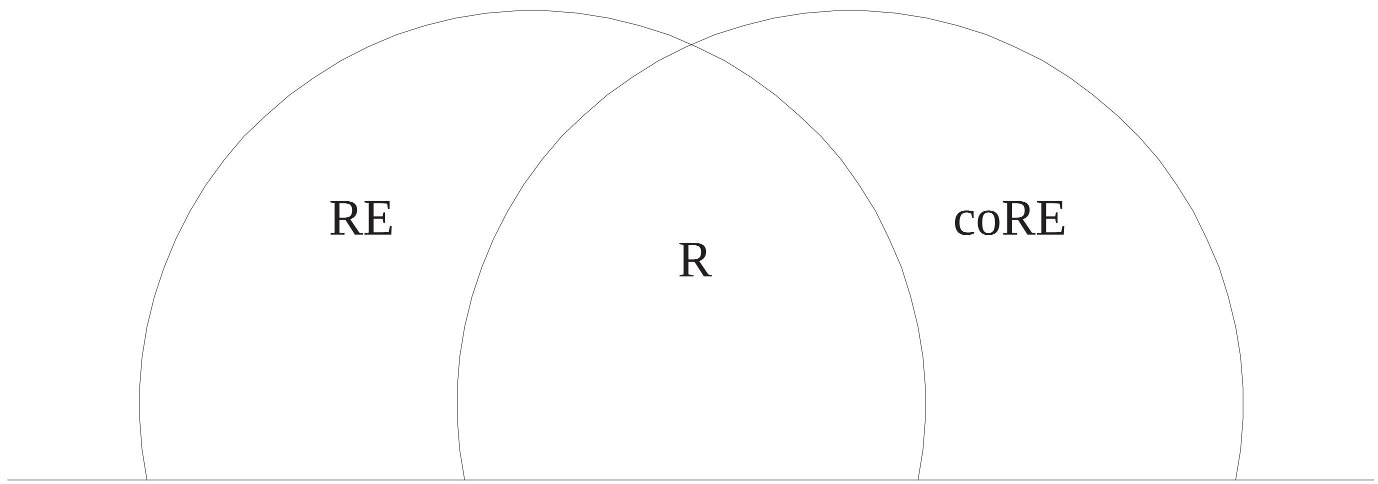
coRE: The set of all languages whose complements are recursively enumerable (note that coRE is not $\overline{\text{RE}}$).

- $\text{coRE} = \{ L : \overline{L} \in \text{RE} \}$.
- $\overline{\text{RE}} = \{ L : L \notin \text{RE} \}$.

R: The set of all recursive languages.

R, RE, and coRE (concluded)

- $\mathrm{R} = \mathrm{RE} \cap \mathrm{coRE}$ (p. 131).
- There exist languages in RE but not in R and not in coRE.
  - Such as H (p. 120, p. 121, and p. 132).
- There are languages in coRE but not in RE.
  - Such as $\bar{H}$ (p. 132).
- There are languages in neither RE nor coRE.


RE
R
coRE

Undecidability in Logic and Mathematics

- First-order logic is undecidable (answer to Hilbert's (1928) "*Entscheidungsproblem*").[a]
- Natural numbers with addition and multiplication is undecidable.[b]
- Rational numbers with addition and multiplication is undecidable.[c]

---

[a] Church (1936).
[b] Rosser (1937).
[c] Robinson (1948).

Undecidability in Logic and Mathematics (concluded)

- Natural numbers with addition and equality is decidable and complete.[a]
- Elementary theory of groups is undecidable.[b]

---

[a]Presburger's Master's thesis (1928), his only work in logic. The direction was suggested by Tarski. Mojžesz Presburger (1904–1943) died in a concentration camp during World War II.

[b]Tarski (1949).

Julia Hall Bowman Robinson (1919–1985)

Alfred Tarski (1901–1983)

Boolean Logic

It seemed unworthy of a grown man
to spend his time on such trivialities,
but what was I to do?
— Bertrand Russell (1872–1970),
Autobiography, Vol. I

Boolean Logic[a]

Boolean variables: $x_1, x_2, \ldots$.

Literals: x_i, $\neg x_i$.

Boolean connectives: $\vee, \wedge, \neg$.

Boolean expressions: Boolean variables, $\neg\phi$ (**negation**), $\phi_1 \vee \phi_2$ (**disjunction**), $\phi_1 \wedge \phi_2$ (**conjunction**).

- $\bigvee_{i=1}^{n} \phi_i$ stands for $\phi_1 \vee \phi_2 \vee \cdots \vee \phi_n$.
- $\bigwedge_{i=1}^{n} \phi_i$ stands for $\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$.

Implications: $\phi_1 \Rightarrow \phi_2$ is a shorthand for $\neg\phi_1 \vee \phi_2$.

Biconditionals: $\phi_1 \Leftrightarrow \phi_2$ is a shorthand for $(\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1)$.

---

[a] George Boole (1815–1864) in 1847.

Truth Assignments

- A **truth assignment** T is a mapping from boolean variables to **truth values** `true` and `false`.
- A truth assignment is **appropriate** to boolean expression ϕ if it defines the truth value for every variable in ϕ.
  - $\{x_1 = \texttt{true}, x_2 = \texttt{false}\}$ is appropriate to $x_1 \lor x_2$.
  - $\{x_2 = \texttt{true}, x_3 = \texttt{false}\}$ is not appropriate to $x_1 \lor x_2$.

Satisfaction

- $T \models \phi$ means boolean expression ϕ is true under T; in other words, T **satisfies** ϕ.
- ϕ_1 and ϕ_2 are **equivalent**, written

$$\phi_1 \equiv \phi_2,$$

  if for any truth assignment T appropriate to both of them, $T \models \phi_1$ if and only if $T \models \phi_2$.

Truth Tables

- Suppose ϕ has n boolean variables.
- A **truth table** contains 2^n rows.
- Each row corresponds to one truth assignment of the n variables and records the truth value of ϕ under that truth assignment.
- A truth table can be used to prove if two boolean expressions are equivalent.
  - Just check if they give identical truth values under all appropriate truth assignments.

A Truth Table

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

De Morgan's[a] Laws

- **De Morgan's laws** say that

$$\begin{aligned} \neg(\phi_1 \land \phi_2) &\equiv \neg\phi_1 \lor \neg\phi_2, \\ \neg(\phi_1 \lor \phi_2) &\equiv \neg\phi_1 \land \neg\phi_2. \end{aligned}$$

- Here is a proof of the first law:

ϕ_1	ϕ_2	$\neg(\phi_1 \land \phi_2)$	$\neg\phi_1 \lor \neg\phi_2$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

[a] Augustus DeMorgan (1806–1871).

Conjunctive Normal Forms

- A boolean expression ϕ is in **conjunctive normal form** (**CNF**) if
$$\phi = \bigwedge_{i=1}^{n} C_i,$$
where each **clause** C_i is the disjunction of zero or more literals.[a]
  - For example,
$$(x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (x_2 \vee x_3).$$
- Convention: An empty CNF is satisfiable, but a CNF containing an empty clause is not.

---

[a] Improved by Mr. Aufbu Huang (`R95922070`) on October 5, 2006.

Disjunctive Normal Forms

- A boolean expression ϕ is in **disjunctive normal form** (**DNF**) if

$$\phi = \bigvee_{i=1}^{n} D_i,$$

where each **implicant** D_i is the conjunction of one or more literals.

  – For example,

$$(x_1 \land x_2) \lor (x_1 \land \neg x_2) \lor (x_2 \land x_3).$$

Any Expression ϕ Can Be Converted into CNFs and DNFs

$\phi = x_j$:

- This is trivially true.

$\phi = \neg\phi_1$ and a CNF is sought:

- Turn ϕ_1 into a DNF.
- Apply de Morgan's laws to make a CNF for ϕ.

$\phi = \neg\phi_1$ and a DNF is sought:

- Turn ϕ_1 into a CNF.
- Apply de Morgan's laws to make a DNF for ϕ.

Any Expression ϕ Can Be Converted into CNFs and DNFs (continued)

$\phi = \phi_1 \vee \phi_2$ and a DNF is sought:

- Make ϕ_1 and ϕ_2 DNFs.

$\phi = \phi_1 \vee \phi_2$ and a CNF is sought:

- Turn ϕ_1 and ϕ_2 into CNFs,[a]

$$\phi_1 = \bigwedge_{i=1}^{n_1} A_i, \quad \phi_2 = \bigwedge_{j=1}^{n_2} B_j.$$

- Set

$$\phi = \bigwedge_{i=1}^{n_1} \bigwedge_{j=1}^{n_2} (A_i \vee B_j).$$

---

[a] Corrected by Mr. Chun-Jie Yang (`R99922150`) on November 9, 2010.

Any Expression ϕ Can Be Converted into CNFs and DNFs (concluded)

$\phi = \phi_1 \wedge \phi_2$ and a CNF is sought:

- Make ϕ_1 and ϕ_2 CNFs.

$\phi = \phi_1 \wedge \phi_2$ and a DNF is sought:

- Turn ϕ_1 and ϕ_2 into DNFs,

$$\phi_1 = \bigvee_{i=1}^{n_1} A_i, \quad \phi_2 = \bigvee_{j=1}^{n_2} B_j.$$

- Set

$$\phi = \bigvee_{i=1}^{n_1} \bigvee_{j=1}^{n_2} (A_i \wedge B_j).$$

An Example: Turn $\neg((a \wedge y) \vee (z \vee w))$ into a DNF

$$
\begin{aligned}
& \neg((a \wedge y) \vee (z \vee w)) \\
\stackrel{\neg(\text{CNF}\vee\text{CNF})}{=} \quad & \neg(((a) \wedge (y)) \vee ((z \vee w))) \\
\stackrel{\neg(\text{CNF})}{=} \quad & \neg((a \vee z \vee w) \wedge (y \vee z \vee w)) \\
\stackrel{\text{de Morgan}}{=} \quad & \neg(a \vee z \vee w) \vee \neg(y \vee z \vee w) \\
= \quad & (\neg a \wedge \neg z \wedge \neg w) \vee (\neg y \wedge \neg z \wedge \neg w).
\end{aligned}
$$

Satisfiability

- A boolean expression ϕ is **satisfiable** if there is a truth assignment T appropriate to it such that $T \models \phi$.
- ϕ is **valid** or a **tautology**,[a] written $\models \phi$, if $T \models \phi$ for all T appropriate to ϕ.
- ϕ is **unsatisfiable** if and only if ϕ is false under all appropriate truth assignments if and only if $\neg\phi$ is valid.

---

[a] Wittgenstein (1889–1951) in 1922. Wittgenstein is one of the most important philosophers of all time. "God has arrived," the great economist Keynes (1883–1946) said of him on January 18, 1928. "I met him on the 5:15 train." Russell (1919), "The importance of 'tautology' for a definition of mathematics was pointed out to me by my former pupil Ludwig Wittgenstein, who was working on the problem. I do not know whether he has solved it, or even whether he is alive or dead."

Ludwig Wittgenstein (1889–1951)

Wittgenstein (1922), "Whereof one cannot speak, thereof one must be silent."

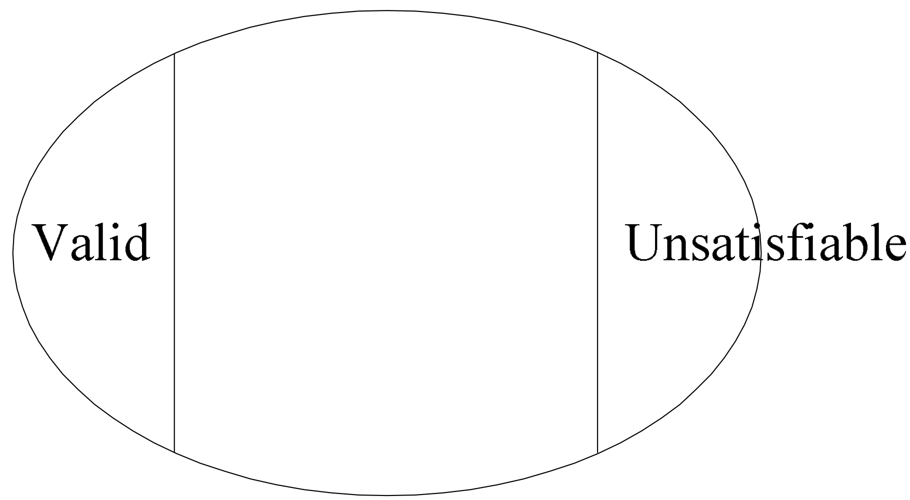
SATISFIABILITY (SAT)

- The **length** of a boolean expression is the length of the string encoding it.
- SATISFIABILITY (SAT): Given a CNF ϕ, is it satisfiable?
- Solvable in exponential time on a TM by the truth table method.
- Solvable in polynomial time on an NTM, hence in NP (p. 87).
- A most important problem in settling the "P $\stackrel{?}{=}$ NP" problem (p. 262).

UNSATISFIABILITY (UNSAT or SAT COMPLEMENT) and VALIDITY

- UNSAT (SAT COMPLEMENT): Given a boolean expression ϕ, is it unsatisfiable?
- VALIDITY: Given a boolean expression ϕ, is it valid?
  - ϕ is valid if and only if $\neg\phi$ is unsatisfiable.
  - ϕ and $\neg\phi$ are basically of the same length.
  - So UNSAT and VALIDITY have the same complexity.
- Both are solvable in exponential time on a TM by the truth table method.

Relations among SAT, UNSAT, and VALIDITY



- The negation of an unsatisfiable expression is a valid expression.
- None of the three problems—satisfiability, unsatisfiability, validity—are known to be in P.

Boolean Functions

- An n-ary boolean function is a function

$$f : \{\texttt{true}, \texttt{false}\}^n \to \{\texttt{true}, \texttt{false}\}.$$

- It can be represented by a truth table.
- There are 2^{2^n} such boolean functions.
  - We can assign `true` or `false` to f under each of the 2^n truth assignments.

Boolean Functions (continued)

Assignment	Truth value
1	`true` or `false`
2	`true` or `false`
$\vdots$	$\vdots$
2^n	`true` or `false`

Boolean Functions (continued)

- A boolean expression expresses a boolean function.
  - Think of its truth value under all truth assignments.
- A boolean function expresses a boolean expression.
  - $\bigvee_{T \models \phi,\ \text{literal } y_i \text{ is true in "row" } T} (y_1 \wedge \cdots \wedge y_n)$.
    - $y_1 \wedge \cdots \wedge y_n$ is called the **minterm** over $\{x_1, \ldots, x_n\}$ for T.
  - The size[a] is $\leq n2^n \leq 2^{2n}$.

[a] We count only the literals here.

Boolean Functions (continued)

x_1	x_2	$f(x_1, x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

The corresponding boolean expression:

$$(\neg x_1 \land \neg x_2) \lor (\neg x_1 \land x_2) \lor (x_1 \land x_2).$$

Boolean Functions (concluded)

Corollary 13 *Every n-ary boolean function can be expressed by a boolean expression of size $O(n2^n)$.*

- In general, the exponential length in n cannot be avoided (p. 169).
- The size of the truth table is also $O(n2^n)$.

Boolean Circuits

- A **boolean circuit** is a graph C whose nodes are the **gates**.
- There are no cycles in C.
- All nodes have indegree (number of incoming edges) equal to 0, 1, or 2.
- Each gate has a **sort** from

$$\{\texttt{true}, \texttt{false}, \vee, \wedge, \neg, x_1, x_2, \ldots\}.$$

Boolean Circuits (concluded)

- Gates with a sort from $\{\texttt{true}, \texttt{false}, x_1, x_2, \ldots\}$ are the **inputs** of C and have an indegree of zero.
- The **output gate**(s) has no outgoing edges.
- A boolean circuit computes a boolean function.
- The same boolean function can be computed by infinitely many boolean circuits.

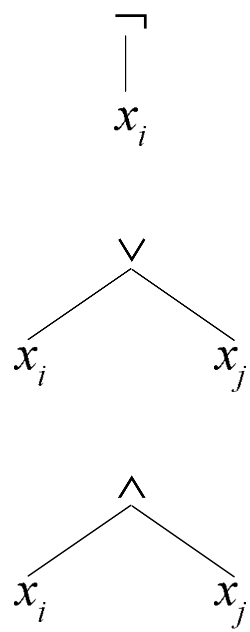
Boolean Circuits and Expressions

- They are equivalent representations.
- One can construct one from the other:

$\neg x_i$

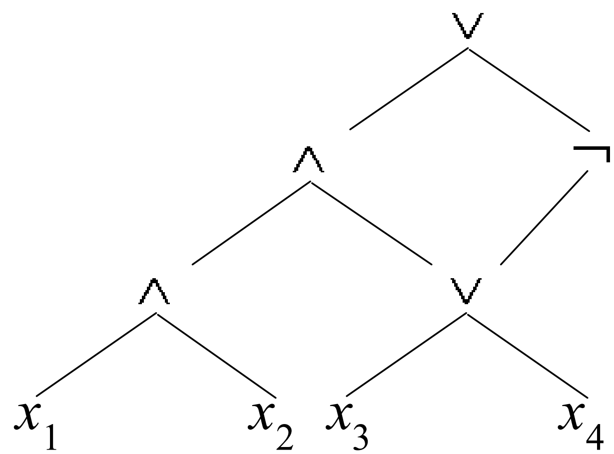
$x_i \lor x_j$

$x_i \land x_j$



An Example

$$((x_1 \land x_2) \land (x_3 \lor x_4)) \lor (\neg(x_3 \lor x_4))$$



- Circuits are more economical because of the possibility of sharing.

CIRCUIT SAT and CIRCUIT VALUE

CIRCUIT SAT: Given a circuit, is there a truth assignment such that the circuit outputs true?

- CIRCUIT SAT $\in$ NP: Guess a truth assignment and then evaluate the circuit.

CIRCUIT VALUE: The same as CIRCUIT SAT except that the circuit has no variable gates.

- CIRCUIT VALUE $\in$ P: Evaluate the circuit from the input gates gradually towards the output gate.

Some Boolean Functions Need Exponential Circuits[a]

Theorem 14 (Shannon (1949)) *For any $n \geq 2$, there is an n-ary boolean function f such that no boolean circuits with $2^n/(2n)$ or fewer gates can compute it.*

- There are 2^{2^n} different n-ary boolean functions (p. 159).
- So it suffices to prove that the number of boolean circuits with $2^n/(2n)$ or fewer gates is less than 2^{2^n}.

---

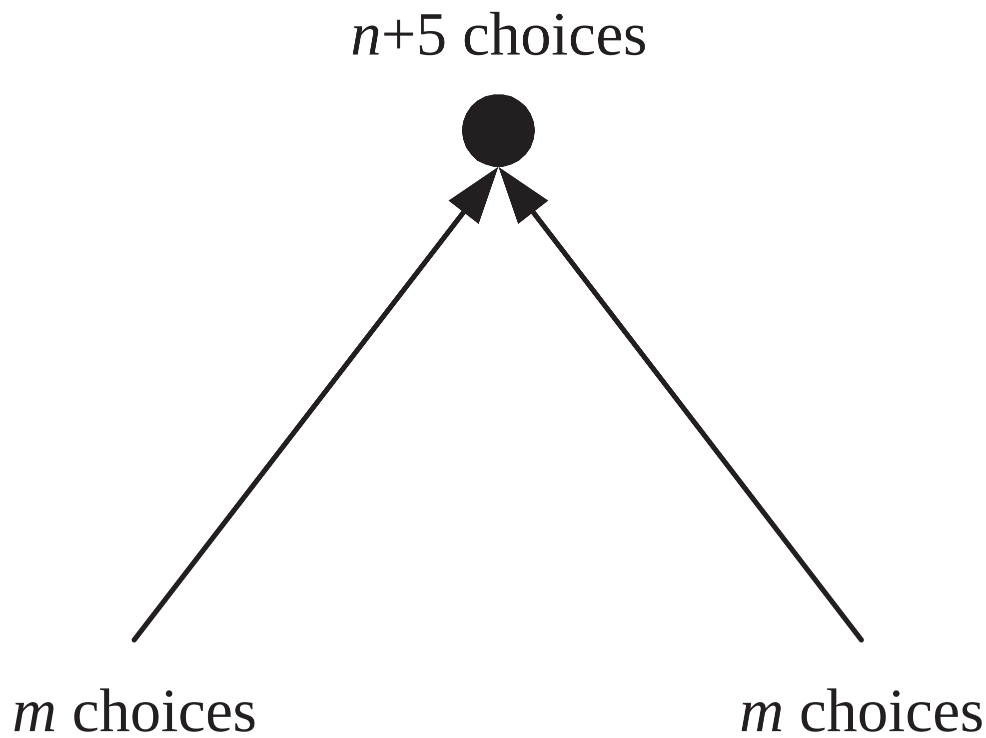
[a] Can be strengthened to "almost all boolean functions ..."

The Proof (concluded)

- There are at most $((n+5) \times m^2)^m$ boolean circuits with m or fewer gates (see next page).
- But $((n+5) \times m^2)^m < 2^{2^n}$ when $m = 2^n/(2n)$:

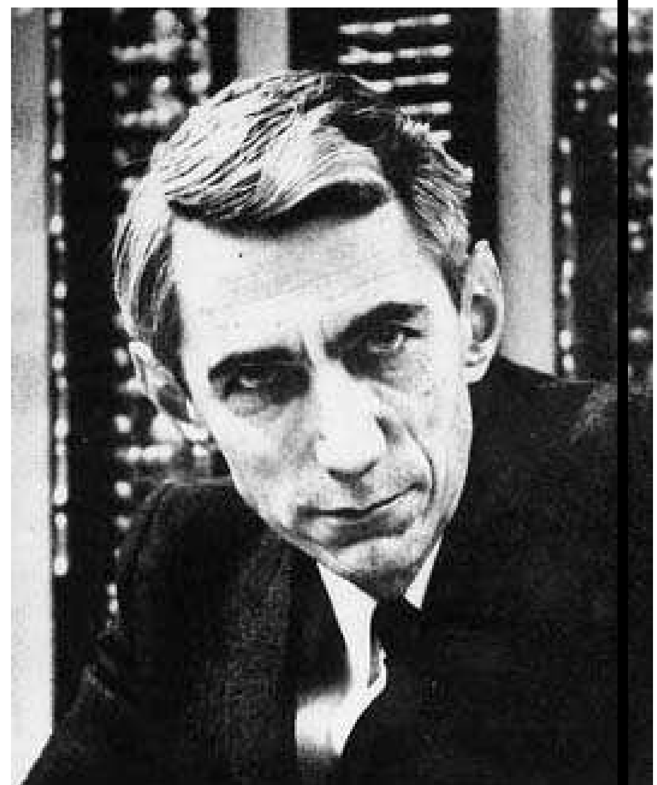
$$
\begin{aligned}
& m \log_2((n+5) \times m^2) \\
= \quad & 2^n \left(1 - \frac{\log_2 \frac{4n^2}{n+5}}{2n} \right) \\
< \quad & 2^n
\end{aligned}
$$

for $n \geq 2$.



n+5 choices
m choices
m choices

Claude Elwood Shannon (1916–2001)



Howard Gardner, "[Shannon's master's thesis is] possibly the most important, and also the most famous, master's thesis of the century."

Comments

- The lower bound $2^n/(2n)$ is rather tight because an upper bound is $n2^n$ (p. 161).
- The proof counted the number of circuits.
  - Some circuits may not be valid at all.
  - Different circuits may also compute the same function.
- Both are fine because we only need an upper bound on the number of circuits.
- We do not need to consider the outdoing edges because they have been counted as incoming edges.

Relations between Complexity Classes

Proper (Complexity) Functions

- We say that $f : \mathbb{N} \to \mathbb{N}$ is a **proper (complexity) function** if the following hold:
  - f is nondecreasing.
  - There is a k-string TM M_f such that $M_f(x) = \sqcap^{f(|x|)}$ for any x.[a]
  - M_f halts after $O(|x| + f(|x|))$ steps.
  - M_f uses $O(f(|x|))$ space besides its input x.
- M_f's behavior depends only on $|x|$ not x's contents.
- M_f's running time is bounded by $f(n)$.

[a]This point will become clear in Proposition 15 (p. 178).

Examples of Proper Functions

- Most "reasonable" functions are proper: c, $\lceil \log n \rceil$, polynomials of n, 2^n, $\sqrt{n}$, $n!$, etc.
- If f and g are proper, then so are $f + g$, fg, and 2^g.
- Nonproper functions when serving as the time bounds for complexity classes spoil "the theory building."
  - For example, $\text{TIME}(f(n)) = \text{TIME}(2^{f(n)})$ for some recursive function f (the **gap theorem**).[a]
- Only proper functions f will be used in $\text{TIME}(f(n))$, $\text{SPACE}(f(n))$, $\text{NTIME}(f(n))$, and $\text{NSPACE}(f(n))$.

[a] Trakhtenbrot (1964); Borodin (1972).

Precise Turing Machines

- A TM M is **precise** if there are functions f and g such that for every $n \in \mathbb{N}$, for every x of length n, and for every computation path of M,
  - M halts after precisely $f(n)$ steps, and
  - All of its strings are of length precisely $g(n)$ at halting.
    * Recall that if M is a TM with input and output, we exclude the first and the last strings.
- M can be deterministic or nondeterministic.

Precise TMs Are General

Proposition 15 *Suppose a TM*[a] *M decides L within time (space) $f(n)$, where f is proper. Then there is a precise TM M' which decides L in time $O(n + f(n))$ (space $O(f(n))$, respectively).*

- M' on input x first simulates the TM M_f associated with the proper function f on x.
- M_f's output of length $f(|\,x\,|)$ will serve as a "yardstick" or an "alarm clock."
- $M'(x)$ halts when and only when the alarm clock runs out—even if M halts earlier.

[a] It can be deterministic or nondeterministic.

Important Complexity Classes

- We write expressions like n^k to denote the union of all complexity classes, one for each value of k.
- For example,

$$\text{NTIME}(n^k) = \bigcup_{j>0} \text{NTIME}(n^j).$$

Important Complexity Classes (concluded)

$$\begin{aligned}
\text{P} &= \text{TIME}(n^k), \\
\text{NP} &= \text{NTIME}(n^k), \\
\text{PSPACE} &= \text{SPACE}(n^k), \\
\text{NPSPACE} &= \text{NSPACE}(n^k), \\
\text{E} &= \text{TIME}(2^{kn}), \\
\text{EXP} &= \text{TIME}(2^{n^k}), \\
\text{L} &= \text{SPACE}(\log n), \\
\text{NL} &= \text{NSPACE}(\log n).
\end{aligned}$$