Notations

- Suppose M is a TM accepting L.
- Write $L(M) = L$.
  - In particular, if $M(x) = \nearrow$ for all x, then $L(M) = \emptyset$.
- If $M(x)$ is never "yes" nor $\nearrow$ (as required by the definition of acceptance), we also let $L(M) = \emptyset$.

Nontrivial Properties of Sets in RE

- A **property** of the recursively enumerable languages can be defined by the set $\mathcal{C}$ of all the recursively enumerable languages that satisfy it.
  - The property of *finite* recursively enumerable languages is

$$\{ L : L = L(M) \text{ for a TM } M,\, L \text{ is finite} \}.$$

- A property is **trivial** if $\mathcal{C} = \text{RE}$ or $\mathcal{C} = \emptyset$.
  - Answer to a trivial property is always "yes"or always "no."

Nontrivial Properties of Sets in RE (concluded)

- Here is a trivial property (always yes): Does the TM accept a recursively enumerable language?[a]
- A property is **nontrivial** if $\mathcal{C} \neq \text{RE}$ and $\mathcal{C} \neq \emptyset$.
  - In other words, answer to a nontrivial property is "yes" for some TMs and "no" for others.
- Here is a nontrivial property: Does the TM accept an empty language?[b]
- Up to now, all nontrivial properties (of recursively enumerable languages) are undecidable (pp. 156–157).
- In fact, Rice's theorem confirms that.

[a] Or, $L(M) \in \text{RE}$?
[b] Or, $L(M) = \emptyset$?

Rice's Theorem

Theorem 13 (Rice, 1956) *Suppose $\mathcal{C} \neq \emptyset$ is a proper subset of the set of all recursively enumerable languages.*[a] *Then the question "$L(M) \in \mathcal{C}$?" is undecidable.*

- Note that the input is a TM program M.
- Assume that $\emptyset \notin \mathcal{C}$ (otherwise, repeat the proof for the class of all recursively enumerable languages *not* in $\mathcal{C}$).
- Let $L \in \mathcal{C}$ be accepted by TM M_L (recall that $\mathcal{C} \neq \emptyset$).
- Let M_H *accept* the undecidable language H.
  - M_H exists (p. 139).

---

[a] A nontrivial property, i.e.

The Proof (continued)

- Construct machine $M_x(y)$:

$$\textbf{if } M_H(x) = \text{“yes”} \textbf{ then } M_L(y) \textbf{ else } \nearrow$$

- On the next page, we will prove that

$$x \in H \text{ if and only if } L(M_x) \in \mathcal{C}. \tag{1}$$

  – As a result, the halting problem is reduced to deciding $L(M_x) \in \mathcal{C}$.
  – Hence $L(M_x) \in \mathcal{C}$ must be undecidable, and we are done.

The Proof (concluded)

- Suppose $x \in H$, i.e., $M_H(x) =$ "yes."
  - $M_x(y)$ determines this, and it either accepts y or never halts, depending on whether $y \in L$.
  - Hence $L(M_x) = L \in \mathcal{C}$.
- Suppose $M_H(x) = \nearrow$.
  - M_x never halts.
  - $L(M_x) = \emptyset \notin \mathcal{C}$.

Comments

- $\mathcal{C}$ must be arbitrary.
- The following $M_x(y)$, though similar, will not work:

$$\textbf{if } M_L(y) = \text{“yes”} \textbf{ then } M_H(x) \textbf{ else } \nearrow.$$

- Rice's theorem is about properties of the languages accepted by Turing machines.
- It then says any nontrivial property is undecidable.
- Rice's theorem is *not* about Turing machines themselves, such as "Does a TM contain 5 states?"

Consequences of Rice's Theorem

Corollary 14 *The following properties of recursively enumerative sets are undecidable.*

- *Emptiness.*
- *Finiteness.*
- *Recursiveness.*
- Σ^*.
- *Regularity.*[a]
- *Context-freedom.*[b]

---

[a] Is it a regular language?
[b] Is it a context-free language?

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

Christianity is either false or true.
— Girolamo Savonarola (1497)

Both of us had said the very same thing.
Did we both speak the truth
—or one of us did
—or neither?
— Joseph Conrad (1857–1924),
Lord Jim (1900)

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 Table[a]

- 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 it.
- 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] Post (1921); Wittgenstein (1922). Here, 1 is used to denote `true`; 0 is used to denote `false`. This is called the **standard representation** (Beigel, 1993).

A Truth Table

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

A Second Truth Table

p	q	$p \lor q$
0	0	0
0	1	1
1	0	1
1	1	1

A Third Truth Table

p	$\neg p$
0	1
1	0

Proof of Equivalency by the Truth Table:

$$p \Rightarrow q \equiv \neg q \Rightarrow \neg p$$

p	q	$p \Rightarrow q$	$\neg q \Rightarrow \neg p$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

De Morgan's Laws[a]

- **De Morgan's laws** state 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) or William of Ockham (1288–1348).

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,
$$(x_1 \lor x_2) \land (x_1 \lor \neg x_2) \land (x_2 \lor x_3).$$
- 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**[a] or simply **term** D_i is the conjunction of zero or more literals.

  - For example,

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

[a] D_i implies ϕ, thus the term.

Clauses and Implicants

- The $\bigvee$ of clauses yields a clause.
  - For example,

$$\begin{aligned} & (x_1 \lor x_2) \lor (x_1 \lor \neg x_2) \lor (x_2 \lor x_3) \\ = \quad & x_1 \lor x_2 \lor x_1 \lor \neg x_2 \lor x_2 \lor x_3. \end{aligned}$$

- The $\bigwedge$ of implicants yields an implicant.
  - For example,

$$\begin{aligned} & (x_1 \land x_2) \land (x_1 \land \neg x_2) \land (x_2 \land x_3) \\ = \quad & x_1 \land x_2 \land x_1 \land \neg x_2 \land x_2 \land x_3. \end{aligned}$$

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

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

Functional Completeness

- A set of logical connectives is called **functionally complete** if every boolean expression is equivalent to one involving only these connectives.
- The set $\{ \neg, \vee, \wedge \}$ is functionally complete.
  - Every boolean expression can be turned into a CNF, which involves only $\neg$, $\vee$, and $\wedge$.
- The sets $\{ \neg, \vee \}$ and $\{ \neg, \wedge \}$ are functionally complete.[a]
  - By the above result and de Morgan's laws.
- { NAND } and { NOR } are functionally complete.[b]

---

[a] Post (1921).
[b] Peirce (c. 1880); Sheffer (1913).

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.

---

[a]Wittgenstein (1922). Wittgenstein is one of the most important philosophers of all time. 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." "God has arrived," the great economist Keynes (1883–1946) said of him on January 18, 1928, "I met him on the 5:15 train."

Satisfiability (concluded)

- ϕ is **unsatisfiable** or a **contradiction** if ϕ is false under all appropriate truth assignments.
  - Or, equivalently, if $\neg\phi$ is valid (prove it).
- ϕ is a **contingency** if ϕ is neither a tautology nor a contradiction.

Ludwig Wittgenstein (1889–1951)

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

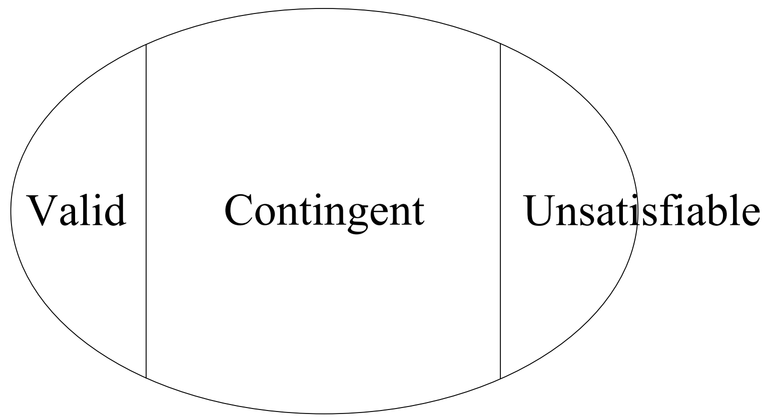
SATISFIABILITY (SAT)

- 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. 121).
- A most important problem in settling the "$\mathrm{P} \stackrel{?}{=} \mathrm{NP}$" problem (p. 323).

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 for 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`

- A boolean expression expresses a boolean function.
  - Think of its truth values under all possible truth assignments.

Boolean Functions (continued)

- A boolean function expresses a boolean expression.
  - $\bigvee_{T \,\models\, \phi,\text{ literal } y_i \text{ is true in ``row'' } T} (y_1 \land \cdots \land y_n)$.[a]
    - The implicant $y_1 \land \cdots \land y_n$ is called the **minterm** over $\{ x_1, \ldots, x_n \}$ for T.
  - The size[b] is $\leq n2^n \leq 2^{2n}$.
  - This DNF is optimal for the parity function, for example.[c]

---

[a]Similar to **programmable logic array**. This is called the **table lookup representation** (Beigel, 1993).

[b]We count only the literals here.

[c]Du & Ko (2000).

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 15 *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. 212).
- The size of the truth table is also $O(n2^n)$.[a]

[a]There are 2^n n-bit strings.

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

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

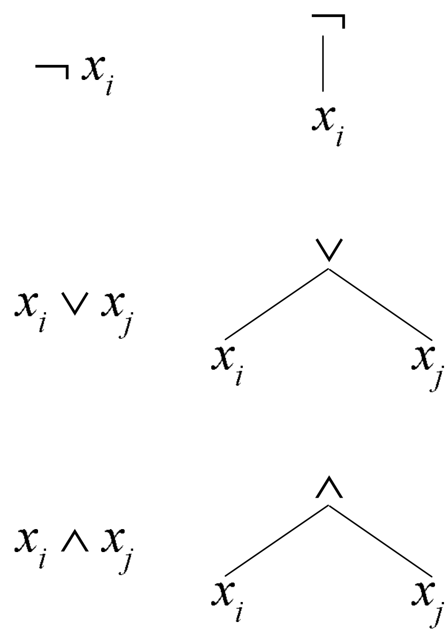
  – There are $n + 5$ sorts.

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.
- A boolean function can be realized by *infinitely many* equivalent boolean circuits.

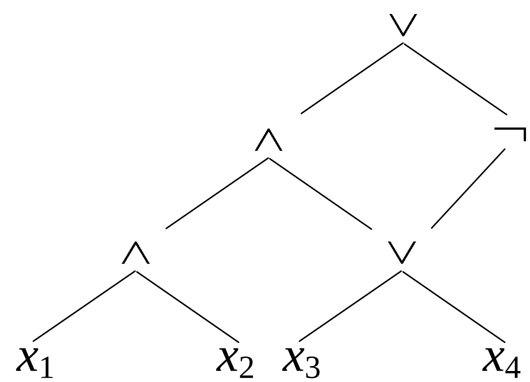
Boolean Circuits and Expressions

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



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.[a]

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.

---

[a]Essentially the same algorithm as the one on p. 121.

Some[a] Boolean Functions Need Exponential Circuits[b]

Theorem 16 *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. 202).
- 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."

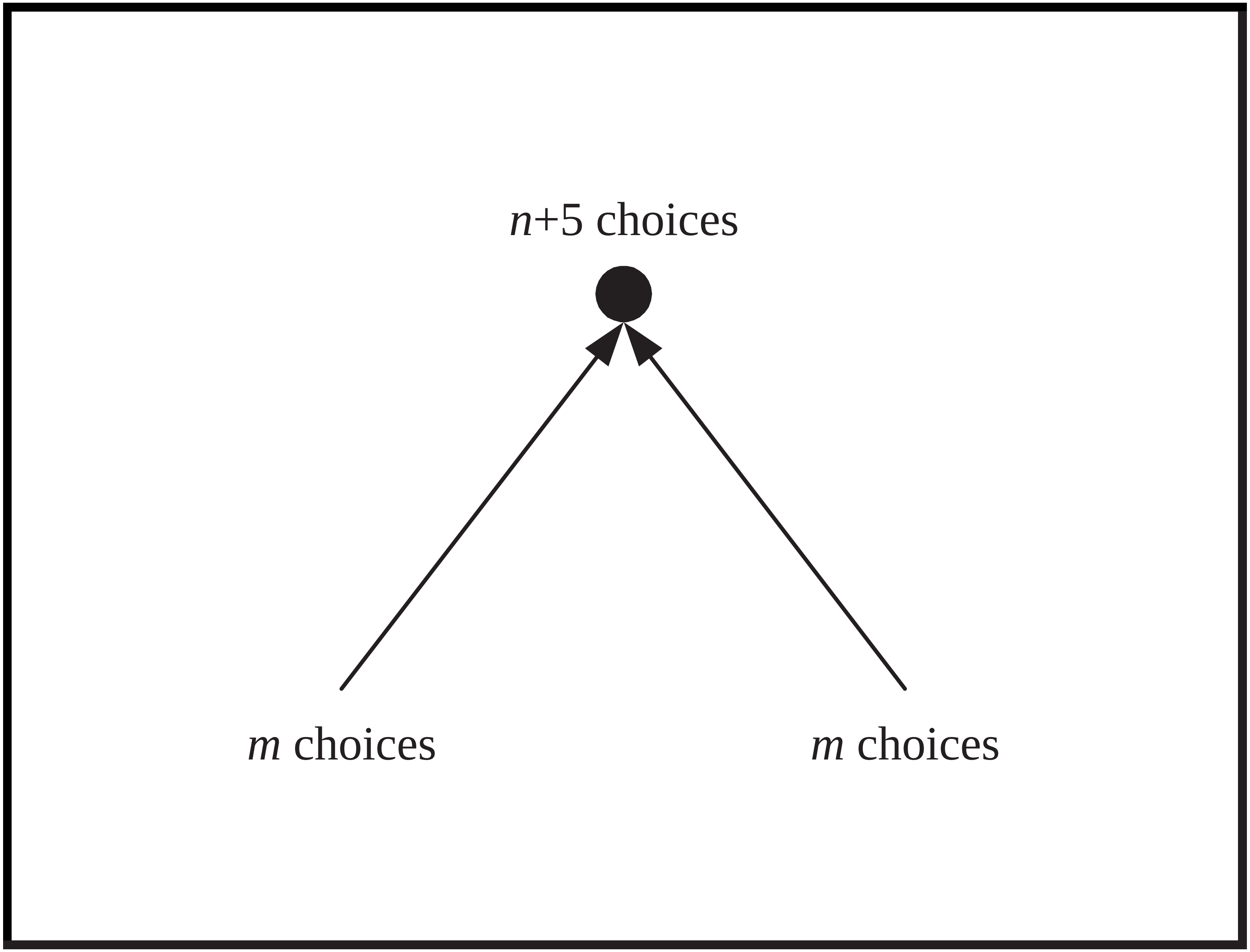
[b] Riordan & Shannon (1942); Shannon (1949).

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 $m = 2^n/(2n)$:

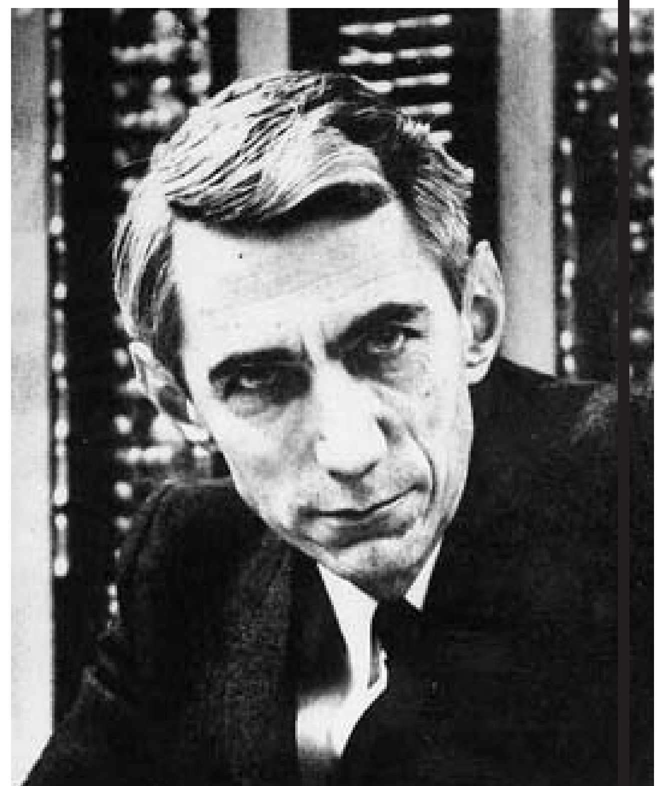
$$
\begin{aligned}
& m \log_2((n+5) \times m^2) \\
= \quad & 2^n \left(1 - \frac{\log_2 \frac{4n^2}{n+5}}{2n} \right) \\
< \quad & 2^n
\end{aligned}
$$

for $n \geq 2$.



n+5 choices
m choices
m choices

Claude Elwood Shannon (1916–2001)



Howard Gardner (1987), "[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. 204).
- 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 *outgoing* edges because they have been counted as incoming edges.[a]

[a] If you prove the theorem by considering outgoing edges, the bound will not be good. (Try it!)

Relations between Complexity Classes

It is, I own, not uncommon to be
wrong in theory
and right in practice.
— Edmund Burke (1729–1797),
A Philosophical Enquiry into the Origin of Our Ideas of the Sublime and Beautiful (1757)

The problem with QE is
it works in practice,
but it doesn't work in theory.
— Ben Bernanke (2014)

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]The textbook calls "$\sqcap$" the quasi-blank symbol. The use of $M_f(x)$ will become clear in Proposition 17 (p. 222).

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.[a]
- Nonproper functions when serving as the time bounds for complexity classes spoil "theory building."
  - For example, $\text{TIME}(f(n)) = \text{TIME}(2^{f(n)})$ for some recursive function f (the **gap theorem**).[b]
- 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] For $f(g(n))$, we need to add $f(n) \geq n$.

[b] Trakhtenbrot (1964); Borodin (1972). Theorem 7.3 on p. 145 of the textbook proves it.

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,[a] and
  - All of its strings are of length precisely[b] $g(n)$ at halting.[c]
    * Recall that if M is a TM with input and output, we exclude the first and last strings.
- M can be deterministic or nondeterministic.

---

[a]Fully time constructible (Hopcroft & Ullman, 1979).
[b]This strong requirement does not seem needed later.
[c]Fully space constructible (Hopcroft & Ullman, 1979).

Precise TMs Are General

Proposition 17 *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."

[a]It can be deterministic or nondeterministic.

The Proof (continued)

- Then M' simulates $M(x)$.
- $M'(x)$ halts when and only when the alarm clock runs out—even if M halts earlier.
- If f is a time bound:
  - The simulation of each step of M on x is matched by advancing the cursor on the "clock" string.
  - Because M' stops at the moment the "clock" string is exhausted—even if $M(x)$ stops earlier, it is precise.
  - The time bound is therefore $O(|\, x \,| + f(|\, x \,|))$.

The Proof (concluded)

- If f is a space bound (sketch):
  - M' simulates M *on* the quasi-blanks of M_f's output string.[a]
  - The total space, not counting the input string, is $O(f(n))$.
  - But we still need a way to make sure there is no infinite loop even if M does not halt.[b]

---

[a]This is to make sure the space bound is precise.
[b]See the proof of Theorem 24 (p. 241).

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,

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

Important Complexity Classes (concluded)

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

Complements of Nondeterministic Classes

- Recall that the complement of L, or $\bar{L}$, is the language $\Sigma^* - L$.
  - SAT COMPLEMENT is the set of unsatisfiable boolean expressions.
- R, RE, and coRE are distinct (p. 161).
  - Again, coRE contains the complements of *languages* in RE, *not* languages that are not in RE.
- How about co$\mathcal{C}$ when $\mathcal{C}$ is a complexity class?

The Co-Classes

- For any complexity class $\mathcal{C}$, co$\mathcal{C}$ denotes the class

$$\{ L : \bar{L} \in \mathcal{C} \}.$$

- Clearly, if $\mathcal{C}$ is a *deterministic* time or space *complexity class*, then $\mathcal{C} = \text{co}\mathcal{C}$.
  - They are said to be **closed under complement**.
  - A deterministic TM deciding L can be converted to one that decides $\bar{L}$ within the same time or space bound by reversing the "yes" and "no" states.[a]
- Whether *non*deterministic classes for time are closed under complement is not known (see p. 113).

[a]See p. 158.

Comments

- As
$$\text{co}\mathcal{C} = \{\, L : \bar{L} \in \mathcal{C} \,\},$$
$L \in \mathcal{C}$ if and only if $\bar{L} \in \text{co}\mathcal{C}$.
- But it is *not* true that $L \in \mathcal{C}$ if and only if $L \notin \text{co}\mathcal{C}$.
  - co$\mathcal{C}$ is not defined as $\bar{\mathcal{C}}$.
- For example, suppose $\mathcal{C} = \{\{\, 2, 4, 6, 8, 10, \ldots \,\}, \ldots \}$.
- Then co$\mathcal{C} = \{\{\, 1, 3, 5, 7, 9, \ldots \,\}, \ldots \}$.
- But $\bar{\mathcal{C}} = 2^{\{\, 1,2,3,\ldots \,\}} - \{\{\, 2, 4, 6, 8, 10, \ldots \,\}, \ldots \}$.

The Quantified Halting Problem

- Let $f(n) \geq n$ be proper.
- Define

$$H_f \stackrel{\Delta}{=} \{ M; x : M \text{ accepts input } x \text{ after at most } f(|x|) \text{ steps} \},$$

where M is deterministic.

- Assume the input is binary as usual.

$H_f \in \mathsf{TIME}(f(n)^3)$

- For each input $M; x$, we simulate M on x with an alarm clock of length $f(|\,x\,|)$.
  - Use the single-string simulator (p. 85), the universal TM (p. 137), and the linear speedup theorem (p. 95).
  - Our simulator accepts $M; x$ if and only if M accepts x before the alarm clock runs out.
- From p. 92, the total running time is $O(\ell_M k_M^2 f(n)^2)$, where ℓ_M is the length to encode each symbol or state of M and k_M is M's number of strings.
- As $\ell_M k_M^2 = O(n)$, the running time is $O(f(n)^3)$, where the constant is independent of M.

$H_f \not\in \mathsf{TIME}(f(\lfloor n/2 \rfloor))$

- Suppose TM M_{H_f} decides H_f in time $f(\lfloor n/2 \rfloor)$.
- Consider machine:

$$
\begin{array}{ll}
D_f(M) & \{ \\
& \quad \textbf{if } M_{H_f}(M; M) = \text{“yes”} \\
& \quad \textbf{then } \text{“no”}; \\
& \quad \textbf{else } \text{“yes”}; \\
\} &
\end{array}
$$

The Proof (continued)

- $M_{H_f}(M; M)$ runs in time $f(\lfloor \frac{2n+1}{2} \rfloor) = f(n)$, where $n = |\, M \,|$.[a]

- By construction, $D_f(M)$ runs in the same amount of time as $M_{H_f}(M; M)$, i.e., $f(n)$, where $n = |\, M \,|$.

[a]Mr. Hsiao-Fei Liu (F92922019) and Mr. Hong-Lung Wang (F92922085) pointed out on October 6, 2004, that this estimation (and the text's Lemma 7.2) forgets to include the time to write down $M; M$.

The Proof (concluded)

- First, suppose $D_f(D_f) =$ "yes".
- This implies
$$D_f; D_f \notin H_f.$$
- Thus D_f does not accept D_f within time $f(\mid D_f \mid)$.
- But $D_f(D_f)$ stops in time $f(\mid D_f \mid)$ with an answer.
- Hence $D_f(D_f) =$ "no", a contradiction
- Similarly, $D_f(D_f) =$ "no" $\Rightarrow D_f(D_f) =$ "yes."

The Time Hierarchy Theorem

Theorem 18 *If $f(n) \geq n$ is proper, then*

$$\text{TIME}(f(n)) \subsetneq \text{TIME}(f(2n+1)^3).$$

- The quantified halting problem makes it so.

Corollary 19 $\text{P} \subsetneq \text{E}$.

- $\text{P} \subseteq \text{TIME}(2^n)$ because $\text{poly}(n) \leq 2^n$ for n large enough.
- But by Theorem 18,

$$\text{TIME}\left(2^n\right) \subsetneq \text{TIME}\left((2^{2n+1})^3\right) \subseteq \text{E}.$$

- So $\text{P} \subsetneq \text{E}$.

The Space Hierarchy Theorem

Theorem 20 (Hennie & Stearns, 1966) *If $f(n)$ is proper, then*

$$\mathrm{SPACE}(f(n)) \subsetneq \mathrm{SPACE}(f(n) \log f(n)).$$

Corollary 21 $\mathrm{L} \subsetneq \mathrm{PSPACE}$.

Nondeterministic Time Hierarchy Theorems

Theorem 22 (Cook, 1973) $\text{NTIME}(n^r) \subsetneq \text{NTIME}(n^s)$ *whenever* $1 \leq r < s$.

Theorem 23 (Seiferas, Fischer, & Meyer, 1978) *If* $T_1(n)$ *and* $T_2(n)$ *are proper, then*

$$\text{NTIME}(T_1(n)) \subsetneq \text{NTIME}(T_2(n))$$

whenever $T_1(n+1) = o(T_2(n))$.