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 $\{\neg, \lor, \land\}$ is functionally complete.
  - Every boolean expression can be turned into a CNF, which involves only $\neg$, $\lor$, and $\land$.
- The sets $\{\neg, \lor\}$ and $\{\neg, \land\}$ are functionally complete.
  - By the above result and de Morgan's laws.
- $\{$ NAND $\}$ and $\{$ NOR $\}$ are functionally complete.[a]

[a] Peirce (c. 1880) and 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."

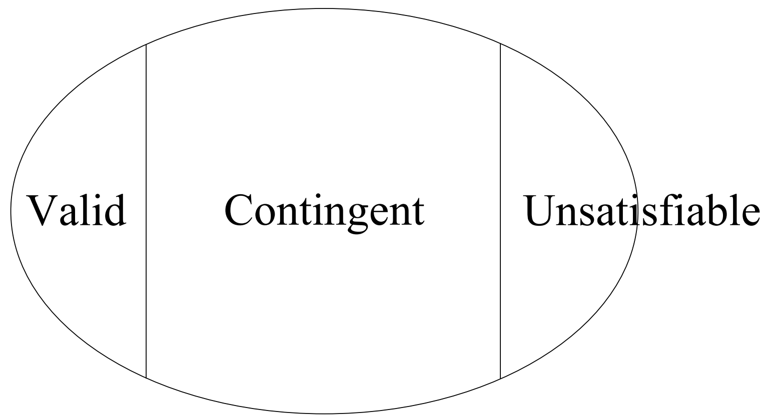
SATISFIABILITY (SAT)

- 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. 114).
- A most important problem in settling the "$P \stackrel{?}{=} NP$" problem (p. 299).

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`

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.
  - $\bigvee_{T \,\models\, \phi,\ \text{literal } y_i \text{ is true in ``row'' } T} (y_1 \land \cdots \land y_n)$.
    - $y_1 \land \cdots \land y_n$ is called the **minterm** over $\{ x_1, \ldots, x_n \}$ for T.[a]
  - The size[b] is $\leq n 2^n \leq 2^{2n}$.

---

[a]Similar to **programmable logic array**.

[b]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. 195).
- 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

$$\{\texttt{true}, \texttt{false}, \lor, \land, \neg, x_1, x_2, \ldots\}.$$

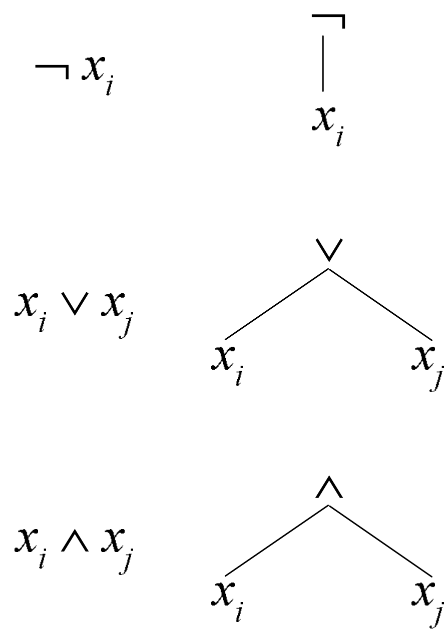
  - There are $n + 5$ sorts.

Boolean Circuits (concluded)

- Gates with a sort from $\{$ `true`, `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.

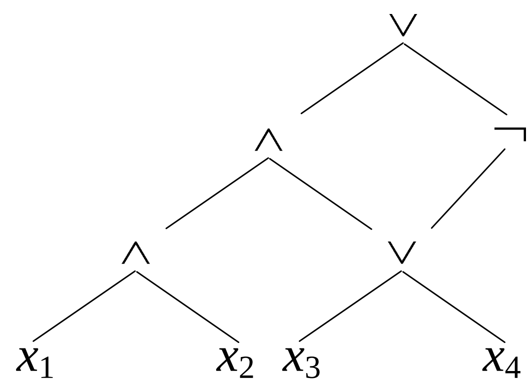
Boolean Circuits and Expressions

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



An Example

$$((x_1 \wedge x_2) \wedge (x_3 \vee x_4)) \vee (\neg (x_3 \vee x_4))$$



- 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. 185).
- So it suffices to prove that the number of boolean circuits with $2^n/(2n)$ or fewer gates is less than 2^{2^n}.

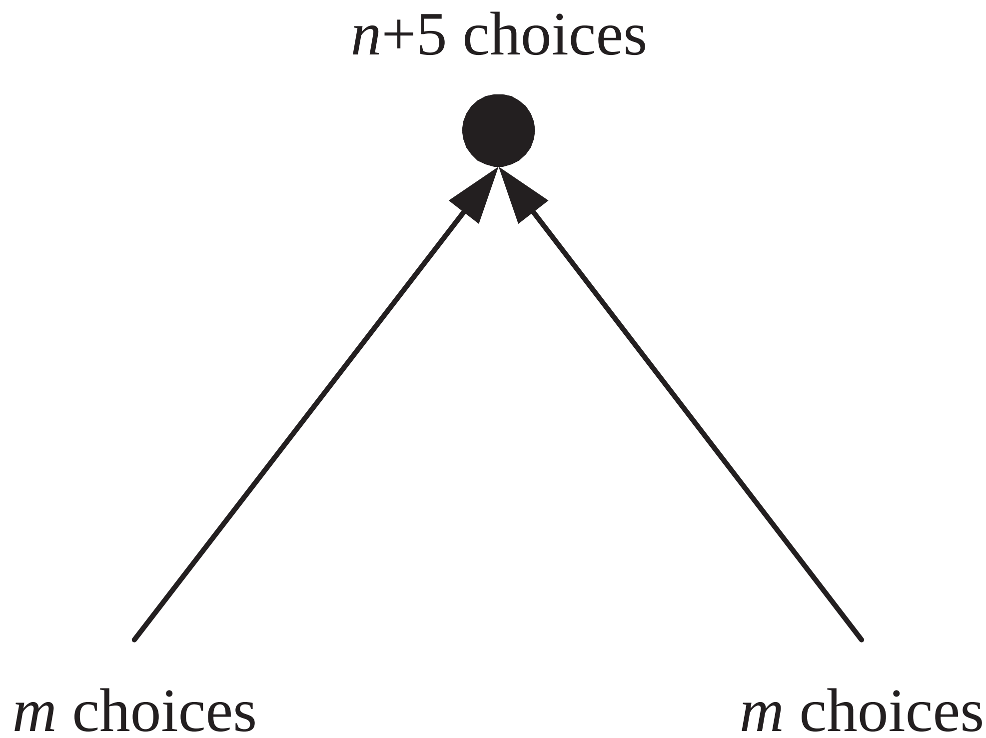
[a] Can be strengthened to "almost all boolean functions …"

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

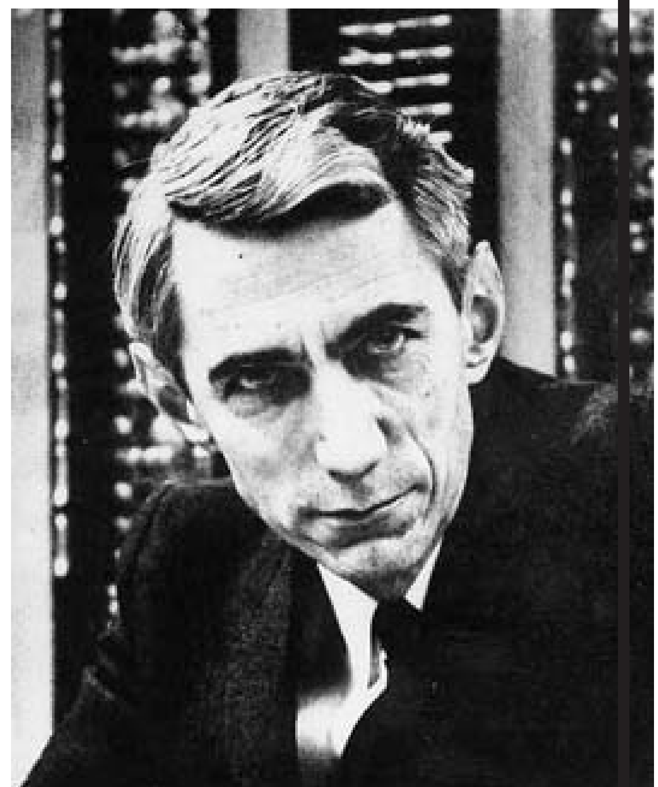
$$
\begin{aligned}
& m \log_2((n+5) \times m^2) \\
= \quad & 2^n \left(1 - \frac{\log_2 \frac{4n^2}{n+5}}{2n} \right) \\
< \quad & 2^n
\end{aligned}
$$

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. 187).
- 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 it 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)

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 15 (p. 205).

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 $f(g)$, we need to add $f(n) \geq n$.
[b]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 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."

---

[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.
  - 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.[a]

---

[a]See the proof of Theorem 22 on p. 223.

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)

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

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. 152).
  - 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 (p. 149).
- Whether nondeterministic classes for time are closed under complement is not known (see p. 106).

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

The Quantified Halting Problem

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

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

where M is deterministic.

- Assume the input is binary.

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

- For each input $M; x$, we simulate M on x with an alarm clock of length $f(|\, x \,|)$.
  - Use the single-string simulator (p. 80), the universal TM (p. 130), and the linear speedup theorem (p. 90).
  - Our simulator accepts $M; x$ if and only if M accepts x before the alarm clock runs out.
- From p. 87, 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.

$H_f \notin \mathsf{TIME}(f(\lfloor n/2 \rfloor))$

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

$$
\begin{aligned}
D_f(M) \quad \{ & \\
& \textbf{if } M_{H_f}(M;M) = \text{"yes"} \\
& \textbf{then } \text{"no"}; \\
& \textbf{else } \text{"yes"}; \\
\} &
\end{aligned}
$$

- D_f on input M runs in the same time as M_{H_f} on input $M;M$, i.e., in time $f(\lfloor \frac{2n+1}{2} \rfloor) = f(n)$, where $n = |\, M \,|$.[a]

[a] A student pointed out on October 6, 2004, that this estimation forgets to include the time to write down $M;M$.

The Proof (concluded)

- First,

$$\begin{aligned} & D_f(D_f) = \text{“yes”} \\ \Rightarrow\ & D_f; D_f \notin H_f \\ \Rightarrow\ & D_f \text{ does not accept } D_f \text{ within time } f(\mid D_f \mid) \\ \Rightarrow\ & D_f(D_f) \neq \text{“yes”} \end{aligned}$$

  a contradiction

- Similarly, $D_f(D_f) =$ “no” $\Rightarrow D_f(D_f) =$ “yes.”

The Time Hierarchy Theorem

Theorem 16 *If $f(n) \geq n$ is proper, then*

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

- The quantified halting problem makes it so.

Corollary 17 $\mathrm{P} \subsetneq \mathrm{E}$.

- $\mathrm{P} \subseteq \mathrm{TIME}(2^n)$ because $\mathrm{poly}(n) \leq 2^n$ for n large enough.
- But by Theorem 16,

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

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

The Space Hierarchy Theorem

Theorem 18 (Hennie and Stearns (1966)) *If $f(n)$ is proper, then*

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

Corollary 19 $\mathrm{L} \subsetneq \mathrm{PSPACE}$.

Nondeterministic Time Hierarchy Theorems

Theorem 20 (Cook (1973)) $\text{NTIME}(n^r) \subsetneq \text{NTIME}(n^s)$ *whenever* $1 \leq r < s$.

Theorem 21 (Seiferas, Fischer, and Meyer (1978)) *If* $T_1(n), T_2(n)$ *are proper, then*

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

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

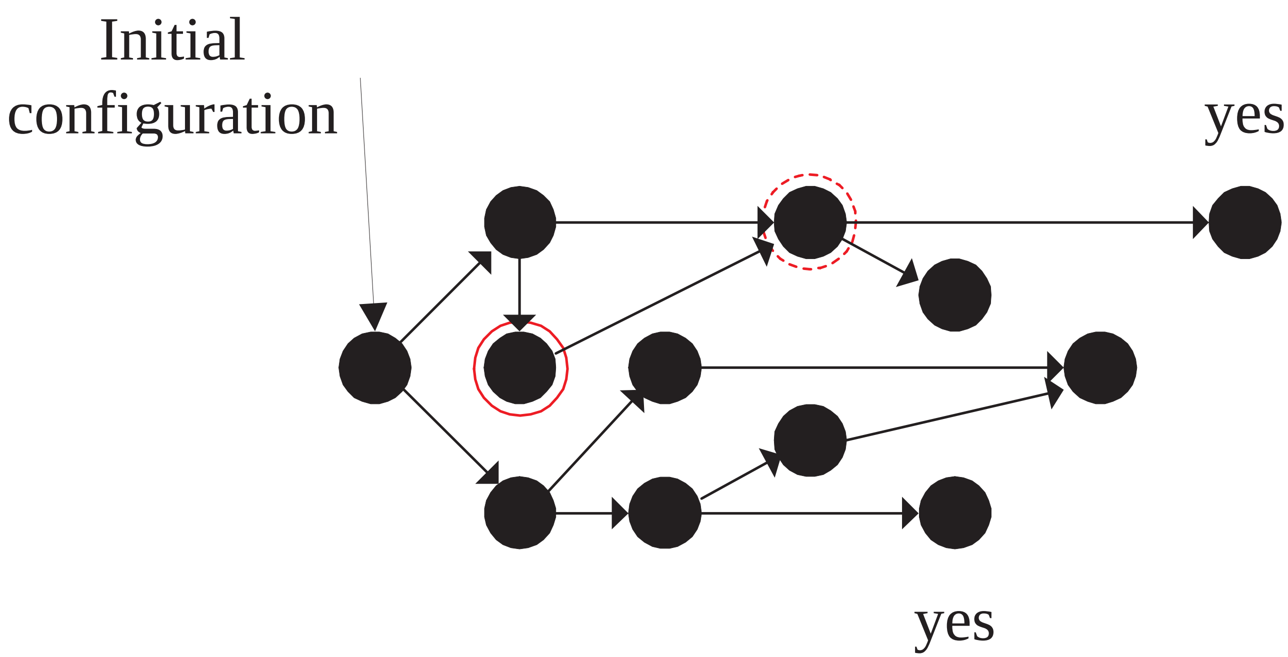
The Reachability Method

- The computation of a time-bounded TM can be represented by a directed graph.
- The TM's configurations constitute the nodes.
- There is a directed edge from node x to node y if x yields y in one step.
- The start node representing the initial configuration has zero in-degree.

The Reachability Method (concluded)

- When the TM is nondeterministic, a node may have an out-degree greater than one.
  - The graph is the same as the computation tree earlier except that identical configurations are merged into one node.
- So M accepts the input if and only if there is a path from the start node to a node with a "yes" state.
- It is the reachability problem.

Illustration of the Reachability Method



Relations between Complexity Classes

Theorem 22 *Suppose $f(n)$ is proper. Then*

1. $\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$,
   $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$.
2. $\mathrm{NTIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))$.
3. $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{TIME}(k^{\log n + f(n)})$.

- Proof of 2:
  - Explore the computation *tree* of the NTM for "yes."
  - Specifically, generate an $f(n)$-bit sequence denoting the nondeterministic choices over $f(n)$ steps.

Proof of Theorem 22(2)

- (continued)
  - Simulate the NTM based on the choices.
  - Recycle the space and repeat the above steps.
  - Halt with "yes" when a "yes" is encountered or "no" if the tree is exhausted.
  - Each path simulation consumes at most $O(f(n))$ space because it takes $O(f(n))$ time.
  - The total space is $O(f(n))$ because space is recycled.

Proof of Theorem 22(3)

- Let k-string NTM

$$M = (K, \Sigma, \Delta, s)$$

with input and output decide $L \in \text{NSPACE}(f(n))$.

- Use the reachability method on the configuration graph of M on input x of length n.
- A configuration is a $(2k+1)$-tuple

$$(q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k).$$

Proof of Theorem 22(3) (continued)

- We only care about

$$(q, i, w_2, u_2, \ldots, w_{k-1}, u_{k-1}),$$

where i is an integer between 0 and n for the position of the first cursor.

- The number of configurations is therefore at most

$$|\,K\,| \times (n+1) \times |\,\Sigma\,|^{2(k-2)f(n)} = O(c_1^{\log n + f(n)}) \tag{1}$$

for some c_1, which depends on M.

- Add edges to the configuration graph based on M's transition function.

Proof of Theorem 22(3) (concluded)

- $x \in L \Leftrightarrow$ there is a path in the configuration graph from the initial configuration to a configuration of the form ("yes", $i, \ldots$).[a]
- This is REACHABILITY on a graph with $O(c_1^{\log n + f(n)})$ nodes.
- It is in TIME$(c^{\log n + f(n)})$ for some c because REACHABILITY $\in$ TIME(n^j) for some j and

$$\left[c_1^{\log n + f(n)} \right]^j = (c_1^j)^{\log n + f(n)}.$$

---

[a]There may be many of them.

Space-Bounded Computation and Proper Functions

- In the definition of *space-bounded* computations earlier (p. 105), the TMs are not required to halt at all.
- When the space is bounded by a proper function f, computations can be assumed to halt:
  - Run the TM associated with f to produce a quasi-blank output of length $f(n)$ first.
  - The space-bounded computation must repeat a configuration if it runs for more than $c^{\log n + f(n)}$ steps for some c (p. 226).

Space-Bounded Computation and Proper Functions (concluded)

- (continued)
  - So an infinite loop occurs during simulation for a computation path longer than $c^{\log n + f(n)}$ steps.
  - Hence we only simulate up to $c^{\log n + f(n)}$ time steps per computation path.

A Grand Chain of Inclusions[a]

- It is an easy application of Theorem 22 (p. 223) that

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}.$$

- By Corollary 19 (p. 218), we know $\mathrm{L} \subsetneq \mathrm{PSPACE}$.
- So the chain must break somewhere between L and EXP.
- It is suspected that all four inclusions are proper.
- But there are no proofs yet.

[a] With input from Mr. Chin-Luei Chang (`R93922004`, `D95922007`) on October 22, 2004.

What Is Wrong with the Proof?[a]

- By Theorem 22(2) (p. 223),

$$\mathrm{NL} \subseteq \mathrm{TIME}\left(k^{O(\log n)}\right) \subseteq \mathrm{TIME}\left(n^{c_1}\right)$$

  for some $c_1 > 0$.

- By Theorem 16 (p. 217),

$$\mathrm{TIME}\left(n^{c_1}\right) \subsetneq \mathrm{TIME}\left(n^{c_2}\right) \subseteq \mathrm{P}$$

  for some $c_2 > c_1$.

- So

$$\mathrm{NL} \neq \mathrm{P}.$$

---

[a]Contributed by Mr. Yuan-Fu Shao (`R02922083`) on November 11, 2014.

What Is Wrong with the Proof? (concluded)

- Recall from p. 208 that $\text{TIME}(k^{O(\log n)})$ is a shorthand for

$$\bigcup_{j>0} \text{TIME}\left(j^{O(\log n)}\right).$$

- So the correct proof runs more like

$$\text{NL} \subseteq \bigcup_{j>0} \text{TIME}\left(j^{O(\log n)}\right) \subseteq \bigcup_{c>0} \text{TIME}\left(n^c\right) = \text{P}.$$

- And

$$\text{NL} \neq \text{P}$$

  no longer follows.

Nondeterministic and Deterministic Space

- By Theorem 6 (p. 111),

$$\text{NTIME}(f(n)) \subseteq \text{TIME}(c^{f(n)}),$$

  an exponential gap.

- There is no proof yet that the exponential gap is inherent.
- How about NSPACE vs. SPACE?
- Surprisingly, the relation is only quadratic—a polynomial—by Savitch's theorem.

Savitch's Theorem

Theorem 23 (Savitch (1970))

$$\text{REACHABILITY} \in \text{SPACE}(\log^2 n).$$

- Let $G(V, E)$ be a graph with n nodes.
- For $i \geq 0$, let
$$\text{PATH}(x, y, i)$$
mean there is a path from node x to node y of length at most 2^i.
- There is a path from x to y if and only if
$$\text{PATH}(x, y, \lceil \log n \rceil)$$
holds.

The Proof (continued)

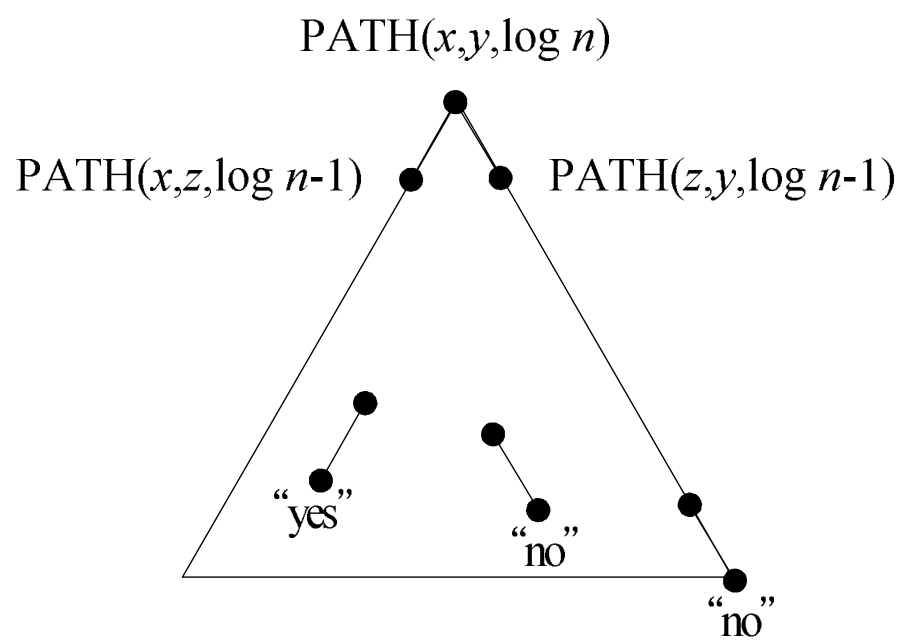
- For $i > 0$, PATH(x, y, i) if and only if there exists a z such that PATH$(x, z, i-1)$ and PATH$(z, y, i-1)$.
- For PATH$(x, y, 0)$, check the input graph or if $x = y$.
- Compute PATH$(x, y, \lceil \log n \rceil)$ with a depth-first search on a graph with nodes (x, y, z, i)s (see next page).[a]
- Like stacks in recursive calls, we keep only the current path of (x, y, i)s.
- The space requirement is proportional to the depth of the tree ($\lceil \log n \rceil$) times the size of the items stored at each node.

[a]Contributed by Mr. Chuan-Yao Tan on October 11, 2011.

The Proof (continued): Algorithm for PATH(x, y, i)

```
if i = 0 then
    if x = y or (x, y) ∈ E then
        return true;
    else
        return false;
    end if
else
    for z = 1, 2, ..., n do
        if PATH(x, z, i − 1) and PATH(z, y, i − 1) then
            return true;
        end if
    end for
    return false;
end if
```

The Proof (continued)



The Proof (concluded)

- Depth is $\lceil \log n \rceil$, and each node (x, y, z, i) needs space $O(\log n)$.
- The total space is $O(\log^2 n)$.

The Relation between Nondeterministic and Deterministic Space Is Only Quadratic

Corollary 24 *Let* $f(n) \geq \log n$ *be proper. Then*

$$\text{NSPACE}(f(n)) \subseteq \text{SPACE}(f^2(n)).$$

- Apply Savitch's proof to the configuration graph of the NTM on its input.
- From p. 226, the configuration graph has $O(c^{f(n)})$ nodes; hence each node takes space $O(f(n))$.
- But if we construct *explicitly* the whole graph before applying Savitch's theorem, we get $O(c^{f(n)})$ space!

The Proof (continued)

- The way out is *not* to generate the graph at all.
- Instead, keep the graph implicit.
- We checked node connectedness only when $i = 0$ on p. 236, by examining the input graph G.
- Now, given configurations x and y, we go over the Turing machine's program to determine if there is an instruction that can turn x into y in one step.[a]

[a]Thanks to a lively class discussion on October 15, 2003.

The Proof (concluded)

- The z variable in the algorithm on p. 236 simply runs through all possible valid configurations.
  - Let $z = 0, 1, \ldots, O(c^{f(n)})$.
  - Make sure z is a valid configuration before using it.[a]
- Each z has length $O(f(n))$.
- So each node needs space $O(f(n))$.
- The depth of the recursive call on p. 236 is $O(\log c^{f(n)})$, which is $O(f(n))$.
- The total space is therefore $O(f^2(n))$.

---

[a] Thanks to a lively class discussion on October 13, 2004.

Implications of Savitch's Theorem

- PSPACE = NPSPACE.
- Nondeterminism is less powerful with respect to space.
- Nondeterminism may be very powerful with respect to time as it is not known if P = NP.

Nondeterministic Space Is Closed under Complement

- Closure under complement is trivially true for deterministic complexity classes (p. 211).
- It is known that[a]

$$\text{coNSPACE}(f(n)) = \text{NSPACE}(f(n)). \qquad (2)$$

- So

$$\text{coNL} \quad = \quad \text{NL}.$$

- But it is not known whether coNP = NP.

[a] Szelepscényi (1987); Immerman (1988).