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. 120.

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.[c]
- We next prove that there are fewer than 2^{2^n} boolean circuits with up to $2^n/(2n)$ gates.

---

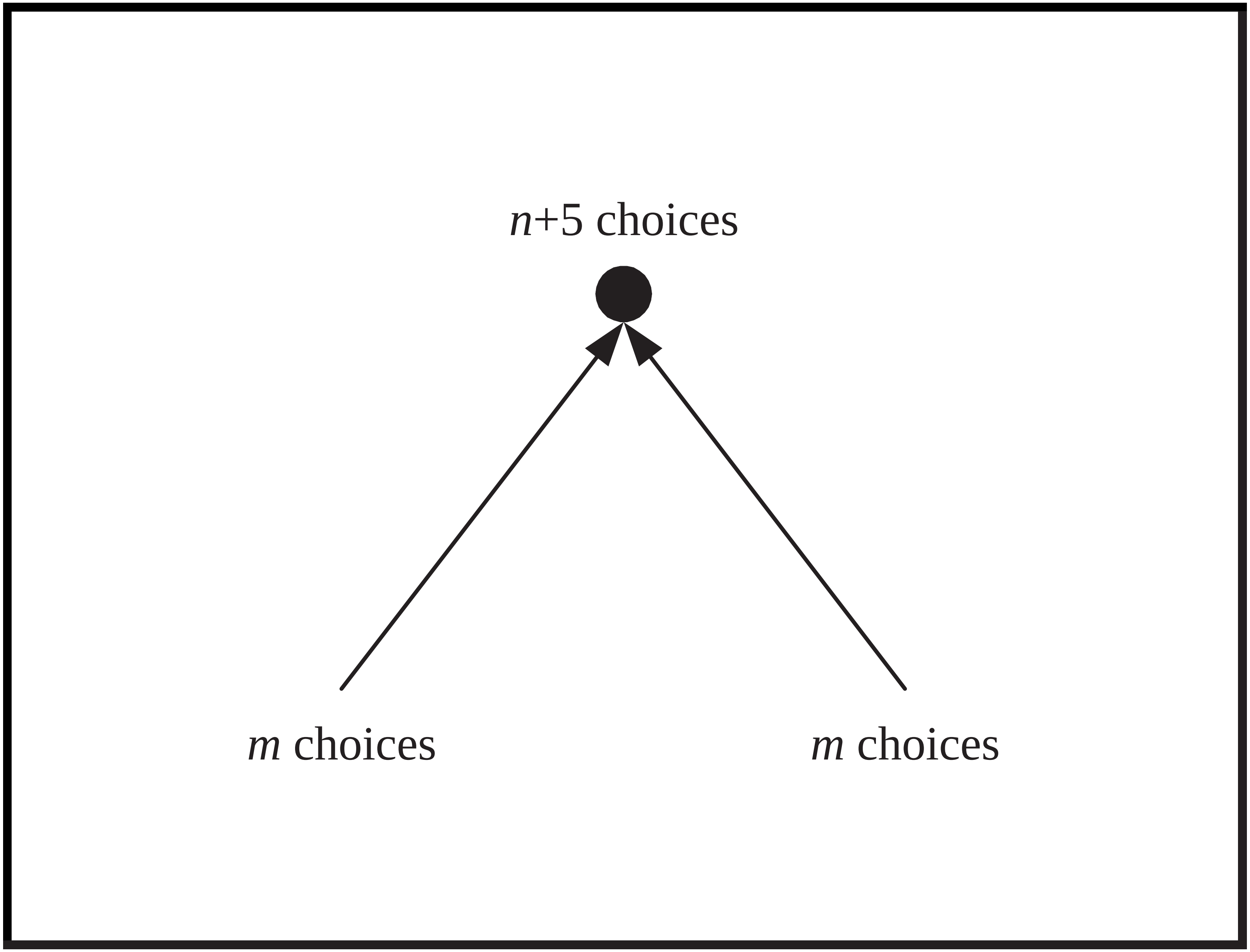
[a] Can be strengthened to "Almost all" (Lupanov, 1958).
[b] Riordan & Shannon (1942); Shannon (1949).
[c] Recall p. 209.

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

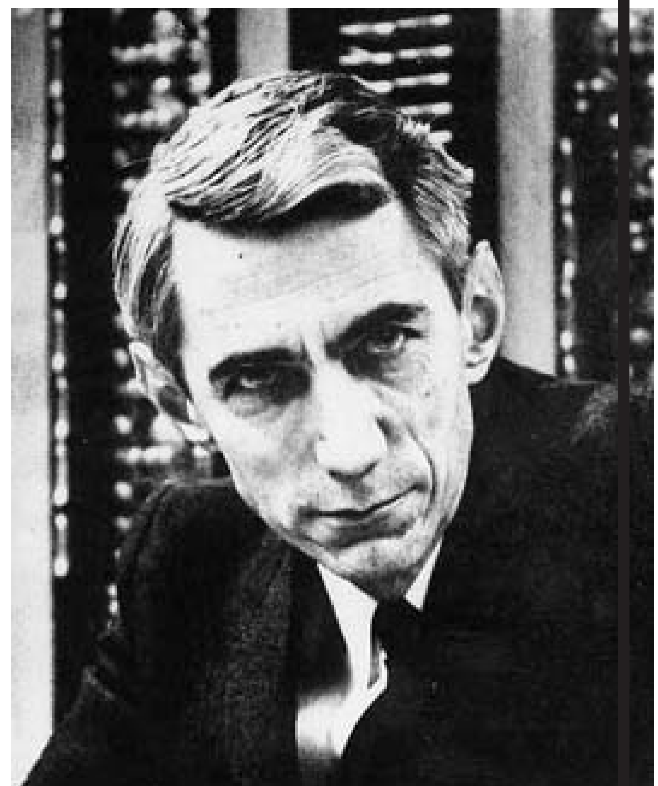
$$
\begin{aligned}
& m \log_2((n+5) \times m^2) \\
= \; & 2^n \left(1 - \frac{\log_2 \frac{4n^2}{n+5}}{2n} \right) \\
< \; & 2^n
\end{aligned}
$$

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. 211).
- 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. 229).

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 $g(n)$ at halting.[b]
    * 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]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] 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. 248).

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.[a]
  - Again, coRE contains the complements of *languages* in RE, *not* languages that are not in RE.

---

[a]Recall p. 164.

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**.
- Whether *non*deterministic classes for time are closed under complement is not known.

The Co-Classes (concluded)

- 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.

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

- For each input $M; x$, we simulate M on x with an alarm clock of length $f(|\,x\,|)$.
  - Use the single-string simulator (p. 87), the universal TM (p. 142), and the linear speedup theorem (p. 97).
  - Our simulator accepts $M; x$ if and only if M accepts x before the alarm clock runs out.
- From p. 94, the total running time is $O(\ell_M k_M^2 f^2(n))$, 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 $O(f^3(n))$, 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:

$D_f(M)$ {

  if $M_{H_f}(M; M) =$ "yes"

  then "no";

  else "yes";

}

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*

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

- The quantified halting problem makes it so.

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

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

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

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

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) $\mathrm{NTIME}(n^r) \subsetneq \mathrm{NTIME}(n^s)$ *whenever* $1 \leq r < s$.

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

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

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

The Reachability Method

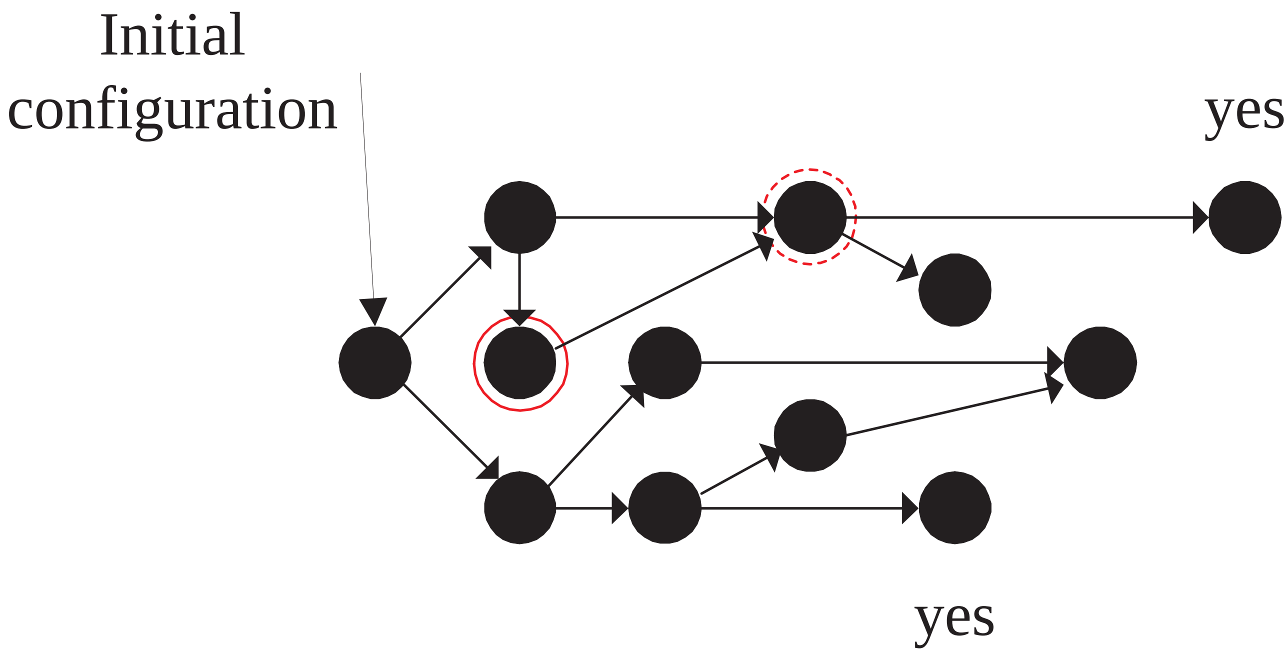
- The computation of a time-bounded TM can be represented by a directed graph.
- The TM's configurations constitute the nodes.
- There is a directed edge from node x to node y if x yields y in one step.
- The start node representing the initial configuration has zero in-degree.

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.
  - But identical configurations are merged into one node.[a]
- 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.

[a] So we end up with a graph not a tree.

Illustration of the Reachability Method



Relations between Complexity Classes

Theorem 24 *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 24(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.
  - Halt with "no" if the tree is exhausted without encountering a "yes."
  - 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 24(3)

- Let k-string NTM

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

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

- 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 24(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

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

  for some $c_1 > 1$, which depends on M.

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

Proof of Theorem 24(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 $c > 1$ 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. 116), 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 $c > 1$.[a]

---

[a] See Eq. (2) on p. 251.

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 need to 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 24 (p. 248) that

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP.$$

- By Corollary 21 (p. 243), we know $L \subsetneq PSPACE$.
- 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 (`B89902053`, `R93922004`, `D95922007`) on October 22, 2004.

What Is Wrong with the Proof?[a]

- By Theorem 24(2) (p. 248),

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

for some $c_1 > 0$.

- By Theorem 18 (p. 242),

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

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. 232 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. 132),

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

  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 25 (Savitch, 1970)

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

- 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)

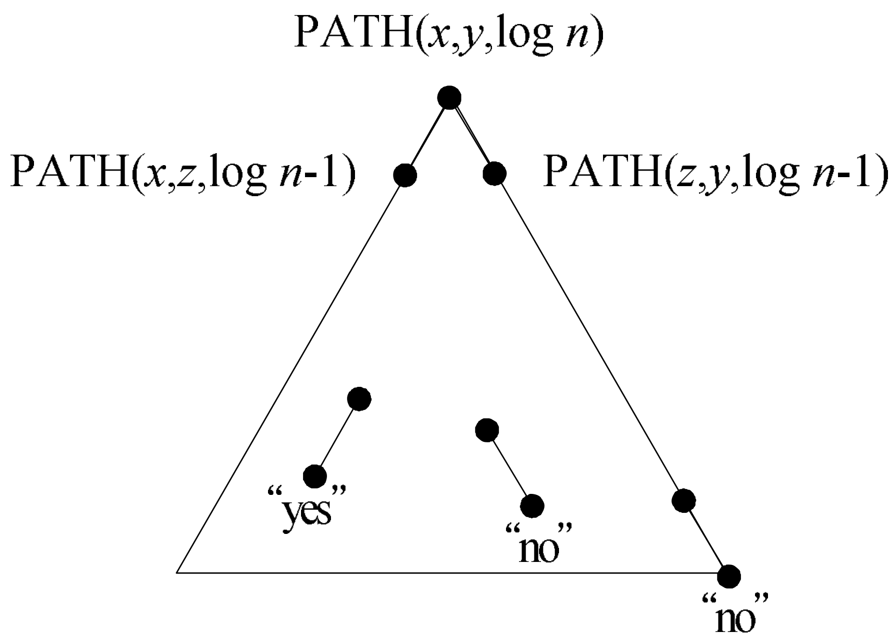
- For $i > 0$, PATH(x, y, i) if and only if there exists a z such that PATH$(x, z, i-1)$ and PATH$(z, y, i-1)$.
- For PATH$(x, y, 0)$, check the input graph or if $x = y$.
- Compute PATH$(x, y, \lceil \log n \rceil)$ with a depth-first search on a graph with nodes (x, y, i)s (see next page).[a]
- Like stacks in recursive calls, we keep only the current path's (x, y, i)s.

[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)

- 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.
- Depth is $\lceil \log n \rceil$, and each node (x, y, 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 26 *Let $f(n) \geq \log n$ be proper. Then*

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

- Apply Savitch's proof to the configuration graph of the NTM on its input.
- From p. 251, 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. 261, by examining the input graph G.
- Suppose we are given configurations x and y.
- Then 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]
- So connectivity is checked locally and on demand.

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

The Proof (continued)

- The z variable in the algorithm on p. 261 simply runs through all possible valid configurations.
  - Let $z = 0, 1, \ldots, O(c^{f(n)})$.
  - Make sure z is a valid configuration before proceeding with it.[a]
    * Adopt the same width for each symbol and state of the NTM and for the cursor position on the input string.[b]
  - If it is not, advance to the next z.

---

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

[b] Contributed by Mr. Jia-Ming Zheng (`R04922024`) on October 17, 2017.

The Proof (concluded)

- Each z has length $O(f(n))$.
- So each node needs space $O(f(n))$.
- The depth of the recursive call on p. 261 is $O(\log c^{f(n)})$, which is $O(f(n))$.
- The total space is therefore $O(f^2(n))$.

Implications of Savitch's Theorem

Corollary 27 *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.[a]
- It is known that[b]

$$\text{coNSPACE}(f(n)) = \text{NSPACE}(f(n)). \tag{3}$$

- So

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

- But it is not known whether coNP = NP.[c]

---

[a] Recall p. 235.

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

[c] If P = NP, then coNP = NP. Contributed by Mr. Yu-Ming Lu (`R06723032`, `D08922008`) on October 21, 2021.

Reductions and Completeness

It is unworthy of excellent men
to lose hours like slaves
in the labor of computation.
— Gottfried Wilhelm von Leibniz (1646–1716)

I thought perhaps you might be members of
that lowly section of the university
known as the Sheffield Scientific School.
F. Scott Fitzgerald (1920), "May Day"

Degrees of Difficulty

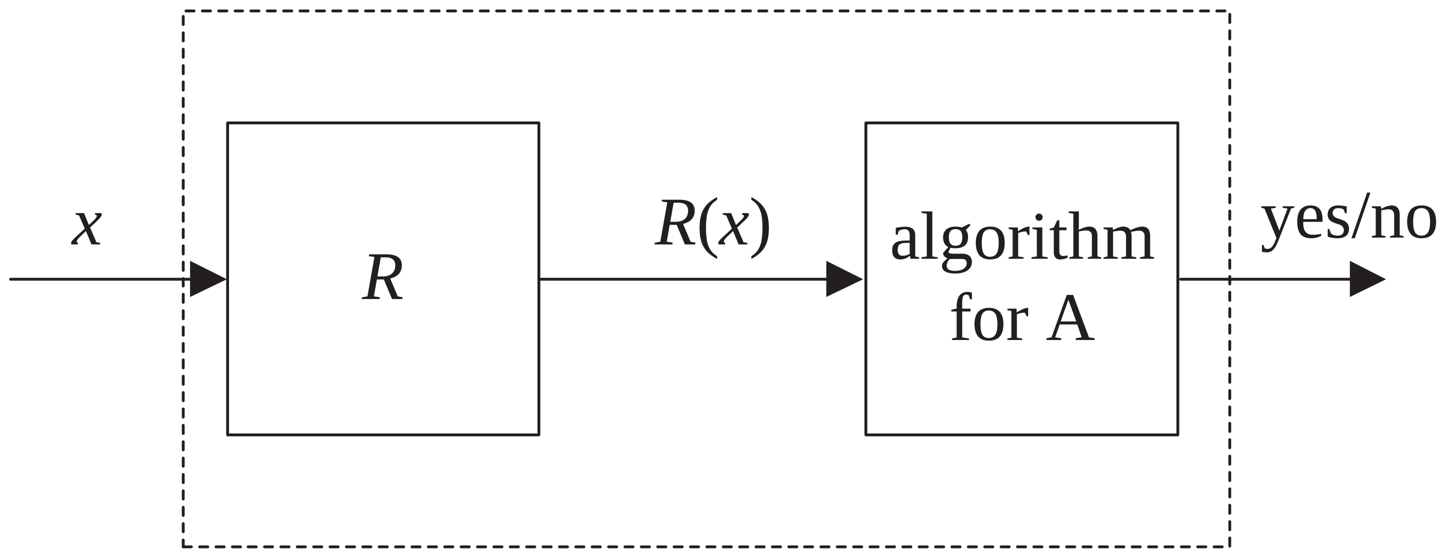
- When is a problem more difficult than another?
- B **reduces to** A if:
  - There is a transformation R which for every problem instance x of B yields a problem instance $R(x)$ of A.[a]
  - The answer to "$R(x) \in$ A?" is the same as the answer to "$x \in$ B?"
  - R is easy to compute.
- We say problem A is at least as hard as[b] problem B if B reduces to A.

---

[a]See also p. 156.

[b]Or simply "harder than" for brevity.

Reduction



Solving problem B by calling the algorithm for problem A *once* and *without* further processing its answer.[a]

[a]More general reductions are possible, such as the Turing (1939) reduction and the Cook (1971) reduction.

Degrees of Difficulty (concluded)

- This makes intuitive sense: If A is able to solve your problem B after only a little bit of work of R, then A must be at least as hard.
  - If A is easy to solve, it combined with R (which is also easy) would make B easy to solve, too.[a]
  - So if B is hard to solve, A must be hard, too!

[a]Thanks to a lively class discussion on October 13, 2009.

Comments[a]

- Suppose B reduces to A via a transformation R.[b]
- The input x is an instance of B.
- The output $R(x)$ is an instance of A.
- $R(x)$ may not span all possible instances of A.[c]
  - Some instances of A may never appear in R's range.
- But x must be an *arbitrary* instance for B.

---

[a] Contributed by Mr. Ming-Feng Tsai (`D92922003`) on October 29, 2003.

[b] Sometimes, we say "B can be reduced to A."

[c] $R(x)$ may not be onto; Mr. Alexandr Simak (`D98922040`) on October 13, 2009.

Comments (concluded)

- Usually, $R(x)$'s range A$'$ is a small subset of A.
- If A$'$ or a subset of A that contains A$'$ is an interesting problem in its own right, it will be given a name.[a]

[a] Contributed by Mr. Yu-Ming Lu (`R06723032`, `D08922008`) on November 18, 2021.

Is "Reduction" a Confusing Choice of Word?[a]

- If B reduces to A, doesn't that intuitively make A smaller and simpler?
- But our definition means the opposite.
- Our definition says in this case B is a special case of A.[b]
- Hence A is harder.

---

[a] Moore & Mertens (2011).

[b] See also p. 157.

Reduction between Languages

- Language L_1 is **reducible to** L_2 if there is a function R computable by a deterministic TM in space $O(\log n)$.
- Furthermore, for all inputs x, $x \in L_1$ if and only if $R(x) \in L_2$.
- R is said to be a (**Karp**) **reduction** from L_1 to L_2.

Reduction between Languages (concluded)

- Note that by Theorem 24 (p. 248), R runs in polynomial time.
  - In most cases, a polynomial-time R suffices for proofs.[a]
- Suppose R is a reduction from L_1 to L_2.
- Then solving "$R(x) \in L_2$?" is an algorithm for solving "$x \in L_1$?"[b]

[a] In fact, unless stated otherwise, we will only require that the reduction R run in polynomial time. It is often called a **polynomial-time many-one reduction**.

[b] Of course, it may not be the most efficient one.

A Paradox?

- Degree of difficulty is not defined in terms of *absolute* complexity.
- So a language $B \in \text{TIME}(n^{99})$ may be "easier" than a language $A \in \text{TIME}(n^3)$ if B reduces to A.
- But isn't this a contradiction if the best algorithm for B requires n^{99} steps?
- That is, how can a problem *requiring* n^{99} steps be reducible to a problem solvable in n^3 steps?

Paradox Resolved

- The "contradiction" is the result of flawed logic.
- Suppose we solve the problem "$x \in$ B?" via "$R(x) \in$ A?"
- We must consider the time spent by $R(x)$ and its length $|\, R(x) \,|$:
  - It is $R(x)$ — not x — that is solved by A.