Some Boolean Functions Need Exponential Circuits[a]

Theorem 15 (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 (see 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}.

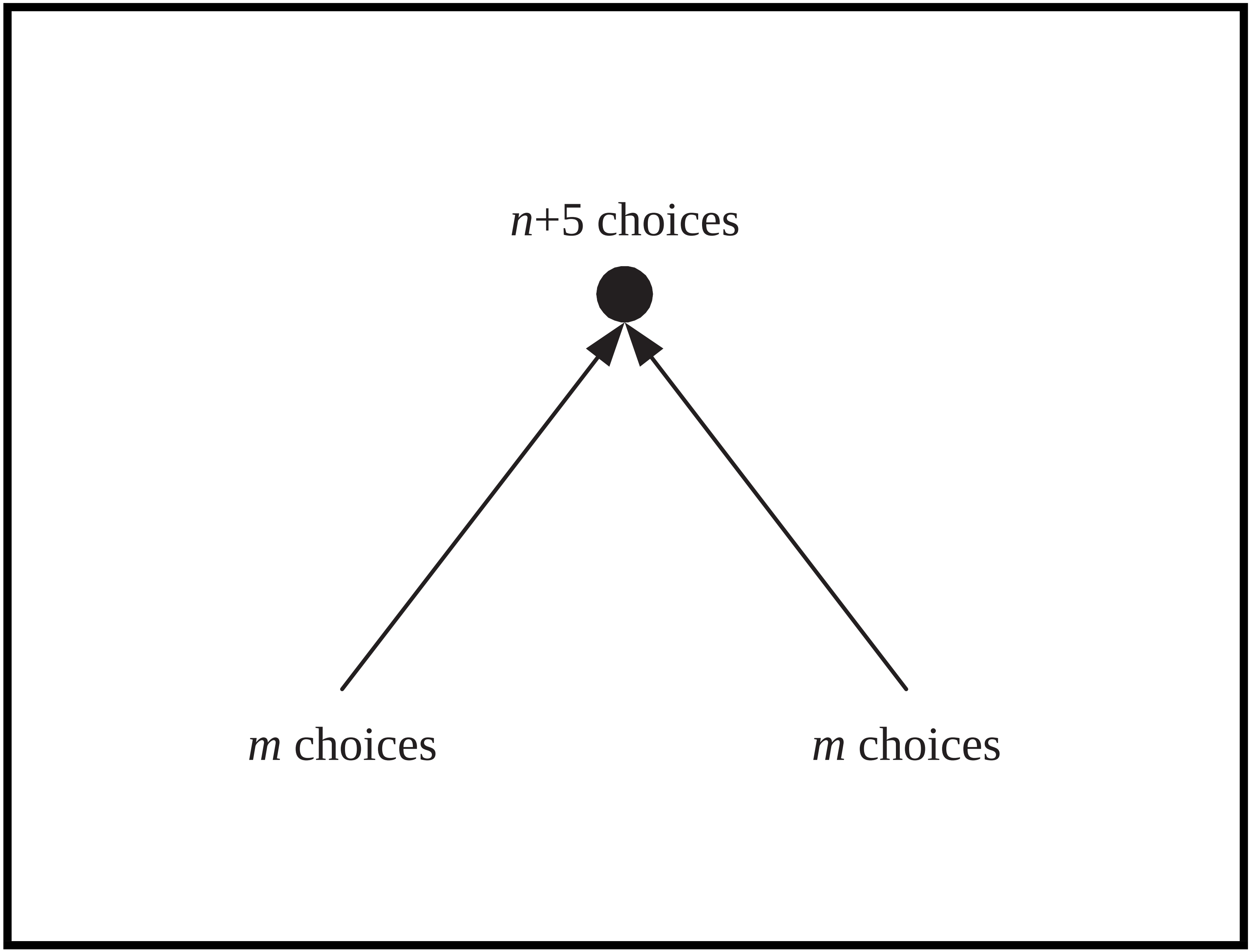
[a] Can be strengthened to "almost all boolean functions ..."

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

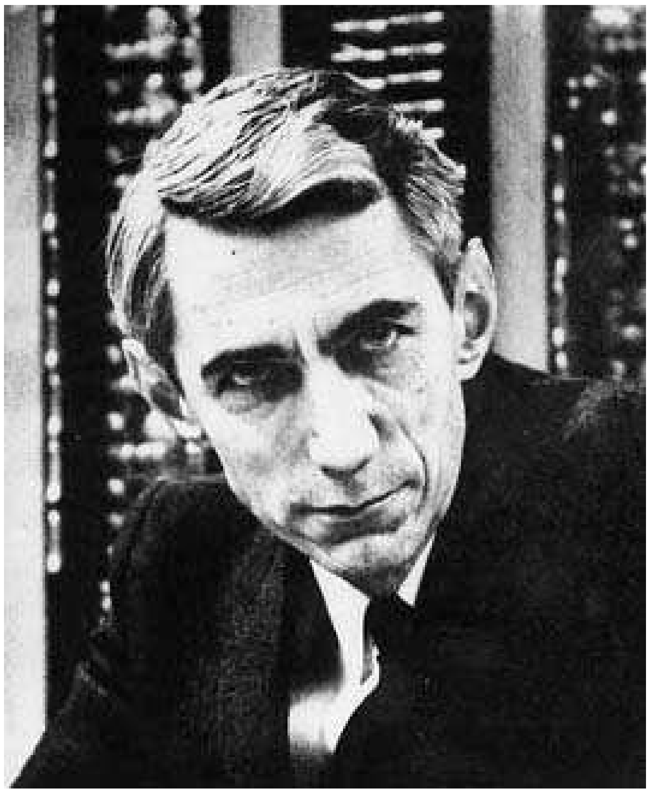
$$\begin{aligned} & m \log_2((n+5) \times m^2) \\ = \quad & 2^n \left(1 - \frac{\log_2 \frac{4n^2}{n+5}}{2n} \right) \\ < \quad & 2^n \end{aligned}$$

for $n \geq 2$.



n+5 choices
m choices
m choices

Claude Elwood Shannon (1916–2001)



Comments

- The lower bound is rather tight because an upper bound is $n2^n$ (p. 160).
- In the proof, we counted the number of circuits.
- Some circuits may not be valid at all.
- Others may compute the same boolean functions.
- Both are fine because we only need an upper bound.
- We do not need to consider the outdoing edges because they have been counted in the 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 basically bounded by $f(n)$.

[a]This point will become clear in Proposition 16 on 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).

Space-Bounded Computation and Proper Functions

- In the definition of *space-bounded* computations, 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 an output of length $f(n)$ first.
  - The space-bound computation must repeat a configuration if it runs for more than $c^{n+f(n)}$ steps for some c (p. 195).
  - So we can count steps to prevent infinite loops.

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.
    * 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 16 *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.

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

- From p. 136, we know R, RE, and coRE are distinct.
  - coRE contains the complements of languages in RE, *not* the languages not in RE.
- Recall that the **complement** of L, denoted by $\bar{L}$, is the language $\Sigma^* - L$.
  - SAT COMPLEMENT is the set of unsatisfiable boolean expressions.
  - HAMILTONIAN PATH COMPLEMENT is the set of graphs without a Hamiltonian path.

The Co-Classes

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

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

- 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.
- Whether nondeterministic classes for time are closed under complement is not known (p. 85).

Comments

- Then co$\mathcal{C}$ is the class
$$\{\bar{L} : L \in \mathcal{C}\}.$$
  - So $L \in \mathcal{C}$ 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

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

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. 65), the universal TM (p. 121), and the linear speedup theorem (p. 71).
  - Our simulator accepts $M; x$ if and only if M accepts x before the alarm clock runs out.
- From p. 70, 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 $D_f(M)$:

$$\textbf{if } M_{H_f}(M;M) = \text{“yes” } \textbf{then} \text{ “no” } \textbf{else} \text{ “yes”}$$

- 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 omits the time to write down $M;M$.

The Proof (concluded)

- First,

$$
\begin{aligned}
& D_f(D_f) = \text{"yes"} \\
\Rightarrow \quad & D_f; D_f \notin H_f \\
\Rightarrow \quad & D_f \text{ does not accept } D_f \text{ within time } f(\mid D_f \mid) \\
\Rightarrow \quad & D_f(D_f) = \text{"no"}
\end{aligned}
$$

  a contradiction

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

The Time Hierarchy Theorem

Theorem 17 *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 18 P $\subsetneq$ EXP.

- P $\subseteq$ TIME(2^n) because poly(n) $\leq 2^n$ for n large enough.
- But by Theorem 17,

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

- So P $\subsetneq$ EXP.

The Space Hierarchy Theorem

Theorem 19 (Hennie and Stearns (1966)) *If $f(n)$ is proper, then*

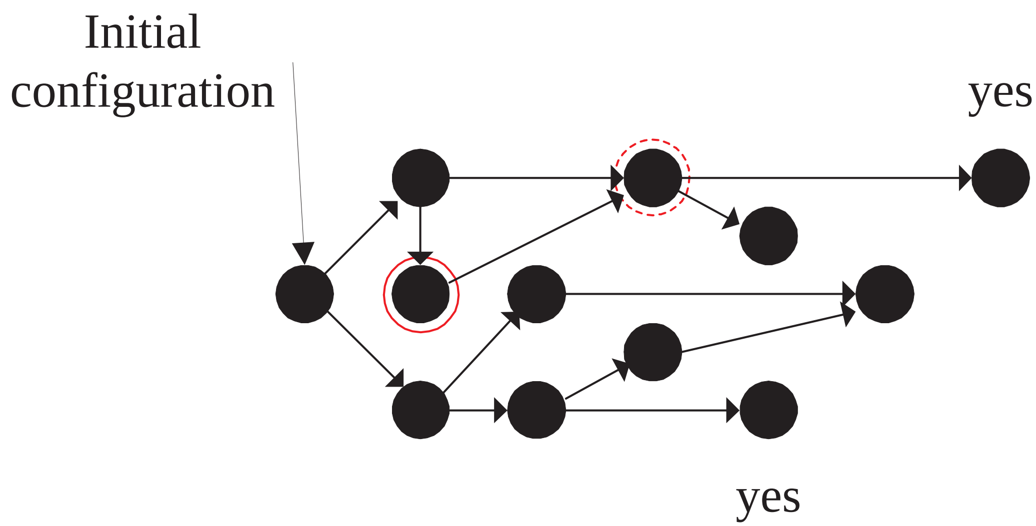
$$\mathrm{SPACE}(f(n)) \subsetneq \mathrm{SPACE}(f(n)\log f(n)).$$

Corollary 20 $\mathrm{L} \subsetneq \mathrm{PSPACE}$.

The Reachability Method

- The computation of a time-bounded TM can be represented by directional transitions between configurations.
- The reachability method constructs a directed graph with all the TM configurations as its nodes and edges connecting two nodes if one yields the other.
- The start node representing the initial configuration has zero in degree.
- When the TM is nondeterministic, a node may have an out degree greater than one.

Illustration of the Reachability Method



Relations between Complexity Classes

Theorem 21 *Suppose $f(n)$ is proper. Then*

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

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

Proof of Theorem 21(2)

- (continued)
  - Simulate the NTM based on the choices.
  - Recycle the space and then repeat the above steps until a "yes" is encountered or 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 $O(f(n))$ as space is recycled.

Proof of Theorem 21(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 21(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

$$|K| \times (n+1) \times |\Sigma|^{(2k-4)f(n)} = O(c_1^{\log n + f(n)}) \tag{2}$$

  for some c_1, which depends on M.

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

Proof of Theorem 21(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", $i, \ldots$) [there may be many of them].
- The problem is therefore that of 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 is in TIME(n^k) for some k and

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

The Grand Chain of Inclusions

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

- By Corollary 20 (p. 189), we know $L \subsetneq PSPACE$.
- The chain must break somewhere between L and PSPACE.
- It is suspected that all four inclusions are proper.
- But there are no proofs yet.[a]

[a]Carl Friedrich Gauss (1777–1855), "I could easily lay down a multitude of such propositions, which one could neither prove nor dispose of."

Nondeterministic Space and Deterministic Space

- By Theorem 5 (p. 95),

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

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

Savitch's Theorem

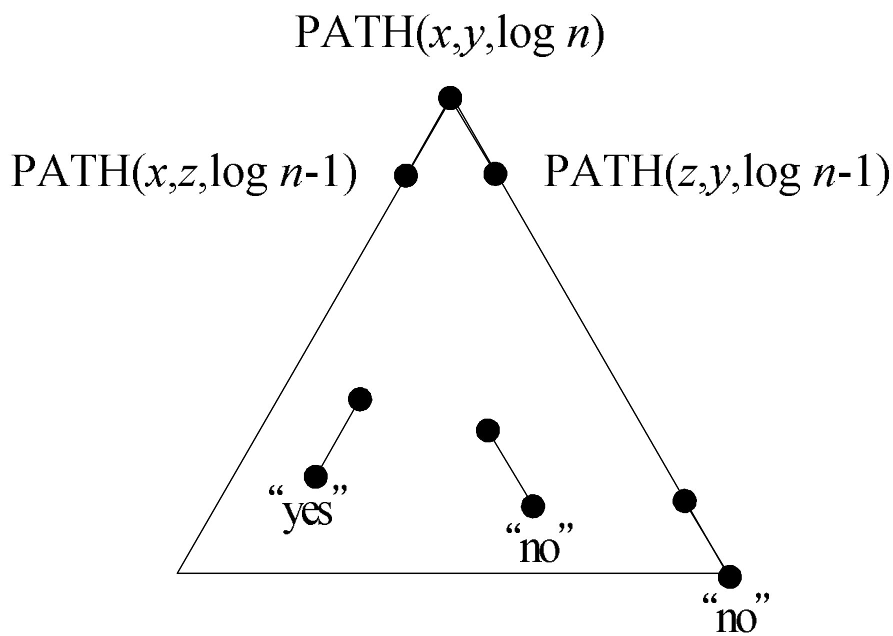
Theorem 22 (Savitch (1970))

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

- Let G 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).
- 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, $\lceil \log n \rceil$.



- 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 Proof (concluded): Algorithm for PATH(x, y, i)

```
if i = 0 then
    if x = y or (x, y) ∈ G 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 Relation between Nondeterministic Space and Deterministic Space Only Quadratic

Corollary 23 *Let $f(n) \geq \log n$ be proper. Then*

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

- Apply Savitch's theorem to the configuration graph of the NTM on the input.
- From p. 195, 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 check for connectedness only when $i = 0$, by examining the input string.
- There, 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. 202 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 in the recursive calls.[a]
- Each z has length $O(f(n))$ by Eq. (2) on p. 195.

[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. 182).
- It is known that[a]

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

- So

$$\begin{aligned} \text{coNL} &= \text{NL}, \\ \text{coNPSPACE} &= \text{NPSPACE}. \end{aligned}$$

- But there are still no hints of coNP = NP.

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

Reductions and Completeness

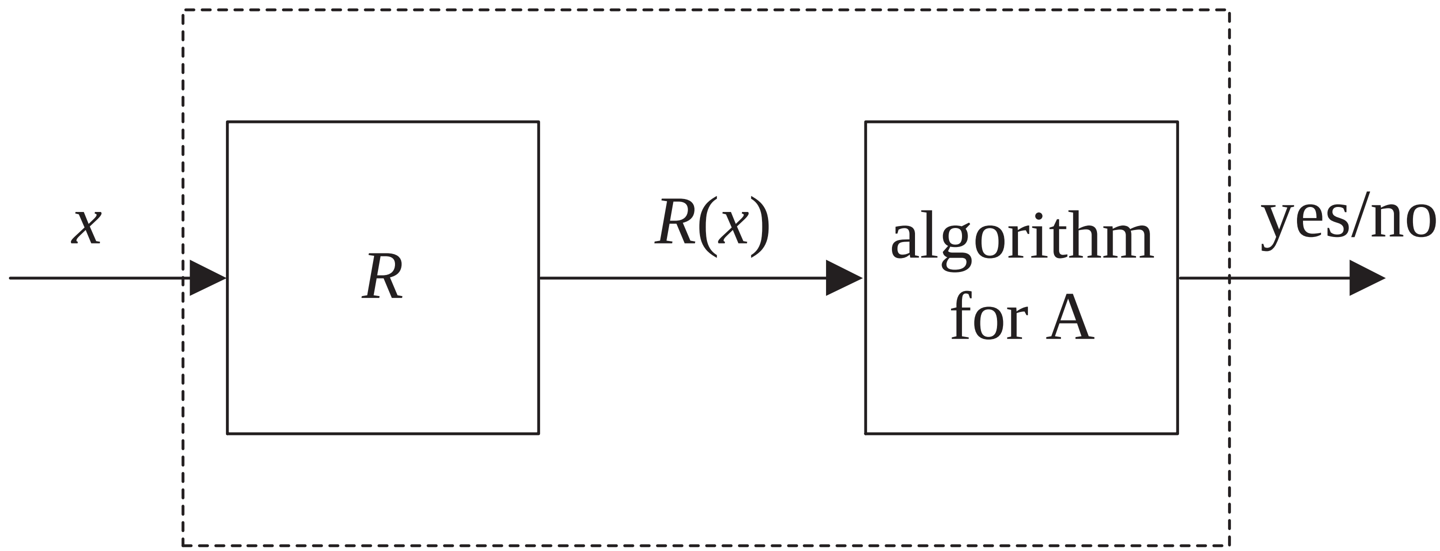
Degrees of Difficulty

- When is a problem more difficult than another?
- B **reduces to** A if there is a transformation R which for every input x of B yields an equivalent input $R(x)$ of A.
  - The answer to x for B is the same as the answer to $R(x)$ for A.
  - There must be restrictions on the complexity of computing R.
  - Otherwise, $R(x)$ might as well solve B.
    - E.g., $R(x) =$ "yes" if and only if $x \in$ B!

Degrees of Difficulty (concluded)

- Problem A is at least as hard as problem B if B reduces to A.
- This makes intuitive sense: If A is able to solve your problem B, then A must be at least as hard.

Reduction



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

Comments[a]

- Suppose B reduces to A via a transformation R.
- 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.
- So some instances of A may never appear in the reduction.

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

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.
- Note that by Theorem 21 (p. 192), R runs in polynomial time.
- Suppose R is a reduction from L_1 to L_2.
- Then solving "$R(x) \in L_2$" is an algorithm for solving "$x \in L_1$."

A Paradox?

- Degree of difficulty is not defined in terms of *absolute* complexity.
- So a language B $\in$ TIME(n^{99}) may be "easier" than a language A $\in$ TIME(n^3).
- This happens when B is reducible to A.
- But isn't this a contradiction when B $\notin$ TIME(n^{98})?
- That is, how can a problem *requiring* n^{33} time be reducible to a problem solvable in n^3 time?

A Paradox? (concluded)

- The so-called contradiction does not hold.
- When we solve the problem "$x \in$ B?" with "$R(x) \in$ A?", we must consider the time spent by $R(x)$ and its length $|\,R(x)\,|$.
- If $|\,R(x)\,| = \Omega(n^{33})$, then the time of answering "$R(x) \in$ A?" becomes $\Omega((n^{33})^3) = \Omega(n^{99})$.
- Suppose, on the other hand, that $|\,R(x)\,| = o(n^{33})$.
- Then $R(x)$ must run in time $\Omega(n^{99})$.
- In either case, there is no contradiction.

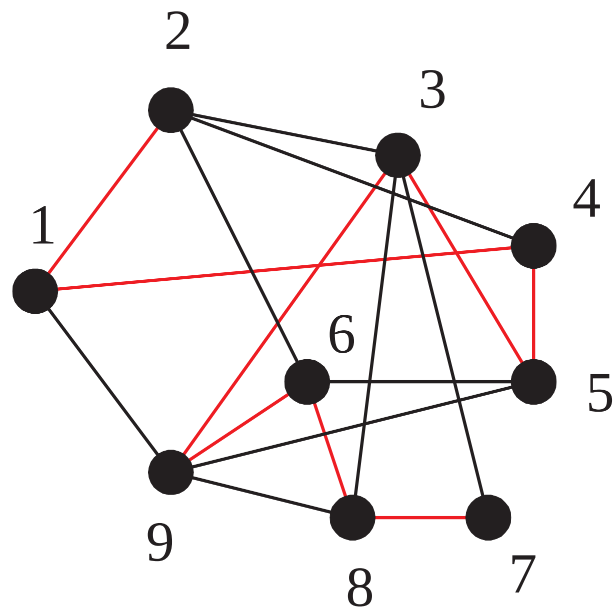
HAMILTONIAN PATH

- A **Hamiltonian path** of a graph is a path that visits every node of the graph exactly once.
- Suppose graph G has n nodes: $1, 2, \ldots, n$.
- A Hamiltonian path can be expressed as a permutation π of $\{ 1, 2, \ldots, n \}$ such that
  - $\pi(i) = j$ means the ith position is occupied by node j.
  - $(\pi(i), \pi(i+1)) \in G$ for $i = 1, 2, \ldots, n-1$.
- HAMILTONIAN PATH asks if a graph has a Hamiltonian path.

Reduction of HAMILTONIAN PATH to SAT

- Given a graph G, we shall construct a CNF $R(G)$ such that $R(G)$ is satisfiable if and only if G has a Hamiltonian path.
- $R(G)$ has n^2 boolean variables x_{ij}, $1 \leq i, j \leq n$.
- x_{ij} means

  the ith position in the Hamiltonian path is occupied by node j.



$$x_{12} = x_{21} = x_{34} = x_{45} = x_{53} = x_{69} = x_{76} = x_{88} = x_{97} = 1.$$

The Clauses of $R(G)$ and Their Intended Meanings

1. Each node j must appear in the path.
   - $x_{1j} \lor x_{2j} \lor \cdots \lor x_{nj}$ for each j.
2. No node j appears twice in the path.
   - $\neg x_{ij} \lor \neg x_{kj}$ for all i, j, k with $i \neq k$.
3. Every position i on the path must be occupied.
   - $x_{i1} \lor x_{i2} \lor \cdots \lor x_{in}$ for each i.
4. No two nodes j and k occupy the same position in the path.
   - $\neg x_{ij} \lor \neg x_{ik}$ for all i, j, k with $j \neq k$.
5. Nonadjacent nodes i and j cannot be adjacent in the path.
   - $\neg x_{ki} \lor \neg x_{k+1,j}$ for all $(i, j) \notin G$ and $k = 1, 2, \ldots, n-1$.

The Proof

- $R(G)$ contains $O(n^3)$ clauses.
- $R(G)$ can be computed efficiently (simple exercise).
- Suppose $T \models R(G)$.
- From Clauses of 1 and 2, for each node j there is a unique position i such that $T \models x_{ij}$.
- From Clauses of 3 and 4, for each position i there is a unique node j such that $T \models x_{ij}$.
- So there is a permutation π of the nodes such that $\pi(i) = j$ if and only if $T \models x_{ij}$.

The Proof (concluded)

- Clauses of 5 furthermore guarantees that $(\pi(1), \pi(2), \ldots, \pi(n))$ is a Hamiltonian path.
- Conversely, suppose G has a Hamiltonian path
$$(\pi(1), \pi(2), \ldots, \pi(n)),$$
where π is a permutation.
- Clearly, the truth assignment
$$T(x_{ij}) = \texttt{true} \text{ if and only if } \pi(i) = j$$
satisfies all clauses of $R(G)$.

A Comment[a]

- An answer to "Is $R(G)$ satisfiable?" does answer "Is G Hamiltonian?"
- But a positive answer does not give a Hamiltonian path for G.
  - Providing witness is not a requirement of reduction.
- A positive answer to "Is $R(G)$ satisfiable?" plus a satisfying truth assignment does provide us with a Hamiltonian path for G.

[a] Contributed by Ms. Amy Liu (`J94922016`) on May 29, 2006.