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 (p. 176).
- So it suffices to prove that the number of boolean circuits with $2^n/(2n)$ or fewer gates is less than 2^{2^n}.

---

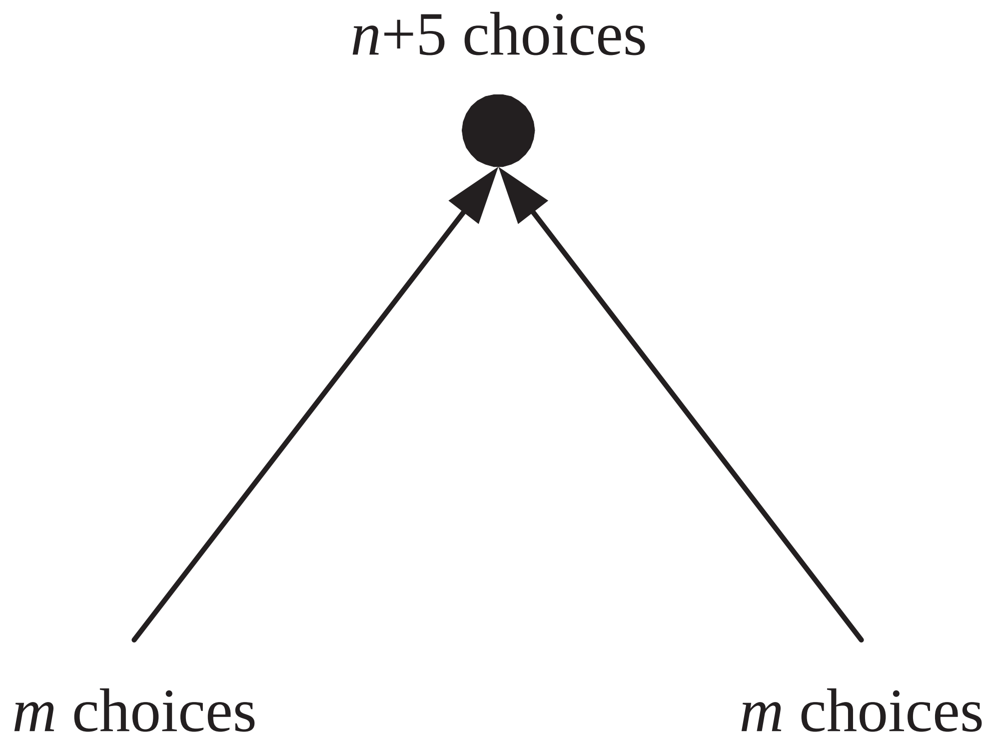
[a] Can be strengthened to "almost all boolean functions ..."

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

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

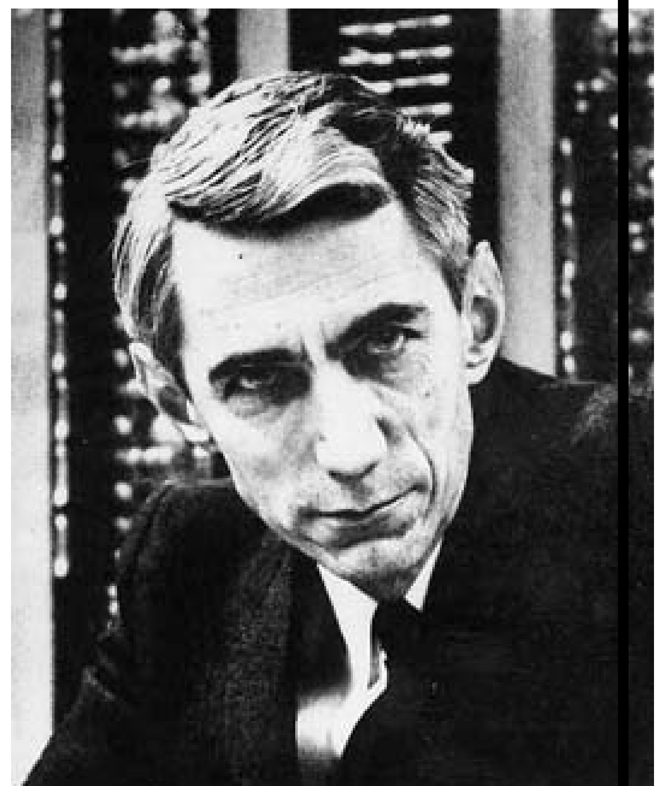
for $n \geq 2$.



n+5 choices
m choices
m choices

Claude Elwood Shannon (1916–2001)

Howard Gardner, "[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. 178).
- The proof counted the number of circuits.
  - Some circuits may not be valid at all.
  - Different circuits may also compute the same function.
- Both are fine because we only need an upper bound on the number of circuits.
- We do not need to consider the outdoing edges because they have been counted as incoming edges.

Relations between Complexity Classes

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 $M_f(x) = \sqcap^{f(\mid x \mid)}$ for any x.[a]
  - M_f halts after $O(\mid x \mid + f(\mid x \mid))$ steps.
  - M_f uses $O(f(\mid x \mid))$ space besides its input x.
- M_f's behavior depends only on $\mid x \mid$ 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 16 (p. 196).

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 "the 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 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."
- $M'(x)$ halts when and only when the alarm clock runs out—even if M halts earlier.

[a] It can be deterministic or nondeterministic.

The Proof (continued)

- 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.
  - M' stops at the moment the "clock" string is exhausted—even if $M(x)$ stops before that time.
  - So it is precise.
  - The time bound is therefore $O(|x| + f(|x|))$.

The Proof (concluded)

- If f is a space bound:
  - M' simulates M *on* the quasi-blanks of M_f's output string.
  - As before, M' stops at the moment the "clock" string is exhausted—even if $M(x)$ stops before that time.
  - So it is again precise.
  - The total space, not counting the input string, is $O(f(n))$.

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

- R, RE, and coRE are distinct (p. 150).
  - 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.

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. 147).
- Whether nondeterministic classes for time are closed under complement is not known (p. 92).

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 $\mathcal{C} = \{\{2, 4, 6, 8, 10, \ldots\}\}$.

- 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. 66), the universal TM (p. 132), and the linear speedup theorem (p. 75).
  - Our simulator accepts $M; x$ if and only if M accepts x before the alarm clock runs out.
- From p. 73, 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 $D_f(M)$:

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

  – “This sentence is false.”

- 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,

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

  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 $\subsetneq$ E.

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

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

- So P $\subsetneq$ E.

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

Nondeterministic Time Hierarchy Theorems

Theorem 21 (Cook (1973)) $\text{NTIME}(n^r) \subsetneq \text{NTIME}(n^s)$ *whenever* $1 \leq r < s$.

Theorem 22 (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 $T_1(n+1) = o(T_2(n))$.

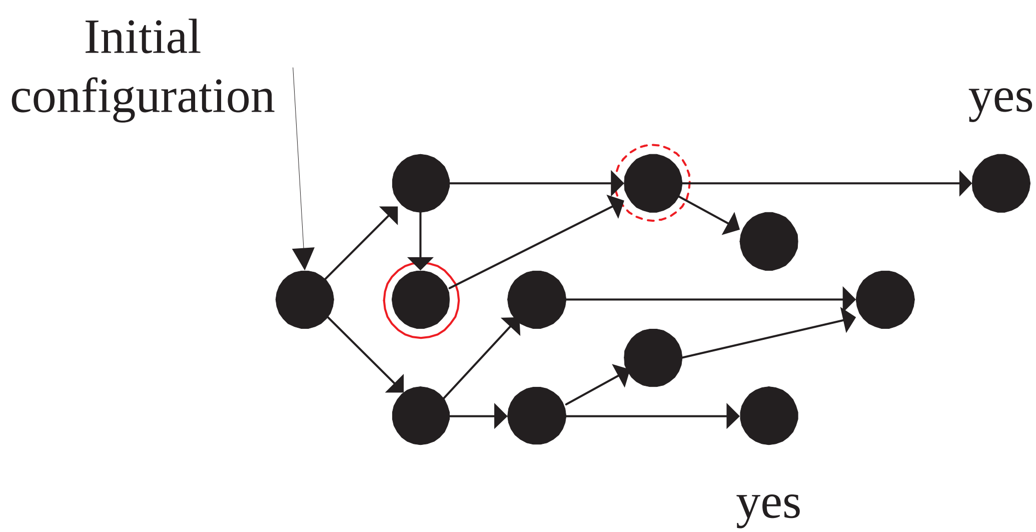
The Reachability Method

- The computation of a time-bounded TM can be represented by a directed graph.
- The TM's configurations constitute the nodes.
- Two nodes are connected by a directed edge if one yields the other 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 configuration nodes 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 23 *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 an $f(n)$-bit sequence denoting the nondeterministic choices over $f(n)$ steps.

Proof of Theorem 23(2)

- (continued)
  - Simulate the NTM based on the choices.
  - Recycle the space and 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 $O(f(n))$ because space is recycled.

Proof of Theorem 23(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 23(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

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

for some c_1, which depends on M.

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

Proof of Theorem 23(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. 89), 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 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. 217).
  - So we can prevent infinite loops during simulation by pruning any path longer than $c^{\log n + f(n)}$.

A Grand Chain of Inclusions[a]

- It is an easy application of Theorem 23 (p. 214) that

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

- By Corollary 20 (p. 209), 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 (`R93922004`, `D95922007`) on October 22, 2004.

Nondeterministic Space and Deterministic Space

- By Theorem 4 (p. 97),

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

  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 24 (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)

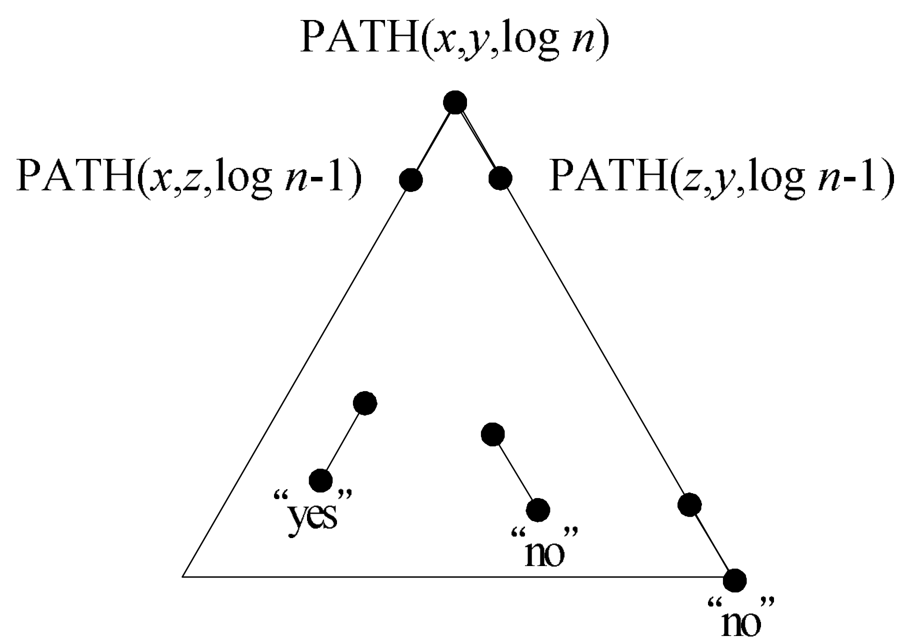
- For $i > 0$, PATH(x, y, i) if and only if there exists a z such that PATH$(x, z, i-1)$ and PATH$(z, y, i-1)$.
- For PATH$(x, y, 0)$, check the input graph or if $x = y$.
- Compute PATH$(x, y, \lceil \log n \rceil)$ with a depth-first search on a graph with nodes (x, y, z, i)s (see next page).[a]
-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

[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 (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 Space and Deterministic Space Only Quadratic

Corollary 25 *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 the input.
- From p. 217, 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.
- In fact, we check node connectedness only when $i = 0$ on p. 224, by examining the input string G.
- 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. 224 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. (1) on p. 217.
- So each node needs space $O(f(n))$.
- As the depth of the recursive call on p. 224 is $O(\log c^{f(n)})$, 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. 202).
- It is known that[a]

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

- So

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

- But it is not known whether coNP = NP.

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