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.
  - M_f halts after $O(|x| + f(|x|))$ steps.
  - M_f uses $O(f(|x|))$ space besides its input x.

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**).
- We shall henceforth use only proper functions in relation to complexity classes $\text{TIME}(f(n))$, $\text{SPACE}(f(n))$, $\text{NTIME}(f(n))$, and $\text{NSPACE}(f(n))$.

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. 128).
  - 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 precise $f(n)$ steps, and
  - All of its strings are at halting of length precisely $g(n)$.
    * 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 10 *Suppose that a (deterministic or nondeterministic) TM M decides L within time (or space) $f(n)$, where f is proper. Then there is a precise TM M' which decides L in time $O(n + f(n))$ (or 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."

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.
  - The simulation stops at the moment the "clock" string is exhausted.
  - The time bound is therefore $O(|x| + f(|x|))$.
- If f is a space bound:
  - M' simulates *on* M_f's output string.
  - The total space, besides the input string, is $O(f(n))$.

The Most 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, $\mathrm{NTIME}(n^k) = \bigcup_{j>0} \mathrm{NTIME}(n^j)$.

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

Complements of Nondeterministic Classes

- From p. 89, 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. 60).

The Halting Problem Quantified

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

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

  where M is deterministic.
- Assume the input is binary.

The Quantified Halting Problem Is in $O(f(n)^3)$

Lemma 11 $H_f \in \text{TIME}(f^3(n))$.

- For each input $M; x$, we simulate M on x with an alarm clock of length $f(|x|)$.
  - Use the simulator (p. 43), the universal TM, and the linear speedup theorem.
- H_f may not be in $\text{TIME}(f(n))$ because the simulator needs to take into account all possible Ms.
  - Just because a Pentium processor can finish a job in 10 seconds does not mean that it takes only 10 seconds to verify that claim.

The Quantified Halting Problem Is Not in $f(\lfloor n/2 \rfloor)$

Lemma 12 $H_f \notin \text{TIME}(f(\lfloor n/2 \rfloor))$.

- Suppose there is a TM M_{H_f} that 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"

- D_f on input M runs in the same time as M_{H_f} on input $M; M$, i.e., in time $f(\lfloor \frac{2n+1}{2} \rfloor) = f(n)$.
- $D_f(D_f) =$ "yes" $\Rightarrow D_f; D_f \notin H_f \Rightarrow D_f(D_f) =$ "no."
- Similarly, $D_f(D_f) =$ "no" $\Rightarrow D_f(D_f) =$ "yes."

The Time Hierarchy Theorem

Theorem 13 *If $f(n) \geq n$ is proper, then*

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

- Combine Lemma 11 and Lemma 12.

Corollary 14 $\text{P} \subsetneq \text{EXP}$.

- $\text{P} \subseteq \text{TIME}(2^n) \subseteq \text{EXP}$ because $\text{poly}(n) \leq 2^n$ for n large enough.
- By Theorem 13,
$\text{TIME}(2^n) \subsetneq \text{TIME}((2^{2n+1})^3) \subseteq \text{TIME}(2^{n^2}) \subseteq \text{EXP}$.

The Space Hierarchy Theorem

Theorem 15 *If $f(n)$ is proper, then*

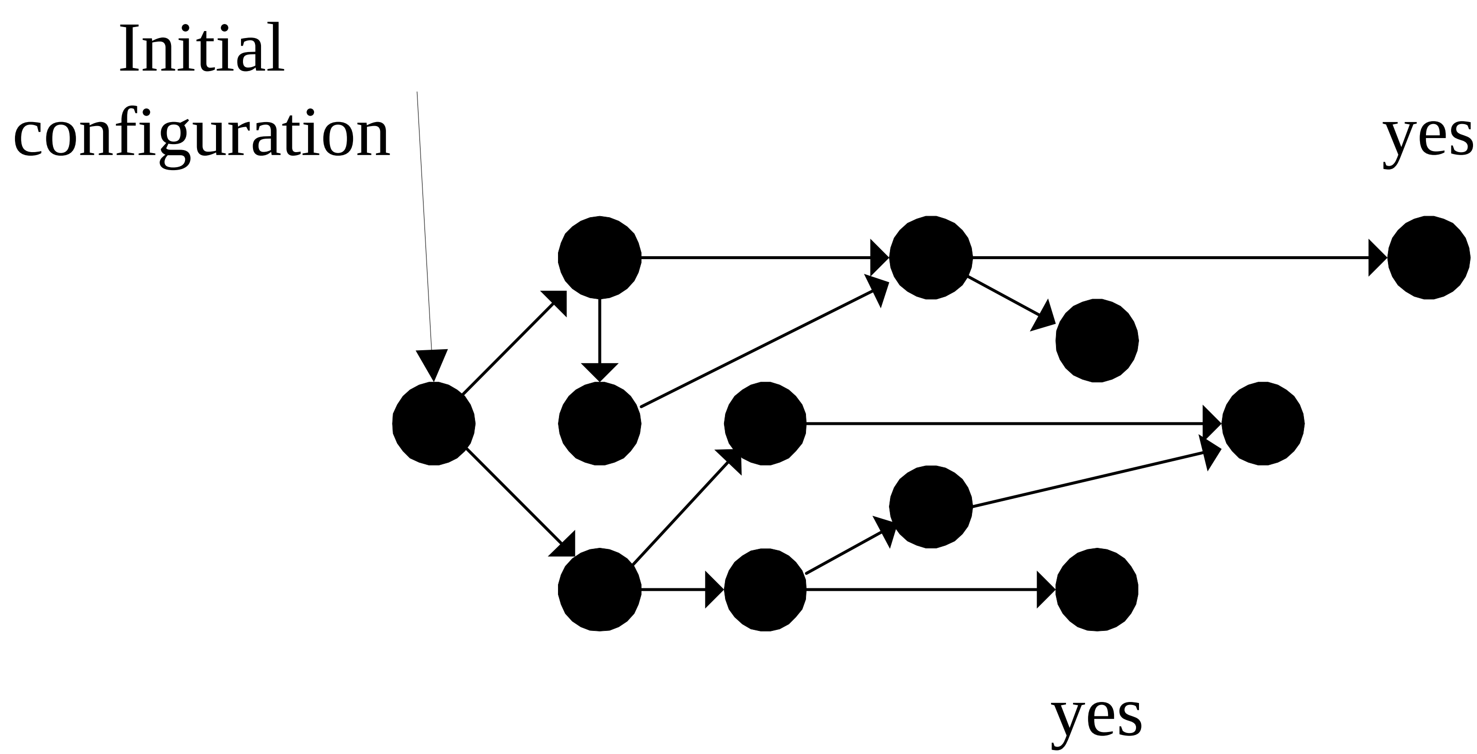
$$\mathrm{SPACE}(f((n) \subsetneq \mathrm{SPACE}(f(n) \log f(n)).$$

Corollary 16 $\mathrm{L} \subsetneq \mathrm{PSPACE}$.

The Reachability Method

- A computation of a TM (deterministic or nondeterministic) can be represented by directional transitions between configurations.
- The reachability method *imagines* 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 17 *Suppose that $f(n)$ is proper. Then*

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

- Proof of 2:
  - Explore the computation tree of the NTM for "yes."
  - Use the *depth-first* search as f is proper.
  - Each path consumes at most $O(f(n))$ space because it takes $O(f(n))$ time, and space can be recycled.

The Proof (continued)

- Proof of 3 (use the reachability method):
  - Generate the configuration graph of a k-string NTM $M = (K, \Sigma, \Delta, s)$ with input and output on input x of length n that decides $L \in \text{NSPACE}(f(n))$.
    * A configuration is a $(2k+1)$-tuple $(q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k)$.
    * 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

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

      for some c_1, which depends on M.

The Proof (continued)

- Add edges to the configuration graph based on the transition function.
- Whether $x \in L$ becomes equivalent to deciding whether there is a path in the configuration graph from the initial configuration to some configuration of the form $(\text{"yes"}, i, \ldots)$ [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 $(c_1^{\log n + f(n)})^k = (c_1^k)^{\log n + f(n)}$.

The Grand Chain of Inclusions

- $\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} \subseteq \mathrm{EXP}$.
- It is known that $\mathrm{PSPACE} \subsetneq \mathrm{EXP}$.
- By Corollary 16 (p. 124), we know $\mathrm{L} \subsetneq \mathrm{PSPACE}$.
- The chain must break somewhere between L and PSPACE.
- We suspect all four inclusions are proper, but there is no proof yet.

Nondeterministic Space and Deterministic Space

- By Theorem 4 (p. 69), $\text{NTIME}(f(n)) \subseteq \text{TIME}(c^{f(n)})$, an exponential gap.
  - There is no proof that the exponential gap is inherent.
- How about NSPACE and SPACE?
- Surprisingly, the relation is only quadratic, a polynomial (Savitch's theorem).

Savitch's Theorem

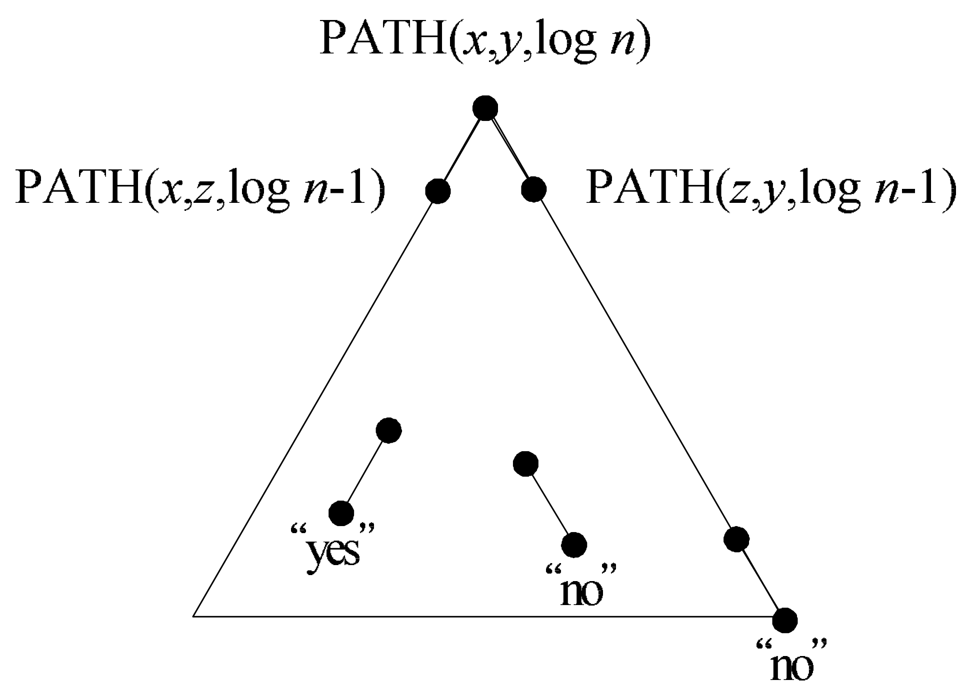
Theorem 18 (Savitch, 1970)
REACHABILITY $\in$ SPACE($\log^2 n$).

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

The Simple Idea for Computing PATH(x, y, i)

- 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 $x = y$.
- We compute PATH$(x, y, \lceil \log n \rceil)$ with a depth-first search on a tree with nodes (x, y, i)s.
- 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, $\lceil \log n \rceil$.

The PATH Tree



- Depth is only $\lceil \log n \rceil$.
- Each node (x, y, i) needs space $O(\log n)$.
- Total space is $O(\log^2 n)$.

The 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 19 *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.
- The graph is implicit—we check for connectedness only when $i = 0$, by examining the input string.
- From p. 128, the configuration graph has $O(c^{f(n)})$ nodes; hence each node takes space $O(f(n))$.

Implications of Savitch's Theorem

- PSPACE = NSPACE.
- Nondeterminism is less powerful with respect to space than it is with respect to time.

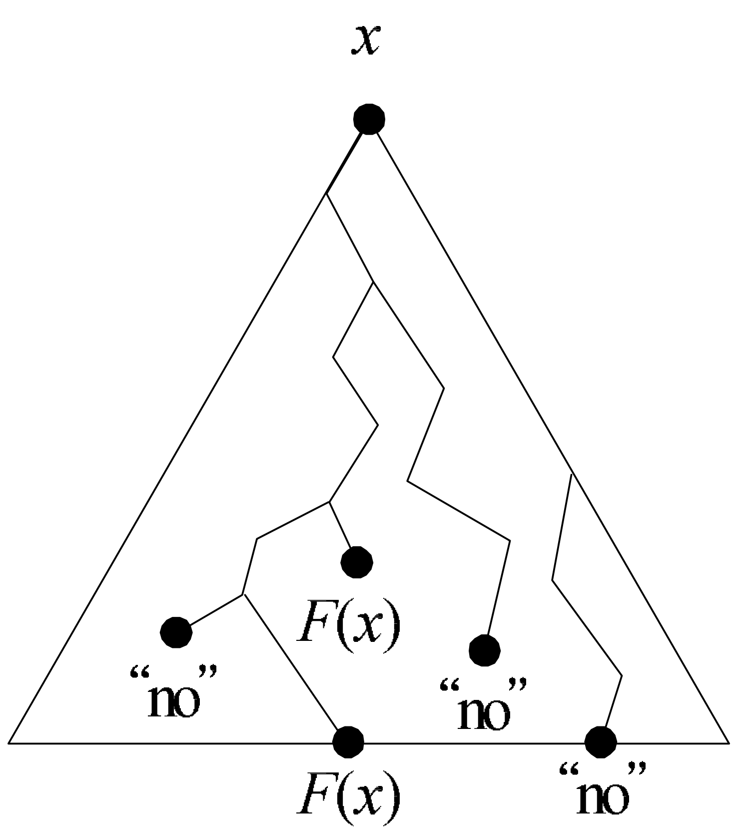
Nondeterministic Space Is Closed under Complement

- We shall prove that
  coNSPACE($f(n)$) = NSPACE($f(n)$).
  - So coNL = NL and coPSPACE = NPSPACE.
  - There is still no hint of coNP = NP.
- The concept is nontrivial only for nondeterministic complexity classes.

Functions and Nondeterministic TMs

- An NTM computes function F if the following hold:
  - On input x, each computation path either outputs the correct answer $F(x)$ or ends up in state "no."
  - At least one computation path ends up with $F(x)$.
  - So all successful paths agree on their output.
- As before, the machine observes a space bound $f(n)$ if at halting all strings (except for the input and output ones) are of length at most $f(|x|)$.

How an NTM Computes a Function



The Immerman-Szelepscényi Theorem

Theorem 20 (Szelepscényi, 1987, Immerman, 1988) *Given a graph G and a node x, the number of nodes reachable from x in G can be computed by an NTM within space $O(\log n)$.*

- The algorithm has four nested loops.
- Let n be the number of nodes.
- $S(k)$ denotes the set of nodes in G that can be reached from x by paths of length at most k.
- So $|S(n-1)|$ is the desired answer.

The Algorithm: Top 2 Levels

```
|S(0)| := 1;
for k = 1, 2, ..., n − 1 do
    {Compute |S(k)| from |S(k − 1)|.}
    ℓ := 0;
    for u = 1, 2, ..., n do
        if u ∈ S(k) then
            ℓ := ℓ + 1;
        end if
    end for
    |S(k)| := ℓ;
end for
return |S(n − 1)|;
```

- Need $|S(k-1)|$, but not earlier ones.

The Third Loop, for $u \in S(k)$

```
m := 0; {Count members of S(k − 1) encountered.}
reply := false;
for v = 1, 2, ..., n do
   if v ∈ S(k − 1) then
      m := m + 1;
      if G(v, u) then
         reply := true;
      end if
   end if
end for
if m < |S(k − 1)| then
   "no";
end if
return reply;
```

The Fourth Loop, for $v \in S(k-1)$

```
s := x;
for i = 1, 2, ..., k − 1 do
    Guess a node t ∈ {1, 2, ..., n}; {Nondeterminism.}
    if (s, t) ∉ G then
        "no";
    end if
    s := t;
end for
if t = v then
    return true;
else
    "no";
end if
```

Wrapping It Up

- The nondeterministic algorithm needs space $O(\log n)$.
  - k, $|S(k-1)|$, ℓ, u, m, v, s, i, t.

Corollary 21 *If* $f \geq \log n$ *is proper, then*

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

- Run the above algorithm on the configuration graph of the NTM M deciding $L \in \text{NSPACE}(f(n))$ on input x.
- We accept only if no accepting configurations have been encountered and if $|S(n-1)|$ is computed.