The Simulation Technique

Theorem 3 *Given any k-string M operating within time $f(n)$, there exists a (single-string) M' operating within time $O(f(n)^2)$ such that $M(x) = M'(x)$ for any input x.*

- The single string of M' implements the k strings of M.
- Represent configuration $(w_1, u_1, w_2, u_2, \ldots, w_k, u_k)$ of M by configuration

$$(q, \triangleright w_1' u_1 \triangleleft w_2' u_2 \triangleleft \cdots \triangleleft w_k' u_k \triangleleft \triangleleft)$$

  of M'.
  - $\triangleleft$ is a special delimiter.
  - w_i' is w_i with the first and last symbols "primed."

The Proof (continued)

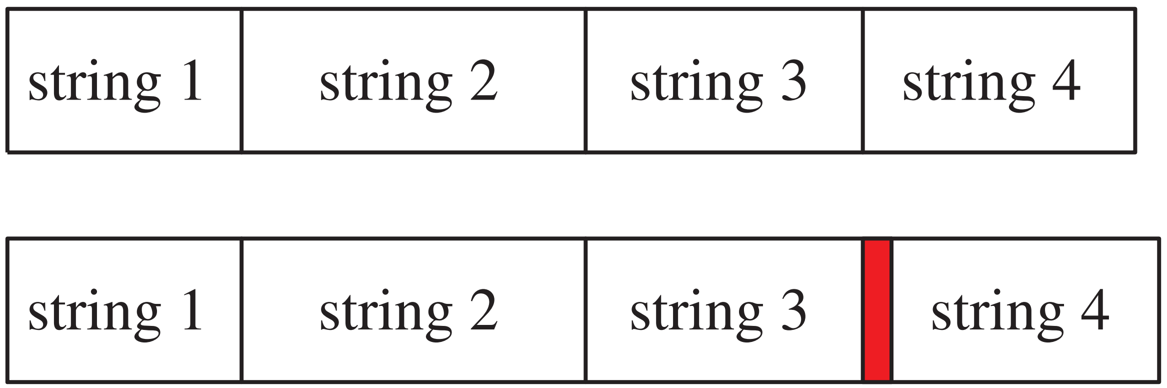
- The initial configuration of M' is

$$(s, \triangleright \triangleright' x \triangleleft \overbrace{\triangleright' \triangleleft \cdots \triangleright' \triangleleft}^{k-1 \text{ pairs}} \triangleleft).$$

- To simulate each move of M:
  - M' scans the string to pick up the k symbols under the cursors.
    - The states of M' must include $K \times \Sigma^k$ to remember them.
    - The transition functions of M' must also reflect it.
  - M' then changes the string to reflect the overwriting of symbols and cursor movements of M.

The Proof (continued)

- It is possible that some strings of M need to be lengthened.
  - The linear-time algorithm on p. 31 can be used for each such string.
- The simulation continues until M halts.
- M' erases all strings of M except the last one.
- Since M halts within time $f(|x|)$, none of its strings ever becomes longer than $f(|x|)$.
- The length of the string of M' at any time is $O(kf(|x|))$.


string 1
string 2
string 3
string 4
string 1
string 2
string 3
string 4

The Proof (concluded)

- Simulating each step of M takes, *per string of M*, $O(kf(|x|))$ steps.
  - $O(f(|x|))$ steps to collect information.
  - $O(kf(|x|))$ steps to write and, if needed, to lengthen the string.
- M' takes $O(k^2 f(|x|))$ steps to simulate each step of M.
- As there are $f(|x|)$ steps of M to simulate, M' operates within time $O(k^2 f(|x|)^2)$.

Linear Speedup

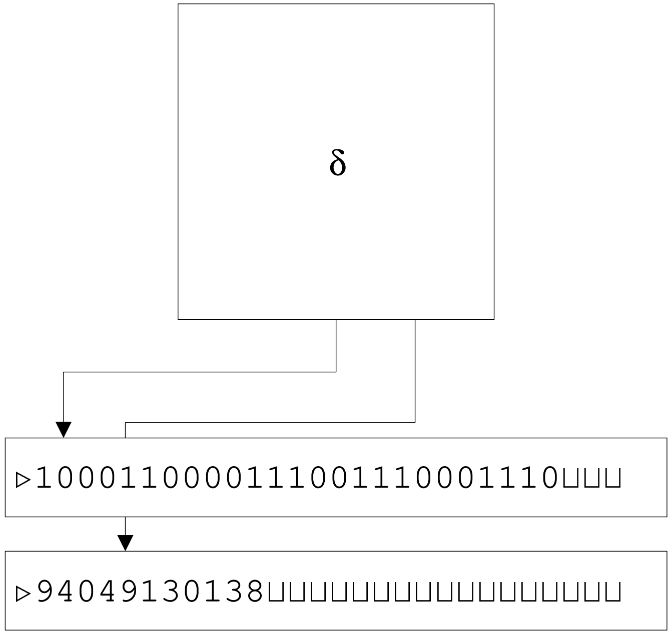
Theorem 4 *Let $L \in TIME(f(n))$. Then for any $\epsilon > 0$, $L \in TIME(f'(n))$, where $f'(n) = \epsilon f(n) + n + 2$.*

- Let L be decided by a k-string TM $M = (K, \Sigma, \delta, s)$ operating within time $f(n)$.
- Our goal is to construct a k'-string $M' = (K', \Sigma', \delta', s')$ operating within the time bound $f'(n)$ and which *simulates* M.
- Set $k' = \max(k, 2)$.
- We encode $m = \lceil 6/\epsilon \rceil$ symbols of M in *one* symbol of M' so that M' can simulate m steps of M within 6 steps.

The Proof (continued)

- $\Sigma' = \Sigma \cup \Sigma^m$.
- Phase one of M':
  - M' has states corresponding to $K \times \Sigma^m$.
  - Map each block of m symbols of the input $\sigma_1 \sigma_2 \cdots \sigma_m$ to the *single* symbol $(\sigma_1 \sigma_2 \cdots \sigma_m) \in \Sigma'$ of M' to the second string.
  - Doable because M' has the states for remembering.
- This phase takes $m \lceil |\, x \,| / m \rceil + 2$ steps.
  - The extra 2 comes from the enclosing symbols $\rhd$ and $\sqcup$.

Compression of Symbols; Increasing the Word Length



- $m = 3$.
- 3-ary representation, with $\sqcup \to 2$.

The Proof (continued)

- Treat the second string as the one containing the input.
  - If $k > 1$, use the first string as an ordinary work string.
- M' simulates m steps of M by six or fewer steps, called a **stage**.
- A stage begins with M' in state $(q, j_1, j_2, \ldots, j_k)$.
  - $q \in K$ and $j_i \leq m$ is the position of the ith cursor within the m-tuple scanned.
  - If the ith cursor of M is at the ℓth symbol after $\triangleright$, then the $(i+1)$st cursor of M' will point to the $\lceil \ell/m \rceil$th symbol after $\triangleright$ and $j_i = ((\ell - 1) \bmod m) + 1$.

The Proof (continued)

$\triangleright$	100	011	000	011	100	111	000	111	0⊔⊔	⊔

- $m = 3$.
- $\ell = 8$.
- $\lceil \ell/m \rceil = \lceil 8/3 \rceil = 3$.
- $j_i = ((8-1) \bmod 3) + 1 = 2$.

The Proof (continued)

- Then M' moves all cursors to the left by one position, then to the right twice, and then to the left once.
  - This takes 4 steps.
  - No cursor of M can in m moves get out of the m-tuples scanned by M' above.
- M' now "remembers" all symbols (of Σ') at or next to all cursors.
  - M' needs states in $K \times \{1, 2, \ldots, m\}^k \times \Sigma^{3mk}$, a $m^k \cdot |\Sigma|^{3mk}$-fold increase.
- M' has all the information needed to know the next m moves of M!

The Proof (concluded)

- M' uses its δ' function to implement the changes in string contents and state brought about by the next m moves of M.
  - This takes 2 steps: One for the current m-tuple and one for one of its two neighbors.
- The total number of M' steps is at most 6 per stage.
- The total number of M' steps is at most

$$|\,x\,| + 2 + 6 \times \left\lceil \frac{f(|\,x\,|)}{m} \right\rceil \leq |\,x\,| + 2 + \epsilon f(|\,x\,|).$$

$$\begin{tikzcd} & \boxed{\alpha \quad \beta \quad \gamma} \arrow[dl] \arrow[dr] & \\ \boxed{\alpha' \quad \beta' \quad \gamma} & & \boxed{\alpha \quad \beta' \quad \gamma'} \end{tikzcd}$$

Implications of the Speedup Theorem

- State size can be traded for speed.
  - $m^k \cdot |\Sigma|^{3mk}$-fold increase to gain a speedup of $O(m)$.
- If $f(n) = cn$ with $c > 1$, then c can be made arbitrarily close to 1.
- If $f(n)$ is superlinear, say $f(n) = 14n^2 + 31n$, then the constant in the leading term (14 in this example) can be made arbitrarily small.
  - *Arbitrary* linear speedup can be achieved.
  - This justifies the asymptotic big-O notation.
- 1-bit, 4-bit, 8-bit, 16-bit, 32-bit, 64-bit, 128-bit CPUs, and so on.

P

- By the linear speedup theorem, any polynomial time bound can be represented by its leading term n^k for some $k \geq 1$.
- If L is a polynomially decidable language, it is in $\mathrm{TIME}(n^k)$ for some $k \in \mathbb{N}$.
- The union of all polynomially decidable languages is denoted by P:

$$\mathrm{P} = \bigcup_{k>0} \mathrm{TIME}(n^k).$$

- Problems in P can be efficiently solved.

Charging for Space

- We do not want to charge the space used only for input and output.
- Let $k > 2$ be an integer.
- A **k-string Turing machine with input and output** is a k-string TM that satisfies the following conditions.
  - The input string is *read-only*.
  - The last string, the output string, is *write-only*.
    - That is, the cursor never moves to the left.
  - The cursor of the input string does not wander off into the $\sqcup$s.

Space Complexity

- Consider a k-string TM M with input x.
- We may assume $\sqcup$ is never written over a non-$\sqcup$ symbol.
- If M halts in configuration $(H, w_1, u_1, w_2, u_2, \ldots, w_k, u_k)$, then the **space required by M on input x** is $\sum_{i=1}^{k} |w_i u_i|$.
- If M is a TM with input and output, then the space required by M on input x is $\sum_{i=2}^{k-1} |w_i u_i|$.
- Machine M **operates within space bound** $f(n)$ for $f : \mathbb{N} \to \mathbb{N}$ if for any input x, the space required by M on x is at most $f(|x|)$.

Space Complexity Classes

- Let L be a language.
- Then

$$L \in \text{SPACE}(f(n))$$

if there is a TM with input and output that decides L and operates within space bound $f(n)$.

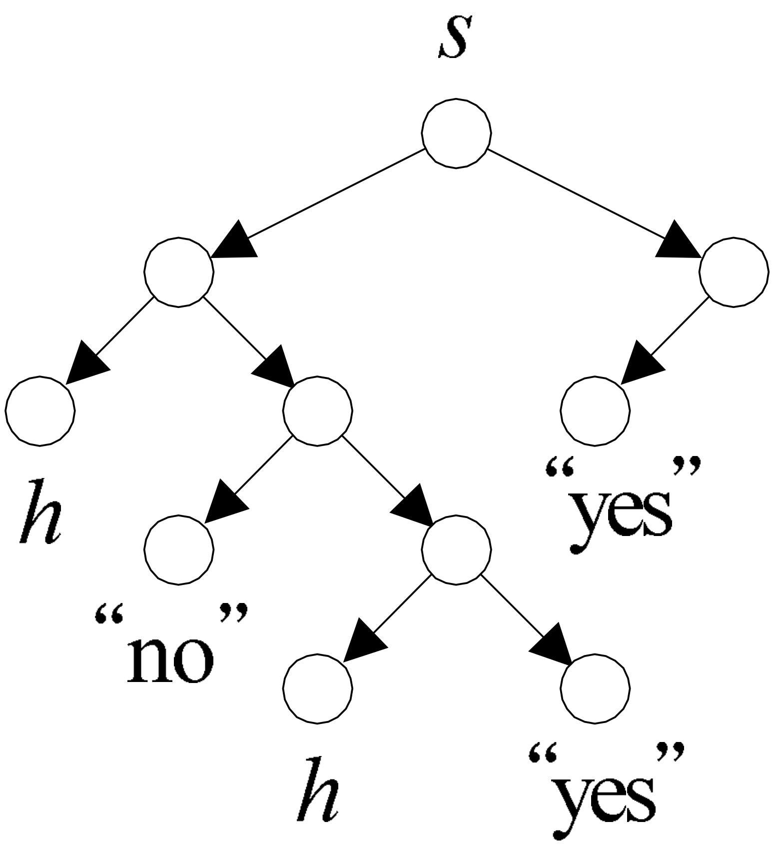
- SPACE($f(n)$) is a set of languages.
  - Palindrome is in SPACE($\log n$): Keep 3 pointers.
- As in the linear speedup theorem (Theorem 4), constant coefficients do not matter.

Nondeterminism[a]

- A **nondeterministic Turing machine** (**NTM**) is a quadruple $N = (K, \Sigma, \Delta, s)$.
- K, Σ, s are as before.
- $\Delta \subseteq K \times \Sigma \to (K \cup \{h, \text{"yes"}, \text{"no"}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ is a relation, not a function.
  - For each state-symbol combination, there may be more than one next steps—or none at all.
- A configuration yields another configuration in one step if there *exists* a rule in Δ that makes this happen.

[a] Rabin, Scott, 1959.

Computation Tree and Computation Path



Decidability under Nondeterminism

- Let L be a language and N be an NTM.
- N **decides** L if for any $x \in \Sigma^*$, $x \in L$ if and only if there is a sequence of valid configurations that ends in "yes."
  - It is not required that the NTM halts in all computation paths.
- So if $x \notin L$, then no nondeterministic choices should lead to a "yes" state.
- Determinism is a special case of nondeterminism.

An Example

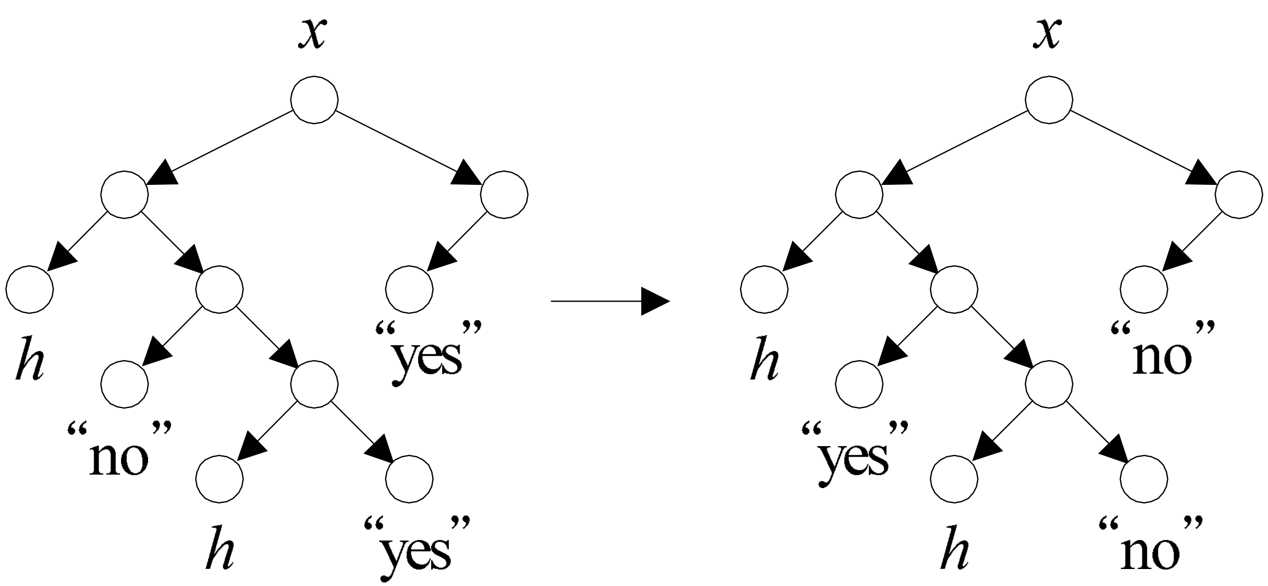
- Let L be the set of logical conclusions of a set of axioms.
- Consider the nondeterministic algorithm:

  1: $b :=$ `false`;
  2: **while** the input predicate $\phi \neq b$ **do**
  3: Generate a logical conclusion of b by applying some of the axioms; {Nondeterministic choice.}
  4: **end while**
  5: "yes";

- This algorithm decides L.

Complementing a TM's Halting States

- Let M decide L, and M' be M after "yes" $\leftrightarrow$ "no".
- If M is a (deterministic) TM, then M' decides $\bar{L}$.
- But if M is an NTM, then M' may not decide $\bar{L}$.
  - It is possible that both M and M' accept x.

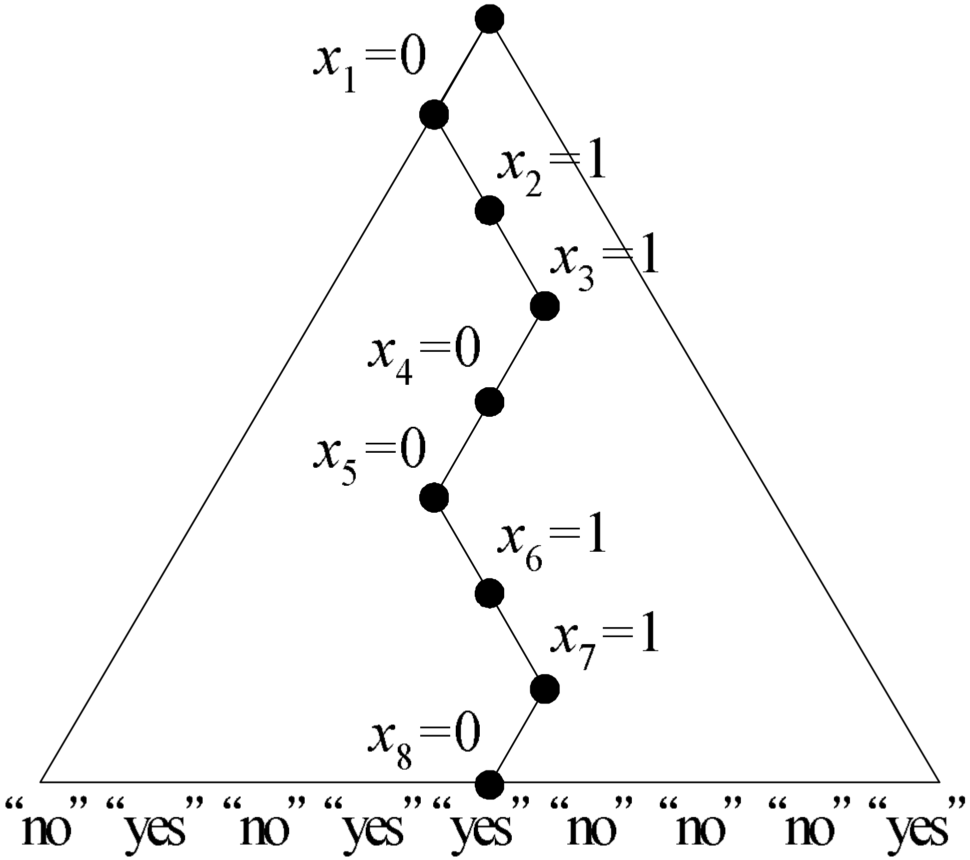


A Nondeterministic Algorithm for Satisfiability

ϕ is a boolean formula with n variables.

for $i = 1, 2, \ldots, n$ **do**
    Guess $x_i \in \{0, 1\}$; {Nondeterministic choice.}
end for
{Verification:}
if $\phi(x_1, x_2, \ldots, x_n) = 1$ **then**
    "yes";
else
    "no";
end if

The Computation Tree for Satisfiability



Analysis

- The algorithm decides language $\{\phi : \phi \text{ is satisfiable}\}$.
  - The computation tree is a complete binary tree of depth n.
  - Every computation path corresponds to a particular truth assignment out of 2^n.
  - ϕ is satisfiable if and only if there is a computation path (truth assignment) that results in "yes."
- General paradigm: Guess a "proof" and then verify it.

The Traveling Salesman Problem

- We are given n cities $1, 2, \ldots, n$ and integer distances d_{ij} between any two cities i and j.
- Assume $d_{ij} = d_{ji}$ for convenience.
- The **traveling salesman problem** (TSP) asks for the total distance of the shortest tour of the cities.
- The decision version TSP (D) asks if there is a tour with a total distance at most B, where B is an input.
- Both problems are extremely important but hard.

A Nondeterministic Algorithm for TSP (D)

for $i = 1, 2, \ldots, n$ **do**
  Guess $x_i \in \{1, 2, \ldots, n\}$; {The ith city.}
end for
$x_{n+1} := x_1$;
{Verification stage:}
if $x_1, x_2, \ldots, x_n$ are distinct and $\sum_{i=1}^{n} d_{x_i, x_{i+1}} \leq B$ **then**
  "yes";
else
  "no";
end if

(The degree of nondeterminism is n.)

Time Complexity under Nondeterminism

- Nondeterministic machine N decides L **in time** $f(n)$, where $f : \mathbb{N} \to \mathbb{N}$, if
  - N decides L, and
  - for any $x \in \Sigma^*$, N does not have a computation path longer than $f(|x|)$.
- We charge only the "depth" of the computation tree.

Time Complexity Classes under Nondeterminism

- NTIME($f(n)$) is the set of languages decided by NTMs within time $f(n)$.
- NTIME($f(n)$) is a complexity class.

NP

- Define

$$\text{NP} = \bigcup_{k>0} \text{NTIME}(n^k).$$

- Clearly $\text{P} \subseteq \text{NP}$.
- Think of NP as efficiently *verifiable* problems.
  - Boolean satisfiability (SAT).
  - TSP (D).
  - Hamiltonian path.
  - Graph colorability.
- The most important open problem in theoretical computer science is whether $\text{P} = \text{NP}$.

Simulating Nondeterministic TMs

Theorem 5 *Suppose language L is decided by an NTM N in time $f(n)$. Then it is decided by a 3-string deterministic TM M in time $O(c^{f(n)})$, where $c > 1$ is some constant depending on N.*

- On input x, M goes down every computation path of N using *depth-first* search (M does *not* know $f(n)$).
- If some path leads to "yes," then M enters the "yes" state.
- If none of the paths leads to "yes," then M enters the "no" state.

NTIME vs. TIME

Corollary 6 $\text{NTIME}(f(n))) \subseteq \bigcup_{c>1} \text{TIME}(c^{f(n)})$.

- Does converting an NTM into a TM require exploring all the computation paths of the NTM as done in Theorem 5?
- That is the six-million-dollar question.

A Nondeterministic Algorithm for Graph Reachability

```
x := 1;
for i = 2, 3, ..., n do
    Guess y ∈ {2, 3, ..., n}; {The next node.}
    if (x, y) ∈ G then
        if y = n then
            "yes"; {Node n is reached from node 1.}
        else
            x := y;
        end if
    else
        "no";
    end if
end for
"no";
```

Space Analysis

- Variables i, x, and y each require $O(\log n)$ bits.
- Testing if $(x, y) \in G$ is accomplished by consulting the input string with counters of $O(\log n)$ bit long.
- Hence

$$\text{REACHABILITY} \in \text{NSPACE}(\log n).$$

  - REACHABILITY with more than one terminal node also has the same complexity.
- It is also known that REACHABILITY $\in$ P (p. 159).

Infinite Sets

- A set is **countable** if it is finite or if it can be put in one-one correspondence with the set of natural numbers.
  - Set of integers $\mathbb{Z}$.
  - Set of positive integers $\mathbb{Z}^+$.
  - Set of odd integers.
  - Set of rational numbers $(1/1, 1/2, 2/1, 1/3, 2/2, 3/1, 1/4, 2/3, 3/2, 4/1, \ldots)$.
  - Set of squared integers.

Rational Numbers Are Countable

1/1	1/2	1/3	1/4	1/5	1/6
2/1	2/2	2/3	2/4	2/5	
3/1	3/2	3/3	3/4		
4/1	4/2	4/3			
5/1					

Cardinality

- Let A denote a set.
- Then 2^A denotes its **power set**, that is $\{B : B \subseteq A\}$.
  - If $|A| = k$, then $|2^A| = 2^k$.
- For any set C, define $|C|$ as C's **cardinality** (size).
- Two sets are said to have the same cardinality (written as $|A| = |B|$ or $A \sim B$) if there exists a one-to-one correspondence between their elements.

Cardinality (concluded)

- $|A| \leq |B|$ if there is a one-to-one correspondence between A and one of B's subsets.
- $|A| < |B|$ if $|A| \leq |B|$ but $|A| \neq |B|$.
- If $A \subseteq B$, then $|A| \leq |B|$.
- But if $A \subsetneq B$, then $|A| < |B|$?

Cardinality and Infinite Sets

- If A and B are infinite sets, it is possible that $A \subsetneq B$ yet $|A| = |B|$.
  - The set of integers *properly* contains the set of odd integers.
  - But the set of integers has the same cardinality as the set of odd integers.
- A lot of "paradoxes."

Hilbert's[a] Paradox of the Grand Hotel

- For a hotel with a finite number of rooms with all the rooms occupied, a new guest will be turned away.
- Now let us imagine a hotel with an infinite number of rooms, and all the rooms are occupied.
- A new guest comes and asks for a room.
- "But of course!" exclaims the proprietor, and he moves the person previously occupying Room 1 into Room 2, the person from Room 2 into Room 3, and so on ....
- The new customer occupies Room 1.

[a]David Hilbert (1862–1943).

Hilbert's Paradox of the Grand Hotel (concluded)

- Let us imagine now a hotel with an infinite number of rooms, all taken up, and an infinite number of new guests who come in and ask for rooms.
- "Certainly, gentlemen," says the proprietor, "just wait a minute."
- He moves the occupant Room 1 into Room 2, the occupant of Room 2 into Room 4, and so on.
- Now all odd-numbered rooms become free and the infinity of new guests can be accommodated in them.
- "There are many rooms in my Father's house, and I am going to prepare a place for you." (*John* 14:3)

Galileo's[a] Paradox (1638)

- The squares of the positive integers can be placed in one-to-one correspondence with all the positive integers.
- This is contrary to the axiom of Euclid that the whole is greater than any of its proper parts.
- Resolution of paradoxes: Which notion results in better mathematics.

[a]Galileo (1564–1642).

Cantor's[a] Theorem

Theorem 7 *The set of all subsets of N (2^N) is infinite and not countable.*

- Suppose it is countable with $f : N \to 2^N$ being a bijection.
- Consider the set $B = \{k \in N : k \notin f(k)\} \subseteq N$.
- Suppose $B = f(n)$ for some n.

---

[a] Georg Cantor (1845–1918).

The Proof (concluded)

- If $n \in f(n)$, then $n \in B$, but then $n \notin B$ by B's definition.
- If $n \notin f(n)$, then $n \notin B$, but then $n \in B$ by B's definition.
- Hence $B \neq f(n)$ for any n.
- f is not a bijection, a contradiction.

A Corollary of Cantor's Theorem

Corollary 8 *For any set T, finite or infinite,*

$$|\,T\,| < |\,2^T\,|.$$

- $|T| \leq |2^T|$ as $f(x) = \{x\}$ maps T into a subset of 2^T.
- The strict inequality uses the same argument as Cantor's theorem.

A Second Corollary of Cantor's Theorem

Corollary 9 *The set of all functions on* $\mathbb{N}$ *is not countable.*

- Every function $f : \mathbb{N} \rightarrow \{0, 1\}$ determines a set

$$\{n : f(n) = 1\} \subseteq \mathbb{N}.$$

- And vice versa.
- So the set of functions from $\mathbb{N}$ to $\{0, 1\}$ has cardinality $|\, 2^{\mathbb{N}} \,|$.
- Corollary 8 (p. 102) then implies the claim.

Existence of Uncomputable Problems

- Every program is a sequence of 0s and 1s.
- Every program corresponds to some integer.
- The set of programs is countable.
- A function is a mapping from integers to integers by Corollary 9 (p. 103).
- The set of functions is not countable.
- So there must exist functions for which there are no programs.

Universal Turing Machine[a]

- A **universal Turing machine** U interprets the input as the *description* of a TM M concatenated with the *description* of an input to that machine, x.
  - Both M and x are over the alphabet of U.
- U simulates M on x so that

$$U(M; x) = M(x).$$

- U is like a modern computer, which executes any valid machine code, or a Java Virtual machine, which executes any valid bytecode.

---
[a]Turing, 1936.

The Halting Problem

- **Undecidable problems** are problems that have no algorithms or languages that are not recursive.
- We knew undecidable problems exist (p. 104).
- We now define a concrete undecidable problem, the **halting problem**:

$$H = \{M; x : M(x) \neq \nearrow\}.$$

  – Does M halt on input x?

H Is Recursively Enumerable

- Use the universal TM U to simulate M on x.
- When M is about to halt, U enters a "yes" state.
- This TM accepts H.
- Membership of x in any recursively enumerative language accepted by M can be answered by asking "$M; x \in H$?"

H Is Not Recursive

- Suppose there is a TM M_H that *decides* H.
- Consider the program $D(M)$ that calls M_H:

```
if M_H(M; M) = "yes" then
    ↗; {Writing an infinite loop is easy, right?}
else
    "yes";
end if
```

- Consider $D(D)$:
  - $D(D) = \nearrow \Rightarrow M_H(D; D) = \text{“yes”} \Rightarrow D; D \in H \Rightarrow D(D) \neq \nearrow$, a contradiction.
  - $D(D) = \text{“yes”} \Rightarrow M_H(D; D) = \text{“no”} \Rightarrow D; D \notin H \Rightarrow D(D) = \nearrow$, a contradiction.

Comments

- Two levels of interpretations of M:
  - A sequence of 0s and 1s (data).
  - An encoding of instructions (programs).
- There are no paradoxes.
  - Concepts are familiar to computer scientists (but not philosophers or mathematicians).
  - Supply a C compiler to a C compiler, a Lisp interpreter to a Lisp interpreter, a Java compiler to a Java compiler, etc.

Self-Loop Paradoxes

Cantor's Paradox (1899): Let T be the set of all sets.

- Then $2^T \subseteq T$, but we know $|2^T| > |T|$!

Russell's[a] Paradox (1901): Consider $S = \{A : A \notin A\}$.

- If $S \in S$, then $S \notin S$ by definition.
- If $S \notin S$, then $S \in S$ also by definition.

Eubulides: The Cretan says, "All Cretans are liars."

Sharon Stone in *The Specialist*: "I'm not a woman you can trust."