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."
- In other words,
  - If $x \in L$, then $N(x) =$ "yes" for some computation path.
  - If $x \notin L$, then $N(x) \neq$ "yes" for all computation paths.

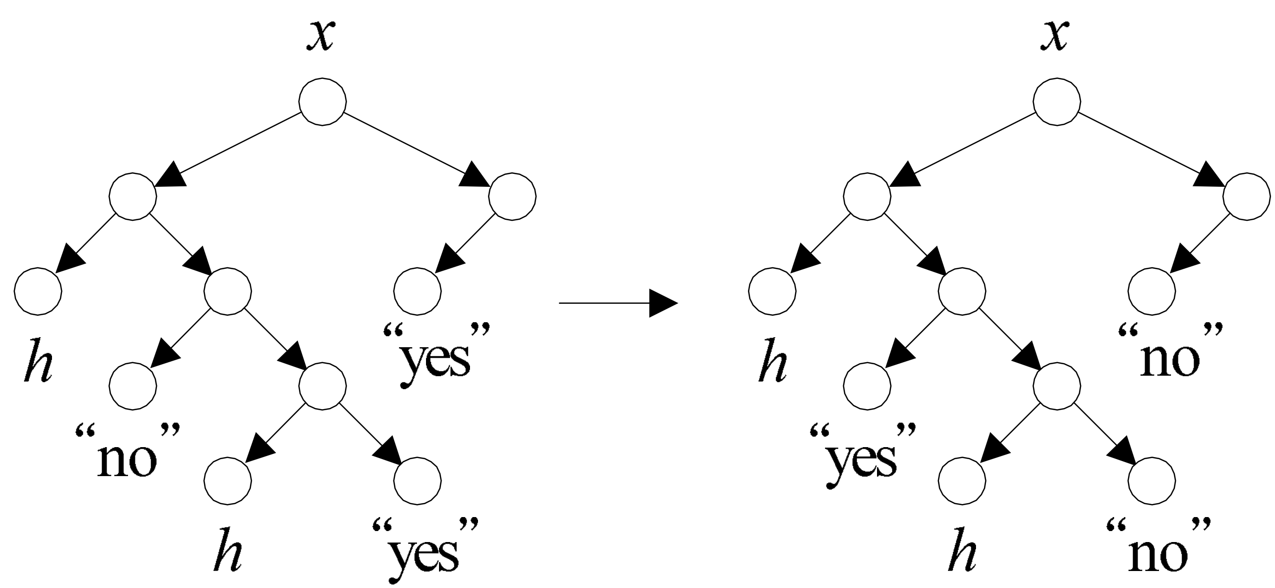
Decidability under Nondeterminism (concluded)

- It is not required that the NTM halts in all computation paths.[a]
- If $x \notin L$, no nondeterministic choices should lead to a "yes" state.
- The key is the algorithm's *overall* behavior not whether it gives a correct answer for each particular run.
- Note that determinism is a special case of nondeterminism.

[a] So "accepts" is a more proper term, and other books use "decides" only when the NTM always halts.

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 $\bar{L}$.
  - So M and M' decide languages that are complements of each other.
- But if M is an NTM, then M' may not decide $\bar{L}$.
  - It is possible that both M and M' accept x (see next page).
  - So M and M' accept languages that are not complements of each other.


x
h
"no"
h
"yes"
"yes"
x
h
"yes"
h
"no"
"no"

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 ("Nondeterministic Polynomial")

- Define
$$\text{NP} = \bigcup_{k>0} \text{NTIME}(n^k).$$
- Clearly $\text{P} \subseteq \text{NP}$.
- Think of NP as efficiently *verifiable* problems (see p. 327).
  - Boolean satisfiability (p. 113 and p. 193).
- The most important open problem in computer science is whether $\text{P} = \text{NP}$.

Simulating Nondeterministic TMs

Nondeterminism does not add power to TMs.

Theorem 6 *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* need to know $f(n)$.
  - As N is time-bounded, the depth-first search will not run indefinitely.

The Proof (concluded)

- If any path leads to “yes,” then M immediately enters the “yes” state.
- If none of the paths leads to “yes,” then M enters the “no” state.
- The simulation takes time $O(c^{f(n)})$ for some $c > 1$ because the computation tree has that many nodes.

Corollary 7 $\text{NTIME}(f(n))) \subseteq \bigcup_{c>1} \text{TIME}(c^{f(n)})$.[a]

---

[a] Mr. Kai-Yuan Hou (B99201038, R03922014) on October 6, 2015: $\bigcup_{c>1} \text{TIME}(c^{f(n)}) \subseteq \text{NTIME}(f(n)))$?

NTIME vs. TIME

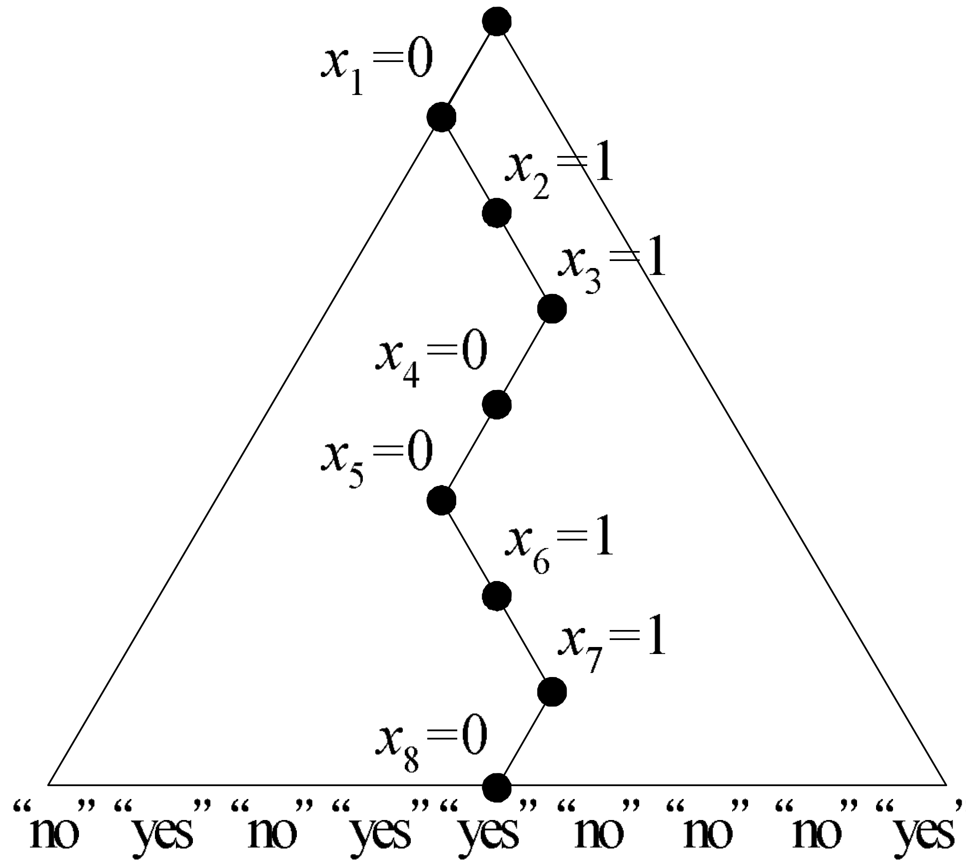
- Does converting an NTM into a TM require exploring all computation paths of the NTM as done in Theorem 6 (p. 110)?
- This is the most important question in theory with important practical implications.

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 choices.}
end for
{Verification:}
if $\phi(x_1, x_2, \ldots, x_n) = 1$ **then**
  "yes";
else
  "no";
end if

Computation Tree for Satisfiability



Analysis

- The computation tree is a complete binary tree of depth n.
- Every computation path corresponds to a particular truth assignment[a] out of 2^n.
- ϕ is satisfiable iff there is a truth assignment that satisfies ϕ.

---

[a]Or a sequence of nondeterministic choices.

Analysis (concluded)

- The algorithm decides language $\{\phi : \phi \text{ is satisfiable}\}$.
  - Suppose ϕ is satisfiable.
    - That means there is a truth assignment that satisfies ϕ.
    - So there is a computation path that results in "yes."
  - Suppose ϕ is not satisfiable.
    - That means every truth assignment makes ϕ false.
    - So every computation path results in "no."
- General paradigm: Guess a "proof" then verify it.

The Traveling Salesman Problem

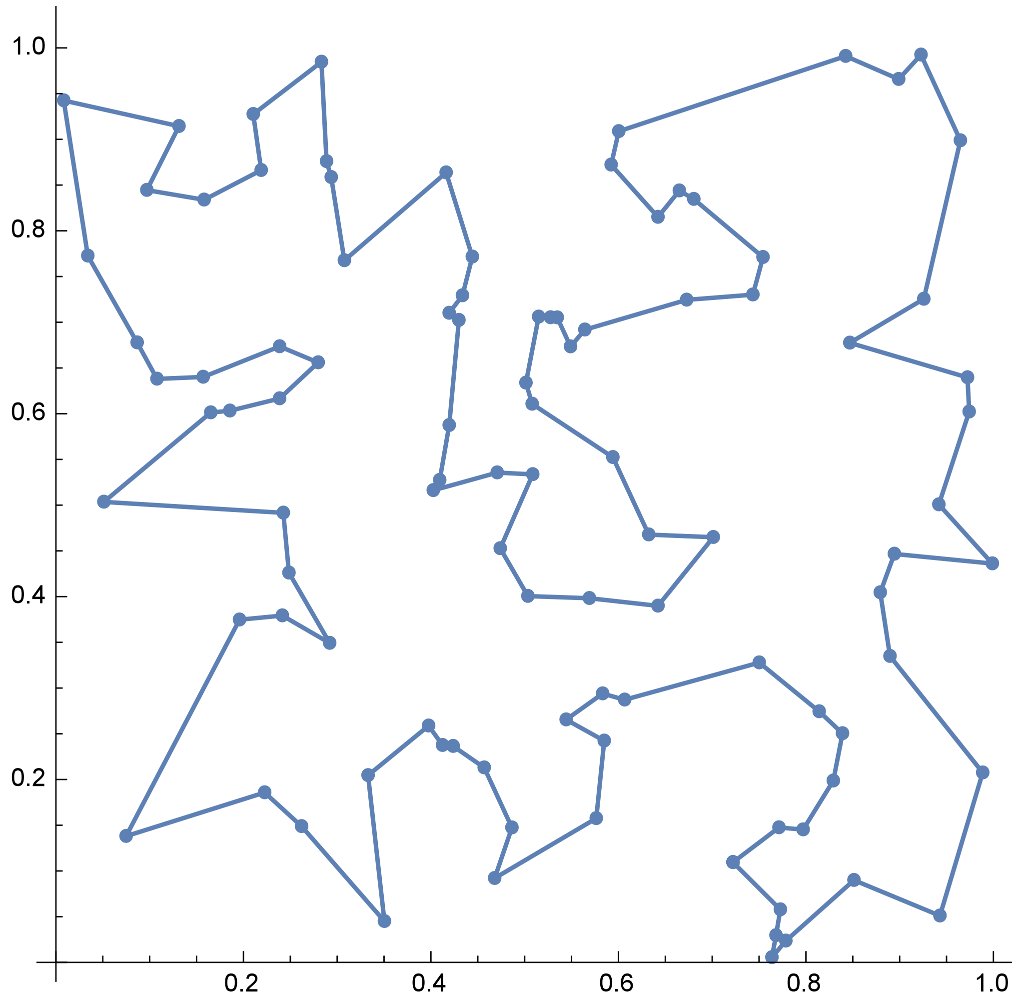
- We are given n cities $1, 2, \ldots, n$ and integer distance d_{ij} between any two cities i and j.
- Assume $d_{ij} = d_{ji}$ for convenience.
- The **traveling salesman problem** (TSP) asks for the total distance of the shortest tour of the cities.[a]
- The decision version TSP (D) asks if there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

---

[a] Each city is visited exactly once.

[b] Both problems are extremely important and are equally hard (p. 391 and p. 493).

A Shortest Path



A Nondeterministic Algorithm for TSP (D)

for $i = 1, 2, \ldots, n$ **do**
  Guess $x_i \in \{1, 2, \ldots, n\}$; {The ith city.}[a]
end for
$x_{n+1} := x_1$;
{Verification:}
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

---

[a] Can be made into a series of $\log_2 n$ *binary* choices for each x_i so that the next-state count (2) is a constant, independent of input size. Contributed by Mr. Chih-Duo Hong (`R95922079`) on September 27, 2006.

Analysis

- Suppose the input graph contains at least one tour of the cities with a total distance at most B.
  - Then there is a computation path for that tour.[a]
  - And it leads to "yes."
- Suppose the input graph contains no tour of the cities with a total distance at most B.
  - Then every computation path leads to "no."

[a]It does not mean the algorithm will follow that path. It just means such a computation path (i.e., a sequence of nondeterministic choices) exists.

Remarks on the $P \stackrel{?}{=} NP$ Open Problem[a]

- Many practical applications depend on answers to the $P \stackrel{?}{=} NP$ question.
- Verification of password should be easy (so it is in NP).
  - A computer should not take a long time to let a user log in.
- A password system should be hard to crack (loosely speaking, cracking it should not be in P).
- It took logicians 63 years to settle the Continuum Hypothesis; how long will it take for this one?

[a] Contributed by Mr. Kuan-Lin Huang (B96902079, R00922018) on September 27, 2011.

Nondeterministic Space Complexity Classes

- Let L be a language.
- Then

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

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

- NSPACE$(f(n))$ is a set of languages.
- As in the linear speedup theorem (Theorem 5 on p. 89), constant coefficients do not matter.

Graph Reachability

- Let $G(V, E)$ be a directed graph (**digraph**).
- REACHABILITY asks, given nodes a and b, does G contain a path from a to b?
- Can be easily solved in polynomial time by breadth-first search.
- How about its nondeterministic space complexity?

The First Try: NSPACE$(n \log n)$

Determine the number of nodes m; $\{$Note $m \leq n$.$\}$
$x_1 := a$; $\{$Assume $a \neq b$.$\}$
for $i = 2, 3, \ldots, m$ **do**
    Guess $x_i \in \{ v_1, v_2, \ldots, v_m \}$; $\{$The ith node.$\}$
end for
for $i = 2, 3, \ldots, m$ **do**
    if $(x_{i-1}, x_i) \not\in E$ **then**
        "no";
    end if
    if $x_i = b$ **then**
        "yes";
    end if
end for
"no";

In Fact, REACHABILITY $\in$ NSPACE$(\log n)$

Determine the number of nodes m; {Note $m \leq n$.}
$x := a$;
for $i = 2, 3, \ldots, m$ **do**
    Guess $y \in \{v_1, v_2, \ldots, v_m\}$; {The next node.}
    if $(x, y) \notin E$ **then**
        "no";
    end if
    if $y = b$ **then**
        "yes";
    end if
    $x := y$;
end for
"no";

Space Analysis

- Variables m, i, x, and y each require $O(\log n)$ bits.
- Testing $(x, y) \in E$ is accomplished by consulting the input string with counters of $O(\log n)$ bits long.
- Hence
$$\text{REACHABILITY} \in \text{NSPACE}(\log n).$$
  - REACHABILITY with more than one terminal node also has the same complexity.
- REACHABILITY $\in$ P (see, e.g., p. 237).

Undecidability

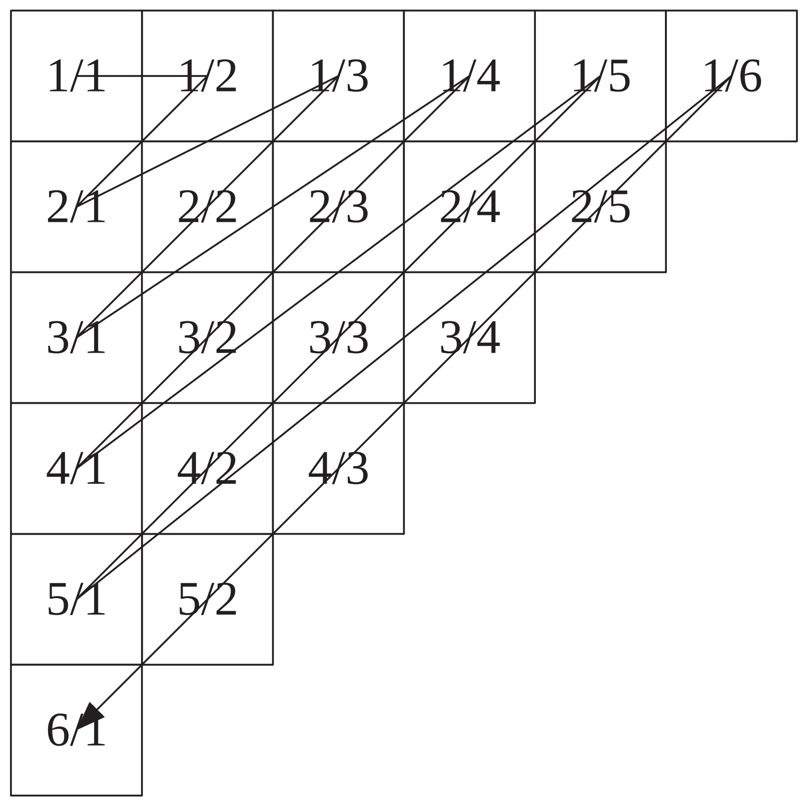
God exists since mathematics is consistent,
and the Devil exists since we cannot prove it.
— André Weil (1906–1998)

Whatsoever we imagine is *finite*.
Therefore there is no idea, or conception
of any thing we call *infinite*.
— Thomas Hobbes (1588–1679), *Leviathan*

Infinite Sets

- A set is **countable** if it is finite or if it can be put in one-one correspondence with $\mathbb{N} = \{ 0, 1, \ldots \}$, the set of natural numbers.
  - Set of integers $\mathbb{Z}$.
    - $0 \leftrightarrow 0$.
    - $1 \leftrightarrow 1, 2 \leftrightarrow 3, 3 \leftrightarrow 5, \ldots$.
    - $-1 \leftrightarrow 2, -2 \leftrightarrow 4, -3 \leftrightarrow 6, \ldots$.
  - Set of positive integers $\mathbb{Z}^+$: $i \leftrightarrow i - 1$.
  - Set of positive odd integers: $i \leftrightarrow (i-1)/2$.
  - Set of (positive) rational numbers $\mathbb{Q}$: See next page.
  - Set of squared integers: $i \leftrightarrow \sqrt{i}$.

Rational Numbers Are Countable



Cardinality

- For any set A, define $|A|$ as A's **cardinality** (size).
- Two sets are said to have the same cardinality, or

$$|\,A\,| = |\,B\,| \quad \text{or} \quad A \sim B,$$

if there exists a one-to-one correspondence between their elements.

- 2^A denotes set A's **power set**, that is $\{B : B \subseteq A\}$.
  - The power set of $\{\,0, 1\,\}$ is

$$2^{\{\,0,1\,\}} = \{\,\emptyset, \{\,0\,\}, \{\,1\,\}, \{\,0, 1\,\}\,\}.$$

- If $|\,A\,| = k$, then $|\,2^A\,| = 2^k$.

Cardinality (concluded)

- Define $|A| \leq |B|$ if there is a one-to-one correspondence between A and a subset of B's.
- Obviously, if $A \subseteq B$, then $|A| \leq |B|$ (prove it!).
  - So $|\mathbb{N}| \leq |\mathbb{Z}|$.
  - So $|\mathbb{N}| \leq |\mathbb{R}|$.
- Define $|A| < |B|$ if $|A| \leq |B|$ but $|A| \neq |B|$.

Theorem 8 (Schröder-Bernstein theorem) *If* $|A| \leq |B|$ *and* $|B| \leq |A|$, *then* $|A| = |B|$.

Cardinality and Infinite Sets

- If $A \subsetneq B$, then $|A| < |B|$?
- If A and B are infinite sets, it is possible that $A \subsetneq B$ yet $|A| = |B|$.
  - $\mathbb{N} \subsetneq \mathbb{Z}$.
  - But $|\,\mathbb{N}\,| = |\,\mathbb{Z}\,|$ (p. 129).[a]
- A lot of "paradoxes."

---

[a] Leibniz (1646–1716) uses it to "prove" that there are no infinite numbers (Russell, 1914).

Galileo's[a] Paradox (1638)

- The squares of positive integers can be placed in one-to-one correspondence with positive integers.
- So they are of the same cardinality.
- But this is contrary to the axiom of Euclid[b] that the whole is greater than any of its proper parts.[c]
- Resolution of paradoxes: Pick the notion that results in "better" mathematics.
- The difference between a mathematical paradox and a contradiction is often a matter of opinions.

[a] Galileo (1564–1642).
[b] Euclid (325 B.C.–265 B.C.).
[c] Leibniz never challenges that axiom (Knobloch, 1999).

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 imagine a hotel with an infinite number of rooms, all of which are occupied.
- A new guest comes and asks for a room.
- "But of course!" exclaims the proprietor.
- He moves the person previously occupying Room 1 to Room 2, the person from Room 2 to Room 3, and so on.
- The new customer now occupies Room 1.

[a]David Hilbert (1862–1943).

Hilbert's Paradox of the Grand Hotel (concluded)

- Now imagine a hotel with an infinite number of rooms, all taken up.
- An infinite number of new guests come in and ask for rooms.
- "Certainly," says the proprietor.
- He moves the occupant of Room 1 to Room 2, the occupant of Room 2 to Room 4, and so on.
- Now all odd-numbered rooms become free and the infinity of new guests can be accommodated in them.[a]

[a] "There are many rooms in my Father's house, and I am going to prepare a place for you." (*John* 14:3)

David Hilbert (1862–1943)