Decidability and Recursive Languages

- Let $L \subseteq (\Sigma - \{\sqcup\})^*$ be a **language**, i.e., a set of strings of symbols with a finite length.
  - For example, $\{0, 01, 10, 210, 1010, \ldots\}$.
- Let M be a TM such that for any string x:
  - If $x \in L$, then $M(x) =$ "yes."
  - If $x \notin L$, then $M(x) =$ "no."
- We say M **decides** L.
- If L is decided by some TM, then L is **recursive**.
  - Palindromes over $\{0, 1\}^*$ are recursive.

Acceptability and Recursively Enumerable Languages

- Let $L \subseteq (\Sigma - \{\bigsqcup\})^*$ be a language.
- Let M be a TM such that for any string x:
  - If $x \in L$, then $M(x) =$ "yes."
  - If $x \notin L$, then $M(x) = \nearrow$.
- We say M **accepts** L.

Acceptability and Recursively Enumerable Languages (concluded)

- If L is accepted by some TM, then L is a **recursively enumerable language**.
  - A recursively enumerable language can be generated by a TM, thus the name.
  - That is, there is an algorithm such that for every $x \in L$, it will be printed out eventually.

Recursive and Recursively Enumerable Languages

Proposition 2 *If L is recursive, then it is recursively enumerable.*

- We need to design a TM that accepts L.
- Let TM M decide L.
- We next modify M's program to obtain M' that accepts L.
- M' is identical to M except that when M is about to halt with a "no" state, M' goes into an infinite loop.
- M' accepts L.

Turing-Computable Functions

- Let $f : (\Sigma - \{\bigsqcup\})^* \to \Sigma^*$.
  - Optimization problems, root finding problems, etc.
- Let M be a TM with alphabet Σ.
- M **computes** f if for any string $x \in (\Sigma - \{\bigsqcup\})^*$, $M(x) = f(x)$.
- We call f a **recursive function**[a] if such an M exists.

[a] Kurt Gödel (1931).

Church's Thesis or the Church-Turing Thesis

- What is computable is Turing-computable; TMs are algorithms (Kleene 1953).
- Many other computation models have been proposed.
  - Recursive function (Gödel), λ calculus (Church), formal language (Post), assembly language-like RAM (Shepherdson & Sturgis), boolean circuits (Shannon), extensions of the Turing machine (more strings, two-dimensional strings, and so on), etc.
- All have been proved to be equivalent.
- No "intuitively computable" problems have been shown not to be Turing-computable (yet).

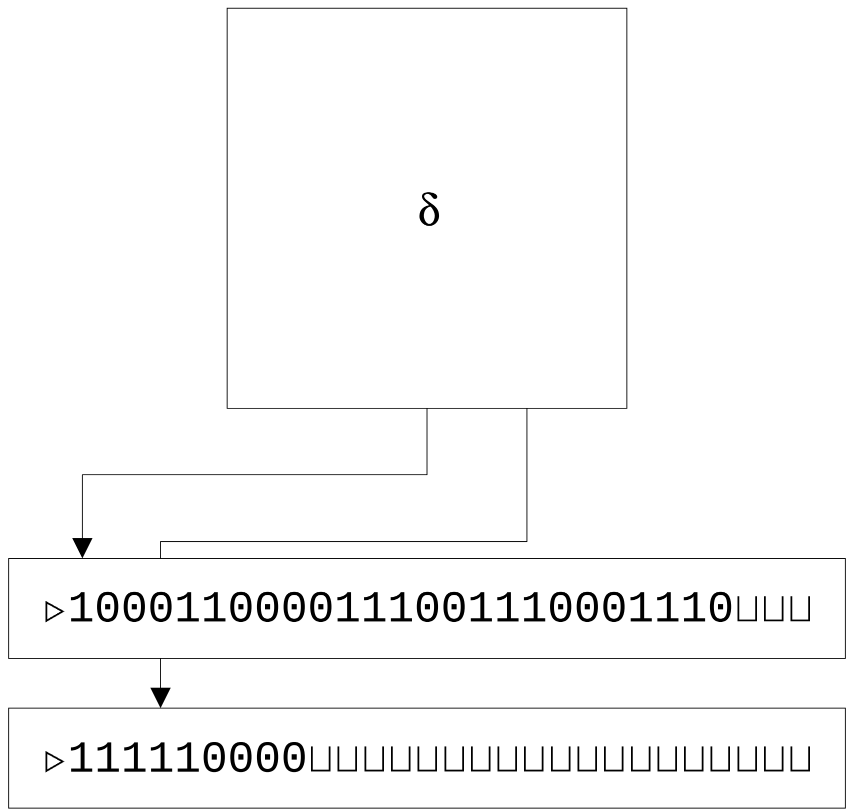
Extended Church's Thesis

- All "reasonably succinct encodings" of problems are *polynomially related*.
  - Representations of a graph as an adjacency matrix and as a linked list are both succinct.
  - The *unary* representation of numbers is not succinct.
  - The *binary* representation of numbers is succinct.
    - 1001 vs. 111111111.
- All numbers for TMs will be binary from now on.

Turing Machines with Multiple Strings

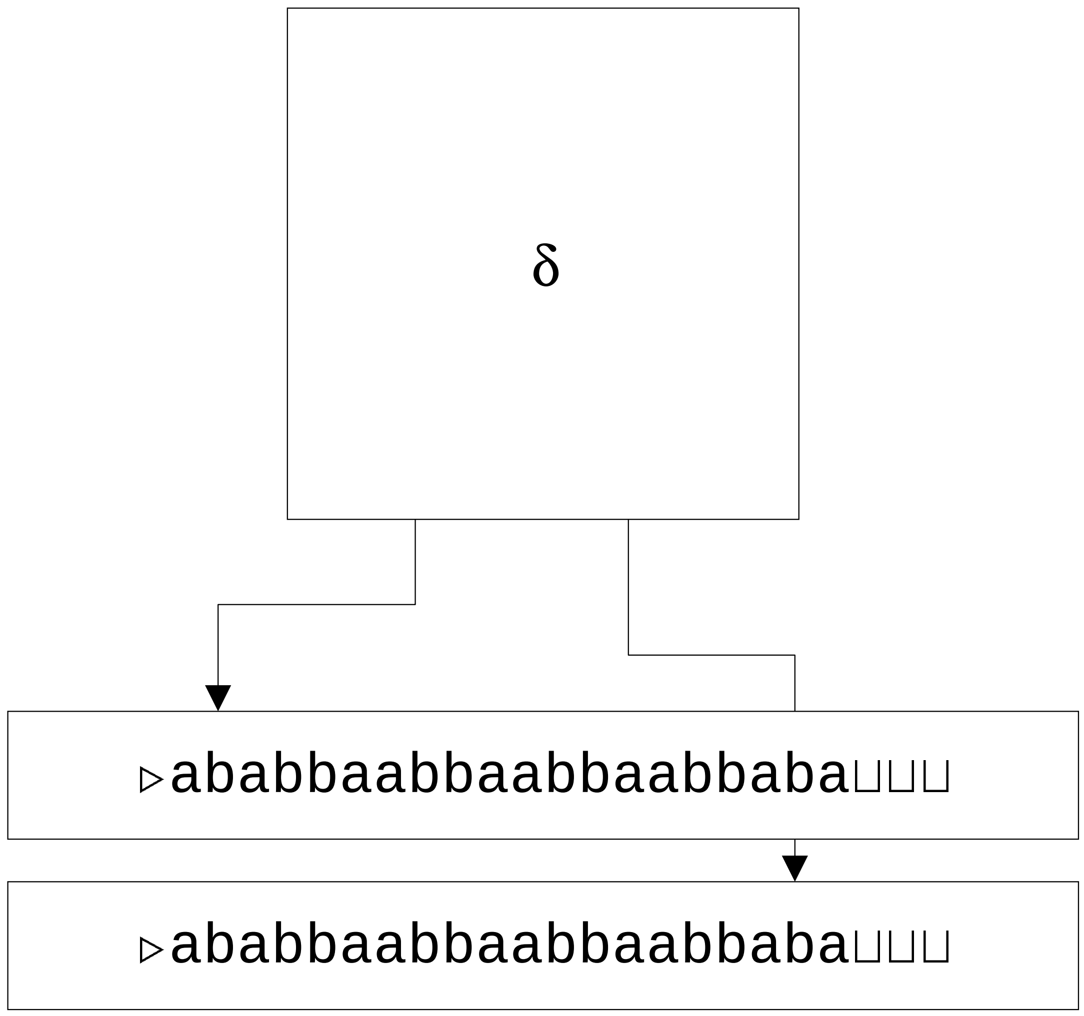
- A k-string Turing machine (TM) is a quadruple $M = (K, \Sigma, \delta, s)$.
- K, Σ, s are as before.
- $\delta : K \times \Sigma^k \to (K \cup \{h, \text{"yes"}, \text{"no"}\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$.
- All strings start with a $\triangleright$.
- The first string contains the input.
- Decidability and acceptability are the same as before.
- When TMs compute functions, the output is on the last (kth) string.

A 2-String TM



PALINDROME Revisited

- A 2-string TM can decide PALINDROME in $O(n)$ steps.
  - It copies the input to the second string.
  - The cursor of the first string is positioned at the first symbol of the input.
  - The cursor of the second string is positioned at the last symbol of the input.
  - The two cursors are then moved in opposite directions until the ends are reached.
  - The machine accepts if and only if the symbols under the two cursors are identical at all steps.


δ
▷ababbaabbaabbaabbaba⊔⊔⊔
▷ababbaabbaabbaabbaba⊔⊔⊔

Configurations and Yielding

- The concept of configuration and yielding is the same as before except that a configuration is a $(2k+1)$-triple

$$(q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k).$$

  - $w_i u_i$ is the ith string.
  - The ith cursor is reading the last symbol of w_i.
  - Recall that $\rhd$ is each w_i's first symbol.

- The k-string TM's initial configuration is

$$(s, \overbrace{\rhd, x, \rhd, \epsilon, \rhd, \epsilon, \ldots, \rhd, \epsilon}^{2k}).$$

Time Complexity

- The multistring TM is the basis of our notion of the time expended by TM computations.
- If for a k-string TM M and input x, the TM halts after t steps, then the **time required by M on input x** is t.
- If $M(x) = \nearrow$, then the time required by M on x is ∞.
- Machine M **operates within time** $f(n)$ for $f : \mathbb{N} \to \mathbb{N}$ if for any input string x, the time required by M on x is at most $f(|x|)$.
  - $|x|$ is the length of string x.
  - Function $f(n)$ is a **time bound** for M.

Time Complexity Classes[a]

- Suppose language $L \subseteq (\Sigma - \{\sqcup\})^*$ is decided by a multistring TM operating in time $f(n)$.
- We say $L \in \text{TIME}(f(n))$.
- $\text{TIME}(f(n))$ is the set of languages decided by TMs with multiple strings operating within time bound $f(n)$.
- $\text{TIME}(f(n))$ is a **complexity class**.
  - PALINDROME is in $\text{TIME}(f(n))$, where $f(n) = O(n)$.

[a] Hartmanis and Stearns (1965); Hartmanis, Lewis, and Stearns (1965).

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 $(q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k)$ of M by configuration

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

  of M'.

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

The Proof (continued)

- The "priming" is to ensure that M' knows which symbol is under the cursor for each simulated string.[a]
- The initial configuration of M' is

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

[a] Added because of comments made by Mr. Che-Wei Chang (R95922093) on September 27, 2006.

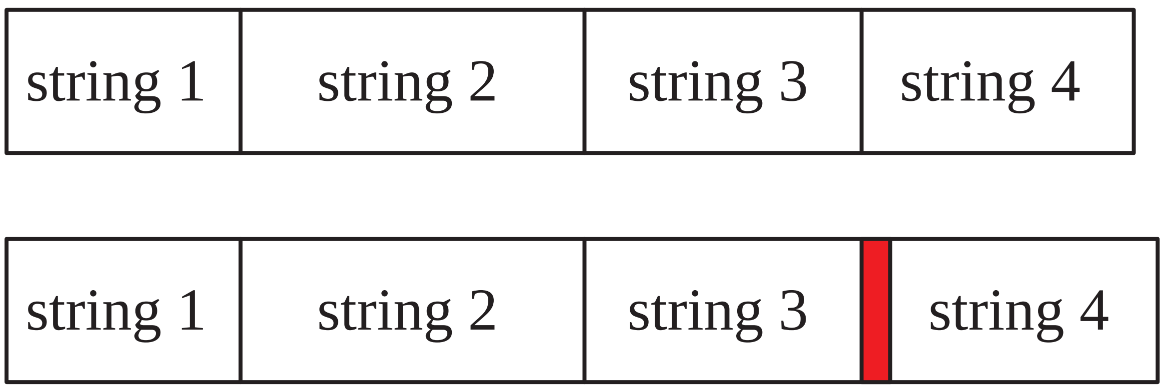
The Proof (continued)

- 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. 30 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 $f(|x|)$.[a]
- The length of the string of M' at any time is $O(kf(|x|))$.

[a] We tacitly assume $f(n) \geq n$.


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[a]

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

[a]Hartmanis and Stearns (1965).

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.

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 $\text{TIME}(n^k)$ for some $k \in \mathbb{N}$.
  - Clearly, $\text{TIME}(n^k) \subseteq \text{TIME}(n^{k+1})$.
- The union of all polynomially decidable languages is denoted by P:

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

- Problems in P can be efficiently solved.

Charging for Space

- We do not 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*.
  - So its 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.
- Assume non-$\sqcup$ is never written over by $\sqcup$.[a]
  - The purpose is not to artificially downplay the space requirement.
- 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 $\sum_{i=1}^{k} |w_i u_i|$.

---

[a]Corrected by Ms. Chuan-Ju Wang (R95922018) on September 27, 2006.

Space Complexity (concluded)

- 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 $f : \mathbb{N} \rightarrow \mathbb{N}$ if for any input x, the space required by M on x is at most $f(|\, x\,|)$.

Space Complexity Classes

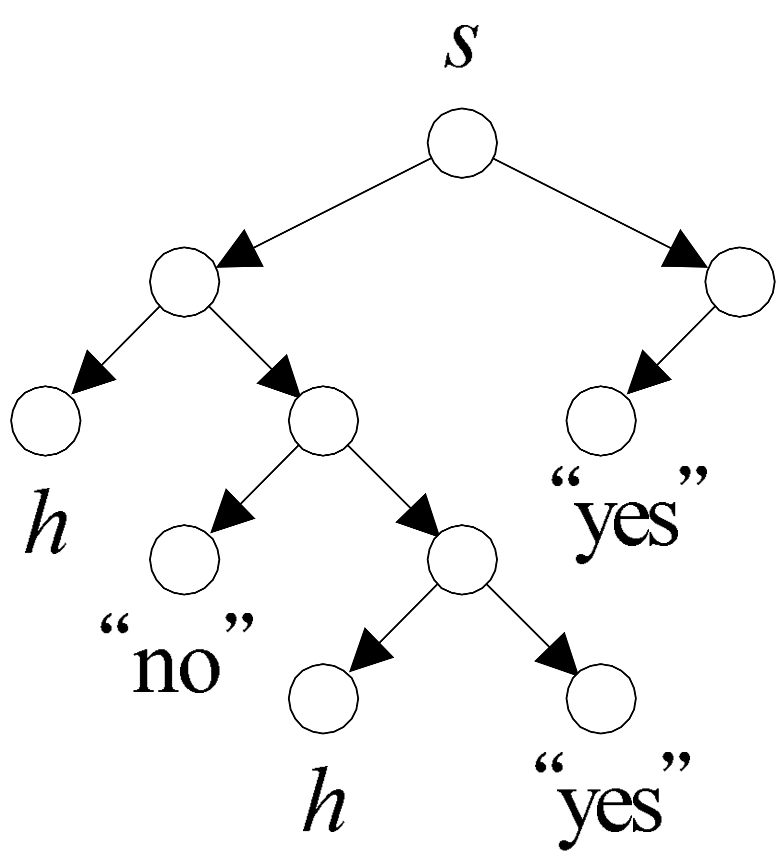
- Let L be a language.
- Then
$$L \in \text{SPACE}(f(n))$$
if there is a TM with input and output that decides L and operates within space bound $f(n)$.
- SPACE$(f(n))$ is a set of languages.
  - PALINDROME $\in$ SPACE$(\log n)$: Keep 3 counters.
- As in the linear speedup theorem (Theorem 4), constant 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 and 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.
  - If $x \notin L$, no nondeterministic choices should lead to a "yes" state.
- What is key is the algorithm's overall behavior not whether it gives a correct answer for each particular run.
- Determinism is a special case of nondeterminism.

An Example

- Let L be the set of logical conclusions of a set of axioms.
  - Predicates not in L may be false under the axioms.
  - They may also be independent of the axioms.
    - That is, they can be assumed true or false without contradicting the axioms.

An Example (concluded)

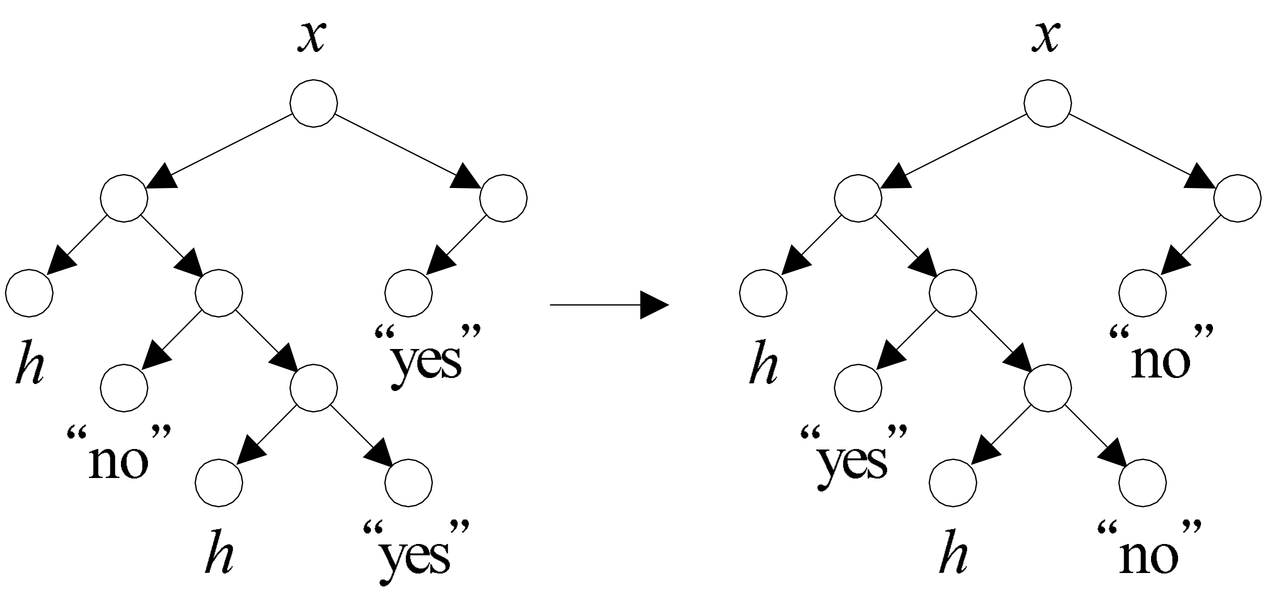
- Let ϕ be a predicate whose validity we would like to prove.
- Consider the nondeterministic algorithm:

  $b :=$ `true`;
  while the input predicate $\phi \neq b$ **do**
      Generate a logical conclusion of b by applying some of the axioms; {Nondeterministic choice.}
      Assign this conclusion to b;
  end while
  "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 (see next page).
  - When this happens, M and M' accept languages that are not complements of each other.



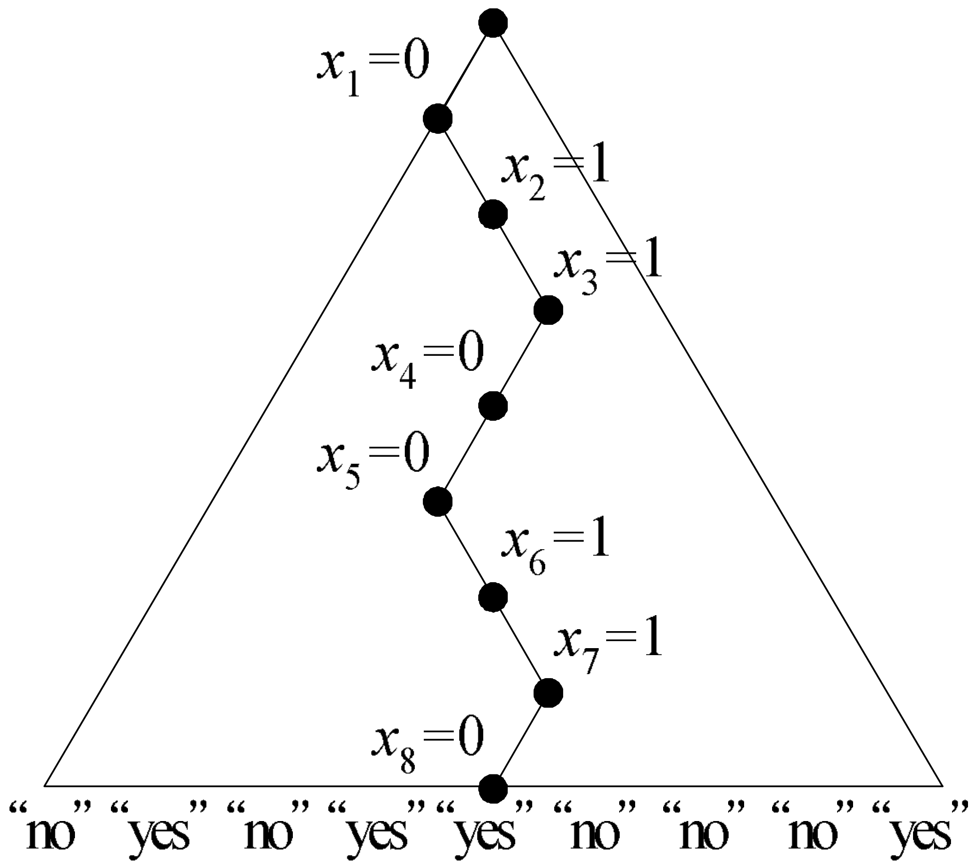
x
h
"no"
h
"yes"
"yes"
x
h
"yes"
h
"no"
"no"

A Nondeterministic Algorithm for Satisfiability

ϕ is a boolean formula with n variables.

```
for i = 1, 2, ..., n do
    Guess x_i ∈ {0, 1}; {Nondeterministic choice.}
end for
{Verification:}
if φ(x_1, x_2, ..., 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 equally hard (p. 313 and p. 391).

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 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

---

[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.

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
$$\mathrm{NP} = \bigcup_{k>0} \mathrm{NTIME}(n^k).$$
- Clearly $\mathrm{P} \subseteq \mathrm{NP}$.
- Think of NP as efficiently *verifiable* problems.
  - Boolean satisfiability (SAT).
  - TSP (D).
- The most important open problem in computer science is whether $\mathrm{P} = \mathrm{NP}$.

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 (but M does *not* know $f(n)$).
  - As M is time-bounded, the depth-first search will not run indefinitely.

The Proof (concluded)

- 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.

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

NTIME vs. TIME

- Does converting an NTM into a TM require exploring all the computation paths of the NTM as done in Theorem 5 (p. 88)?
- This is the most important question in theory with practical implications.

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 4 on p. 66), constant coefficients do not matter.

Graph Reachability

- Let $G(V, E)$ be a directed graph (digraph).
- REACHABILITY asks if, 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 the nondeterministic space complexity?

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

$x_1 := a$; {Assume $a \neq b$.}
for $i = 2, 3, \ldots, n$ **do**
    Guess $x_i \in \{ v_1, v_2, \ldots, v_n \}$; {The ith node.}
end for
for $i = 2, 3, \ldots, n$ **do**
    if $(x_{i-1}, x_i) \notin E$ **then**
        "no";
    end if
    if $x_i = b$ **then**
        "yes";
    end if
end for
"no";

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

$x := a;$
for $i = 2, 3, \ldots, n$ **do**
    Guess $y \in \{2, 3, \ldots, n\}$; {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 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 (p. 181).

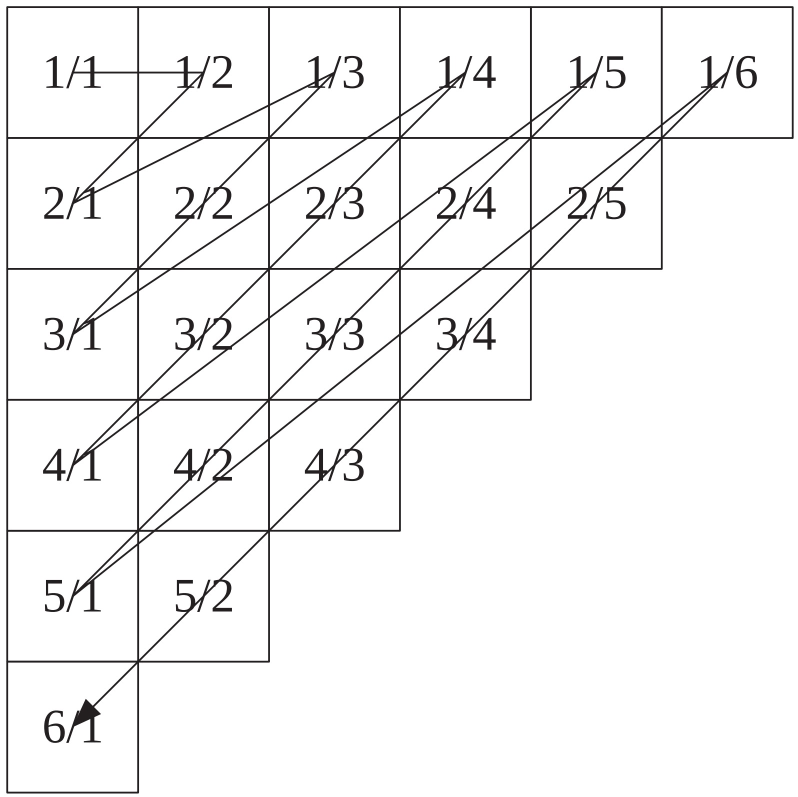
Undecidability

It seemed unworthy of a grown man
to spend his time on such trivialities,
but what was I to do?
— Bertrand Russell (1872–1970),
Autobiography, Vol. I

Infinite Sets

- A set is **countable** if it is finite or if it can be put in one-one correspondence with $\mathbb{N}$, 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 $\mathbb{Z}^+$: $i - 1 \leftrightarrow i$.
  - Set of odd integers: $(i-1)/2 \leftrightarrow i$.
  - Set of rational numbers: 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, written as
$$|A| = |B| \quad \text{or} \quad A \sim B,$$
if there exists a one-to-one correspondence between their elements.
- 2^A denotes set A's **power set**, that is $\{B : B \subseteq A\}$.
  - If $|A| = k$, then $|2^A| = 2^k$.
  - So $|A| < |2^A|$ when A is finite.

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 (p. 98).
- 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 of 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[b] that the whole is greater than any of its proper parts.
- 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 opinion.

[a] Galileo (1564–1642).
[b] Euclid (325 B.C.–265 B.C.).