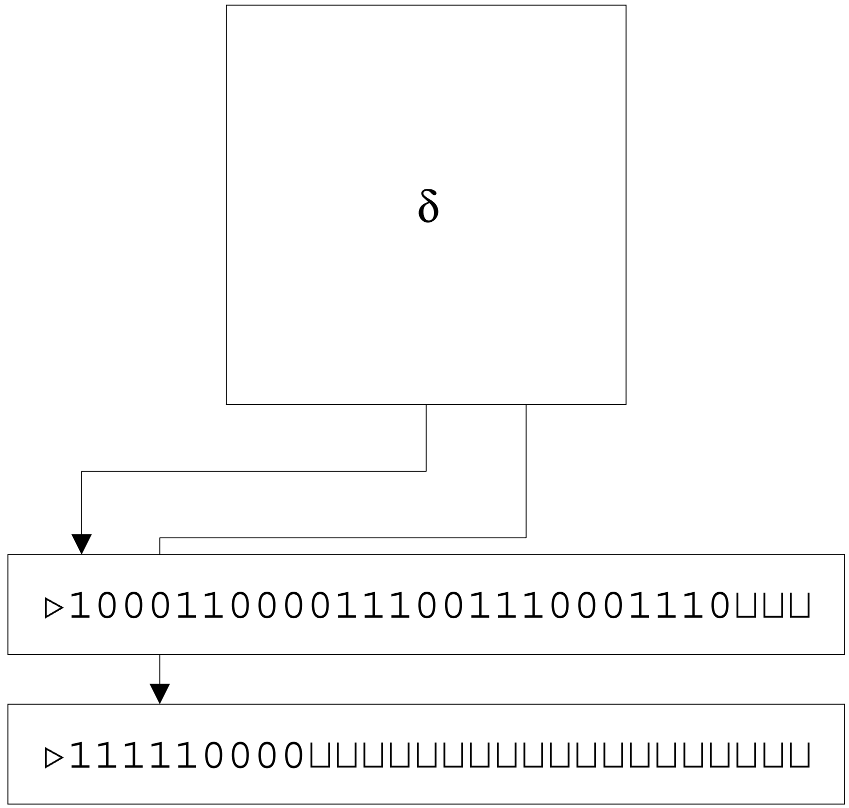
Turing Machines with Multiple Strings

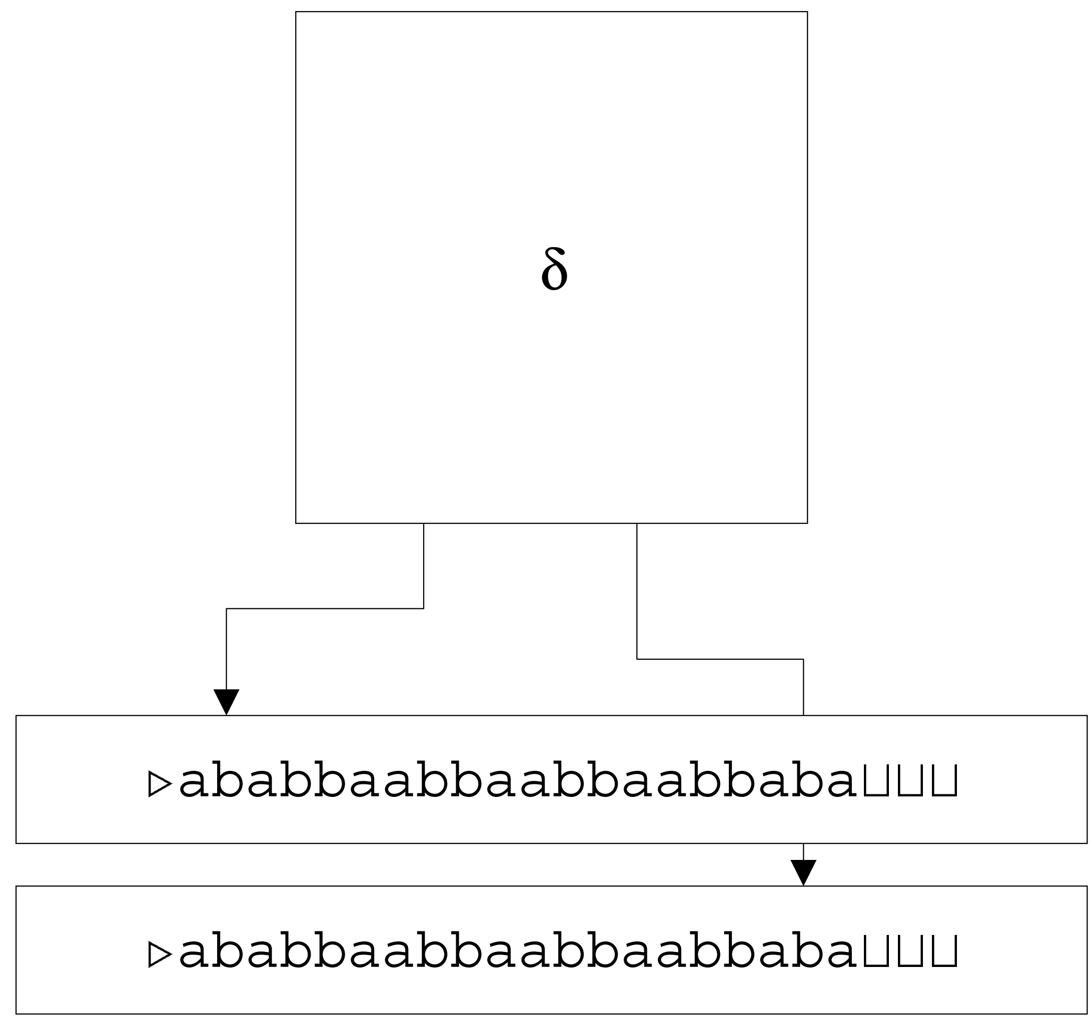
- A k-string Turing machine (TM) is a quadruple $M = (K, \Sigma, \delta, s)$.
- K, Σ, s are as before.
- $\delta : K \times \Sigma^k \to (K \cup \{h, \text{“yes”}, \text{“no”}\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$.
- All strings start with a $\triangleright$.
- The first string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

PALINDROME Revisited (concluded)

- The running times of a 2-string TM and a single-string TM are quadratically related: n^2 vs. n.
- This is consistent with the extended Church's thesis.[a]
  - "Reasonable" models are related polynomially in running times.

---

[a]Recall p. 68.

Configurations and Yielding

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

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

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

- The k-string TM's initial configuration is

$$(s, \overbrace{\underbrace{\triangleright, x}_{1}, \underbrace{\triangleright, \epsilon}_{2}, \underbrace{\triangleright, \epsilon}_{3}, \ldots, \underbrace{\triangleright, \epsilon}_{k}}^{2k}).$$

Time seemed to be
the most obvious measure
of complexity.
— Stephen Arthur Cook (1939–)

Time Complexity

- The multistring TM is the basis of our notion of the time expended by TMs.
- If a k-string TM M halts after t steps on input x, then the **time required by M on input x** is t.
- If $M(x) = \nearrow$, then the time required by M on x is ∞.

Time Complexity (concluded)

- 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 $f(n) = O(n)$.
- Trivially, $\text{TIME}(f(n)) \subseteq \text{TIME}(g(n))$ if $f(n) \leq g(n)$ for all n.

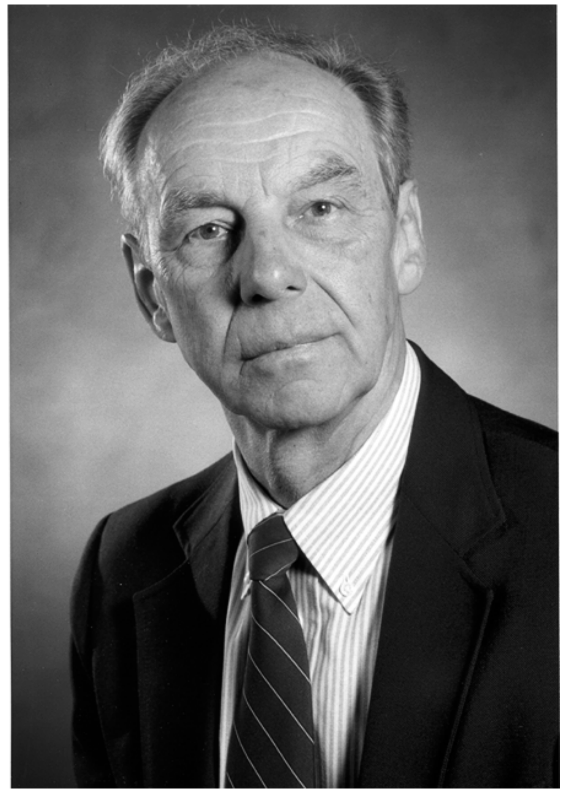
---

[a] Rabin (1963); Hartmanis & Stearns (1965); Hartmanis, Lewis, & Stearns (1965).

Michael O. Rabin[a] (1931–)

[a]Turing Award (1976).

Juris Hartmanis[a] (1928–)



[a] Turing Award (1993).

Richard Edwin Stearns[a] (1936–)

[a]Turing Award (1993).

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.

The Proof

- Represent configuration $(q, w_1, u_1, w_2, u_2, \ldots, w_k, u_k)$ of M by this configuration of M':

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

  - $\lhd$ is a special delimiter.
  - w_i' is w_i with the first[a] and last symbols "primed."
  - It serves the purpose of "," in a configuration.[b]

---

[a]The first symbol is of course $\rhd$.

[b]An alternative is to use $(q, \rhd w_1' | u_1 \lhd w_2' | u_2 \lhd \cdots \lhd w_k' | u_k \lhd \lhd)$ by priming only $\rhd$ in w_i, where "|" is a new symbol.

The Proof (continued)

- The first symbol of w_i' is the primed version of $\triangleright$: $\triangleright'$.
  - Cursors are not allowed to move to the left of $\triangleright$.[a]
  - So the cursor of M' can move *between* the simulated strings of M.[b]
- The "priming" of the last symbol of each w_i ensures that M' knows which symbol is under each cursor of M.[c]

---

[a] Recall p. 24.

[b] Thanks to a lively discussion on September 22, 2009.

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

The Proof (continued)

- The initial configuration of M' is

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

  - $\rhd''$ is double-primed because it is the beginning and the ending symbol as the cursor is reading it.[a]
  - Again, think of it as a new symbol.

[a] Added after the class discussion on September 20, 2011.

The Proof (continued)

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

---

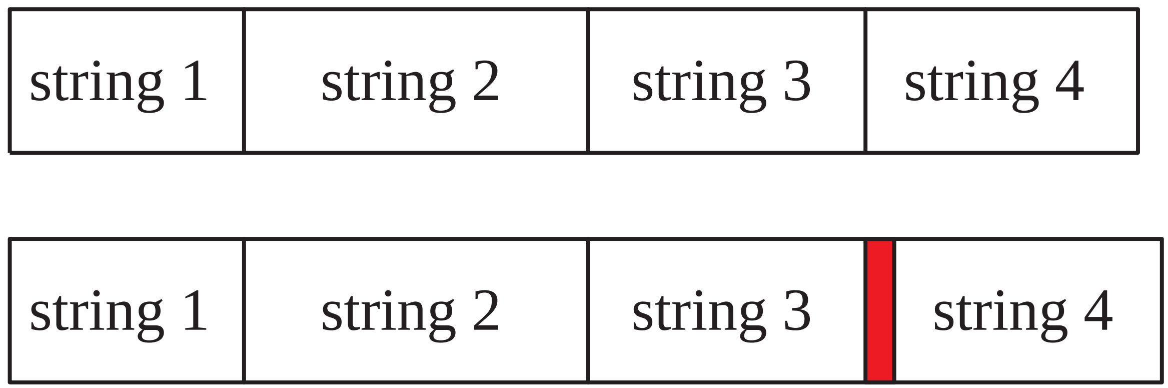
[a]Recall the TM program on p. 36.

The Proof (continued)

- It is possible that some strings of M need to be lengthened (see next page).
  - The linear-time algorithm on p. 39 can be used for each such string.
- The simulation continues until M halts.
- M' then erases all strings of M except the last one.[a]

[a] Whatever remains on the tape of M' before the first $\sqcup$ is considered output by our convention. So $\triangleright'$s and $\triangleright''$s must be removed.

The Proof (continued)[a]



---

[a]If we interleave the strings, the simulation may be easier. Contributed by Mr. Kai-Yuan Hou (`B99201038`, `R03922014`) on September 22, 2015. This is similar to constructing a single-string *multi-track* TM in, e.g., Hopcroft & Ullman (1969). Or one may do the insertion starting from the last string by memorizing what needs to be inserted for each string. Contributed by Mr. Hsi-Kang Hsu (`R10922128`) on September 30, 2021.

The Proof (continued)

- 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 $O(kf(|x|))$.
- Simulating each step of M takes, *per string of M*, $O(kf(|x|))$ steps.
  - $O(f(|x|))$ steps to collect information from this string.
  - $O(kf(|x|))$ steps to write and, if needed, to lengthen the string.

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

The Proof (concluded)

- There are k strings.
- So 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 $O(k^2 f(|x|)^2)$.[a]

---

[a]Is the time reduced to $O(kf(|x|)^2)$ if the interleaving data structure is adopted?

Simulation with Two-String TMs

We can do better with two-string simulating TMs.

Theorem 4 *Given any* k*-string* M *operating within time* $f(n)$, $k > 2$, *there exists a two-string* M' *operating within time* $O(f(n) \log f(n))$ *such that* $M(x) = M'(x)$ *for any input* x.

Linear Speedup[a]

Theorem 5 *Let $L \in TIME(f(n))$. Then for any $\epsilon > 0$, $L \in TIME(f'(n))$, where $f'(n) \triangleq \epsilon f(n) + n + 2$.*

See Theorem 2.2 of the textbook for a proof.

---

[a]Hartmanis & Stearns (1965).

Proof Ideas

- Take the TM program on p. 36.
- It accepts if and only if the input contains two consecutive 1's.
- Assume $M = (K, \Sigma, \delta, s)$, where
  $K = \{ s', s_{00}, s_{01}, s_{10}, s_{11}, \ldots, \text{"yes"}, \text{"no"} \}$,
  $\Sigma = \{ 0, 1, (00), (01), (10), (11), (0\sqcup), (1\sqcup), \sqcup, \triangleright \}$.

Proof Ideas (continued)

- First convert the input into 2-tuples onto the second string.
- So $\overbrace{10011001110}^{11}$ becomes $\overbrace{(10)(01)(10)(01)(11)(0\sqcup)}^{6}$.
- The length is therefore about halved.
- The transition table below covers only the second string for brevity.
- It presents only the key lines of code.

Proof Ideas (continued)

$p \in K$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
$\vdots$	$\vdots$	$\vdots$
s'	(00)	$(s', (00), \rightarrow)$
s'	(01)	$(s_{01}, (01), \rightarrow)$
s'	(10)	$(s', (10), \rightarrow)$
s'	(11)	$(\text{"yes"}, (11), -)$
s'	$(0\sqcup)$	$(\text{"no"}, (0\sqcup), -)$
s'	$(1\sqcup)$	$(\text{"no"}, (1\sqcup), -)$
s'	$\sqcup$	$(\text{"no"}, \sqcup, -)$

Proof Ideas (concluded)[a]

s_{01}	(10)	$(\text{“yes”}, (10), -)$
s_{01}	(11)	$(\text{“yes”}, (11), -)$
s_{01}	(01)	$(s_{01}, (01), \rightarrow)$
s_{01}	(00)	$(s', (00), \rightarrow)$
s_{01}	$(0\sqcup)$	$(\text{“no”}, (1\sqcup), -)$
s_{01}	$(1\sqcup)$	$(\text{“yes”}, (1\sqcup), -)$
s_{01}	$\sqcup$	$(\text{“no”}, \sqcup, -)$
$\vdots$	$\vdots$	$\vdots$

[a]Corrected by Mr. Yu-Ming Lu (R06723032, D08922008) on September 30, 2021.

Implications of the Speedup Theorem

- State size can be traded for speed.[a]
- If the running time is cn with $c > 1$, then c can be made arbitrarily close to 1.
- If the running time is superlinear, say $14n^2 + 31n$, then the constant in the leading term (14 in this example) can be made arbitrarily small.
  - *Arbitrary* linear speedup can be achieved.[b]
  - This justifies the big-O notation in the analysis of algorithms.

---

[a] $m^k \cdot |\Sigma|^{3mk}$-fold increase to gain a speedup of $O(m)$. No free lunch.

[b] Can you apply the theorem multiple times to achieve superlinear speedup? Thanks to a question by a student on September 21, 2010.

P

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

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

- P contains problems that can be efficiently solved.

[a]Cobham (1964).

Philosophers have explained space.
They have not explained time.
— Arnold Bennett (1867–1931),
How To Live on 24 Hours a Day (1910)

I keep bumping into that silly quotation
attributed to me that says
640K of memory is enough.
— Bill Gates (1996)

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 reduce the space needs (see below).
- 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`, `F95922018`) on September 27, 2006.

Space Complexity (continued)

- Suppose 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*.[a]
  - The cursor on the last string never moves to the left.
    * The output string is essentially *write-only*.
  - The cursor of the input string does not go beyond the first $\sqcup$.

[a]Called an **off-line TM** in Hartmanis, Lewis, & Stearns (1965).

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 $f : \mathbb{N} \to \mathbb{N}$ 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 $\in$ SPACE$(\log n)$.[a]
- A linear speedup theorem similar to the one on p. 97 exists, so constant coefficients do not matter.

[a]Maintain 3 counters.

If she can hesitate as to "Yes,"
she ought to say "No" directly.
— Jane Austen (1775–1817),
Emma (1815)

Nondeterminism[a]

- A **nondeterministic Turing machine** (**NTM**) is a quadruple $N = (K, \Sigma, \Delta, s)$.
- K, Σ, s are as before.
- $\Delta \subseteq K \times \Sigma \times (K \cup \{h, \text{"yes"}, \text{"no"}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ is a *relation*, not a *function*.[b]
  - For each state-symbol combination (q, σ), there may be *multiple* valid next steps.
  - Multiple lines of code may be applicable.
  - But only one will be taken.

[a]Rabin & Scott (1959).

[b]Corrected by Mr. Jung-Ying Chen (D95723006) on September 23, 2008.

Nondeterminism (continued)

- As before, a program contains lines of code:

$$\begin{array}{rcl} (q_1, \sigma_1, p_1, \rho_1, D_1) & \in & \Delta, \\ (q_2, \sigma_2, p_2, \rho_2, D_2) & \in & \Delta, \\ & \vdots & \\ (q_n, \sigma_n, p_n, \rho_n, D_n) & \in & \Delta. \end{array}$$

- But we cannot write

$$\delta(q_i, \sigma_i) = (p_i, \rho_i, D_i)$$

as in the deterministic case[a] anymore.

---

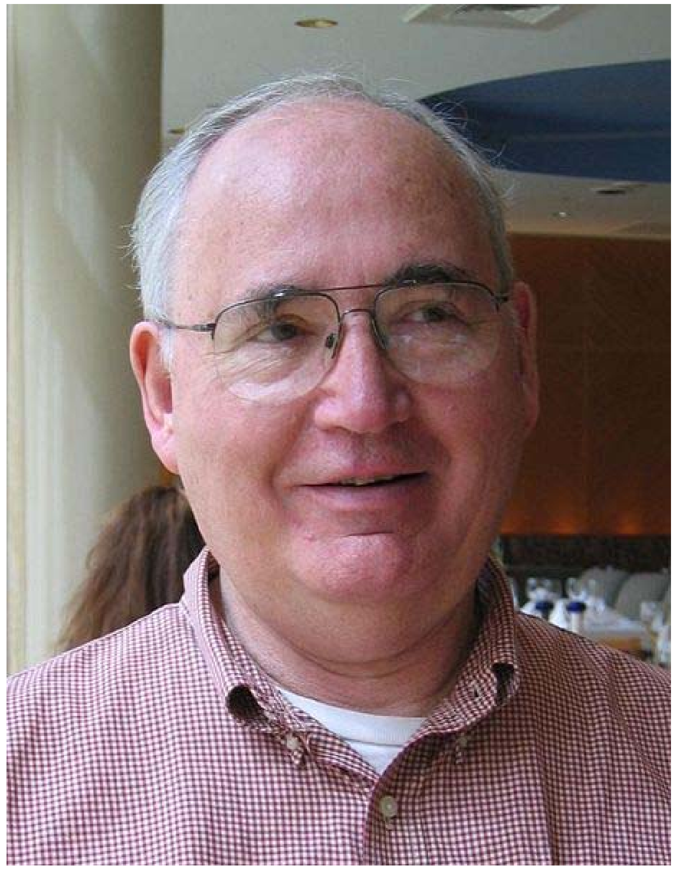
[a]Recall p. 25.

Nondeterminism (concluded)

- A configuration yields another configuration in one step if there *exists* a rule in Δ that makes this happen.
- There remains only one thread of computation.[a]
  - Nondeterminism is *not* parallelism, multiprocessing, multithreading, or quantum computation.

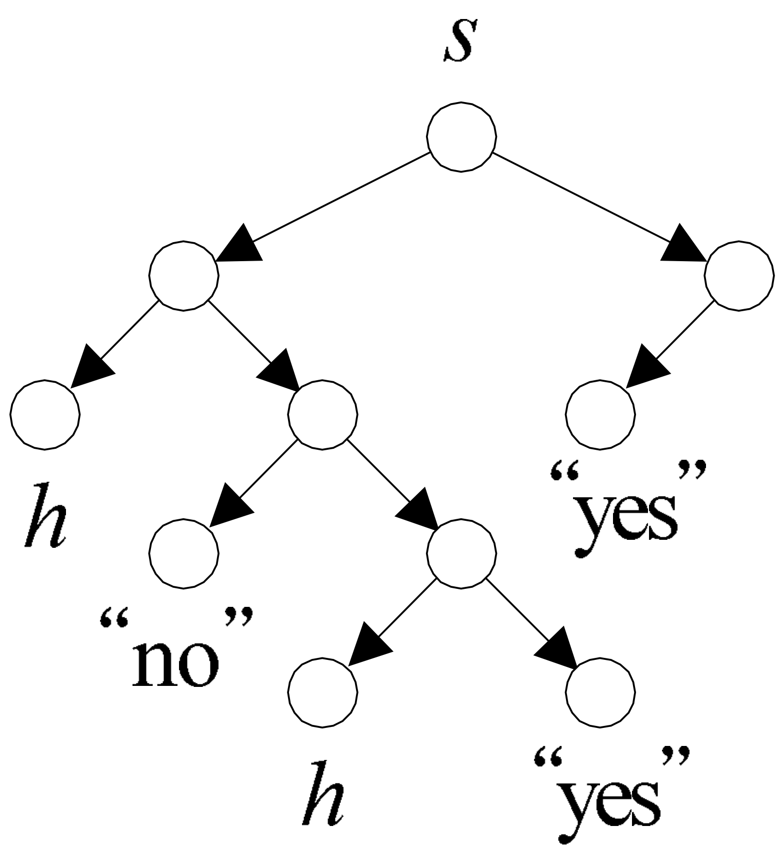
[a] Thanks to a lively discussion on September 22, 2015.

Dana Stewart Scott[a] (1932–)



[a]Turing Award (1976).

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."
- 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 (continued)

- It is not required that the deciding 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] Unlike the deterministic case (p. 53). So "accepts" may be a more proper term. Some books use "decides" only when the NTM always halts.

Decidability under Nondeterminism (concluded)

- For example, suppose L is the set of primes.[a]
- Then we have the primality testing problem.
- An NTM N decides L if:
  - If x is a prime, then $N(x) =$ "yes" for some computation path.
  - If x is not a prime, then $N(x) \neq$ "yes" for all computation paths.

---

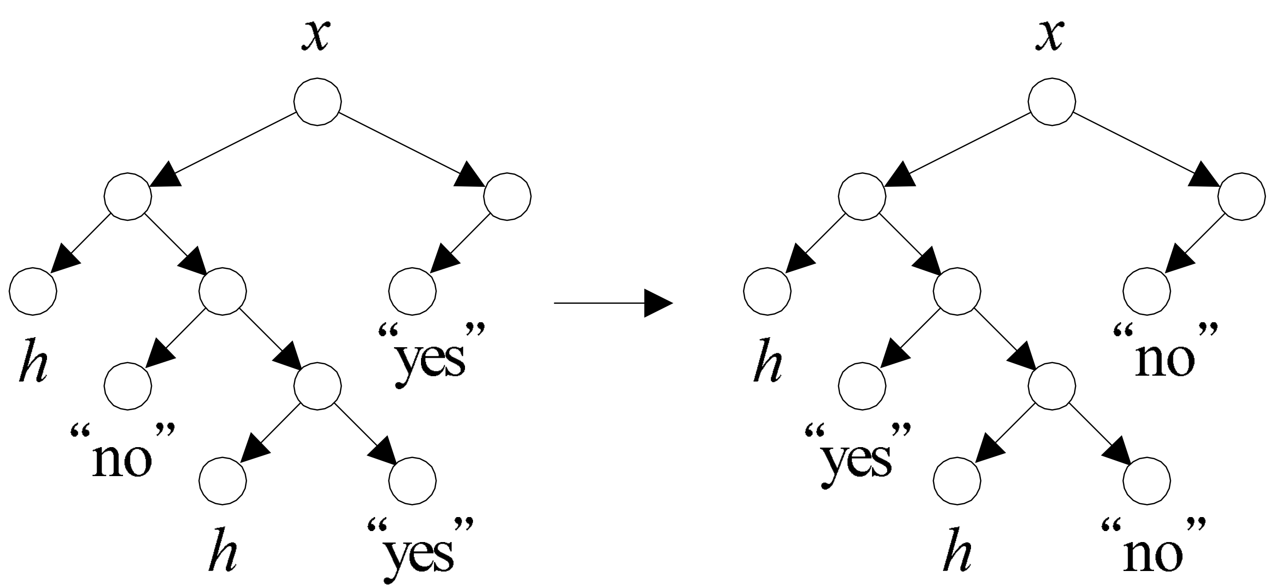
[a]Contributed by Mr. Yu-Ming Lu (`R06723032`, `D08922008`) on March 7, 2019.

Complementing a TM's Halting States

- Let M decide L, and M' be M after "yes" $\leftrightarrow$ "no".
- If M is *deterministic*, then M' decides $\bar{L}$.[a]
  - So M and M' decide languages that complement each other.
- But if M is an NTM, then M' may not decide $\bar{L}$.
  - It is possible that M and M' accept the same input x (see next page).
  - So M and M' may accept languages that are *not* even disjoint.

---

[a]By the definition on p. 53, M must halt on all inputs.

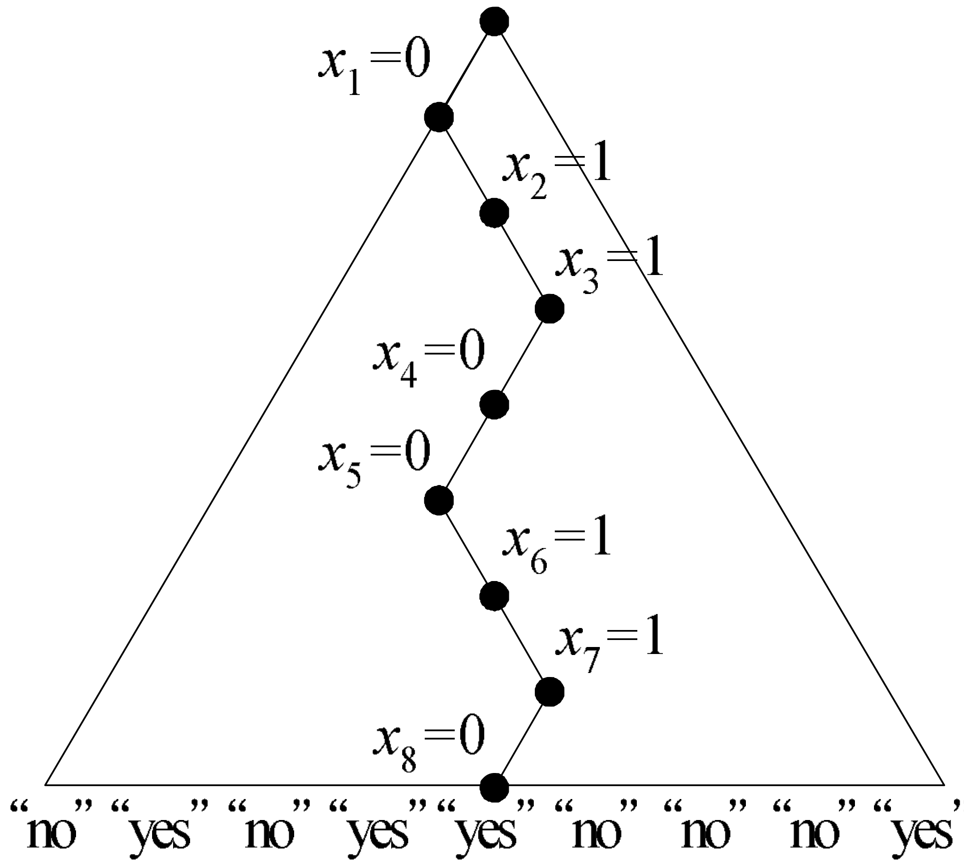

x
h
"no"
h
"yes"
"yes"
x
h
"yes"
h
"no"
"no"

A Nondeterministic Algorithm for Satisfiability

ϕ is a boolean formula with n variables.

for $i = 1, 2, \ldots, n$ **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

- Recall that ϕ is satisfiable if and only if there is a truth assignment that satisfies ϕ.
- Think of the computation tree as a complete binary tree of depth n.
- Every computation path corresponds to a particular truth assignment[a] out of 2^n.

[a]Equivalently, a sequence of nondeterministic choices.

Analysis (concluded)

- The algorithm decides language

$$\{\,\phi : \phi \text{ is satisfiable}\,\}.$$

  - Suppose ϕ is satisfiable.
    - 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

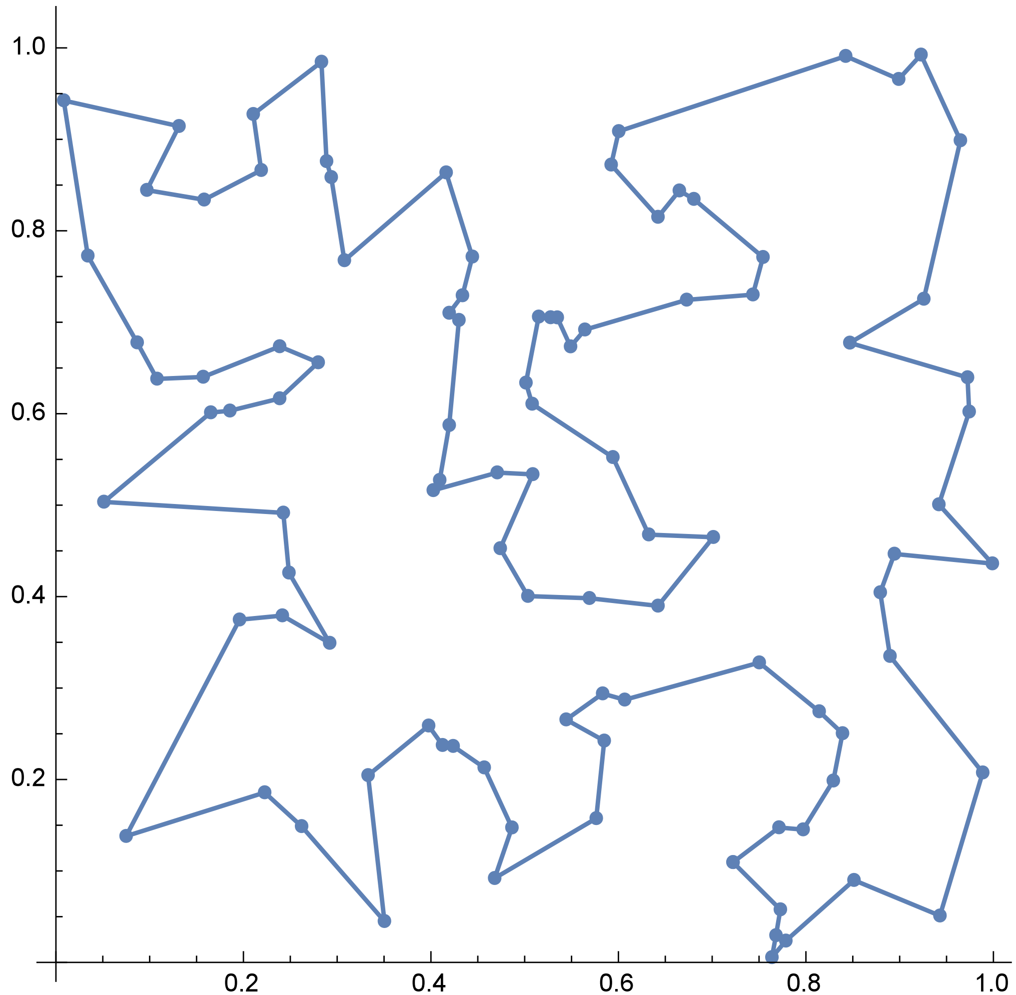
- We are given n cities $1, 2, \ldots, n$ and integer distance d_{ij} between any two cities i and j.
- Assume $d_{ij} = d_{ji}$ for convenience.
- The **traveling salesman problem** (TSP) asks for the total distance of the shortest tour of the cities.[a]
- The decision version TSP (D) asks if there 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. They are equally hard (pp. 419 and 522).

A Shortest Tour



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
{Verification:}
if $x_1, x_2, \ldots, x_n$ are distinct and $\sum_{i=1}^{n-1} d_{x_i, x_{i+1}} \leq B$ **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 merely requires that such a computation path (i.e., a sequence of nondeterministic choices) exists.

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
$$\mathrm{NP} \stackrel{\Delta}{=} \bigcup_{k>0} \mathrm{NTIME}(n^k).$$
- Clearly $\mathrm{P} \subseteq \mathrm{NP}$.
- Think of NP as efficiently *verifiable* problems.[a]
  - Boolean satisfiability (pp. 120 and 203), e.g.
- The most important open problem in computer science is whether $\mathrm{P} = \mathrm{NP}$.

---
[a]See p. 347.

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 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.

Simulating Nondeterministic TMs

Nondeterminism does not add power to TMs.[a]

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 explores the computation tree of $N(x)$ using depth-first search.
  - M does *not* need to know $f(n)$.
  - As N is time-bounded, the depth-first search will halt.[b]

[a] Like finite-state automata, but unlike pushdown automata.

[b] If there is no time bound, breadth-first search is safer.

The Proof (concluded)

- If any path leads to "yes," then M immediately enters the "yes" state.
- If none of the paths lead 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 $\mathrm{NTIME}(f(n)) \subseteq \bigcup_{c>1} \mathrm{TIME}(c^{f(n)})$.[a]

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

NTIME vs. TIME

- Does converting an NTM into a TM *require* exploring all computation paths of the NTM in the worst case as done in Theorem 6 (p. 132)?
- This is a key question in theory with important 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,[a] constant coefficients do not matter.

[a]Theorem 5 (p. 97).

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 $(x_{i-1}, x_i) \notin E$ **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 ≤ n.}
x := a;
for i = 2, 3, ..., m do
    Guess y ∈ { v_1, v_2, ..., v_m }; {The next node.}
    if (x, y) ∉ E then
        "no";
    end if
    if y = b then
        "yes";
    end if
    x := y; {Recycle the space.}
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

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

  - REACHABILITY with more than one terminal node also has the same complexity.
  - In fact, REACHABILITY for *undirected* graphs is in SPACE$(\log n)$.[a]
- It is well-known that REACHABILITY $\in$ P.[b]

---

[a] Reingold (2004).

[b] See, e.g., p. 248.