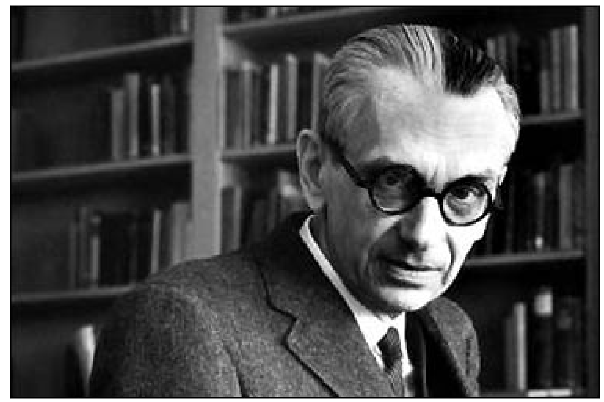
Turing-Computable Functions

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

[a] Kurt Gödel (1931, 1934).

Kurt Gödel[a] (1906–1978)

Quine (1978), "this theorem [· · ·] sealed his immortality."



[a] This photo was taken by Alfred Eisenstaedt (1898–1995).

Church's Thesis or the Church-Turing Thesis

- What is computable is Turing-computable; TMs are algorithms.[a]
- No "intuitively computable" problems have been shown not to be Turing-computable, yet.[b]

---

[a] Church (1935); Kleene (1953).

[b] Quantum computer of Manin (1980) and Feynman (1982) and DNA computer of Adleman (1994).

Church's Thesis or the Church-Turing Thesis (concluded)

- 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.

Alonso Church (1903–1995)

Extended Church's Thesis[a]

- All "reasonably succinct encodings" of problems are *polynomially related* (e.g., n^2 vs. n^6).
  - 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_2 vs. 111111111_1.
- All numbers for TMs will be binary from now on.

[a]Some call it "polynomial Church's thesis," which Lószló Lovász attributed to Leonid Levin.

Extended Church's Thesis (concluded)

- Representations that are not succinct may give misleadingly low complexities.
  - Consider an algorithm with binary inputs that runs in 2^n steps.
  - Suppose the input uses unary representation instead.
  - Then the same algorithm runs in linear time because the input length is now 2^n!
- So a succinct representation means honest accounting.

Physical Church-Turing Thesis

- The **physical Church-Turing thesis** states that:

  Anything computable in physics can also be computed on a Turing machine.[a]

- The universe is a Turing machine.[b]

---

[a]Cooper (2012).

[b]Edward Fredkin's (1992) controversial digital physics.

The Strong Church-Turing Thesis[a]

- The **strong Church-Turing thesis** states that:

  A Turing machine can compute *any* function computable by any "reasonable" physical device with only polynomial slowdown.[b]

- A CPU, a GPU, and a DSP chip are good examples of physical devices.[c]

---

[a] Vergis, Steiglitz, & Dickinson (1986).

[b] `http://ocw.mit.edu/courses/mathematics/18-405j-advanced-complexity-theory-fall-2001/lecture-notes/lecture10.pdf`

[c] Thanks to a lively discussion on September 23, 2014.

The Strong Church-Turing Thesis (concluded)

- Factoring is believed to be a hard problem for Turing machines (but there is no proof yet).
- But a quantum computer can factor numbers in probabilistic polynomial time.[a]
- So if a large-scale quantum computer can be reliably built, the strong Church-Turing thesis may be refuted.[b]

---

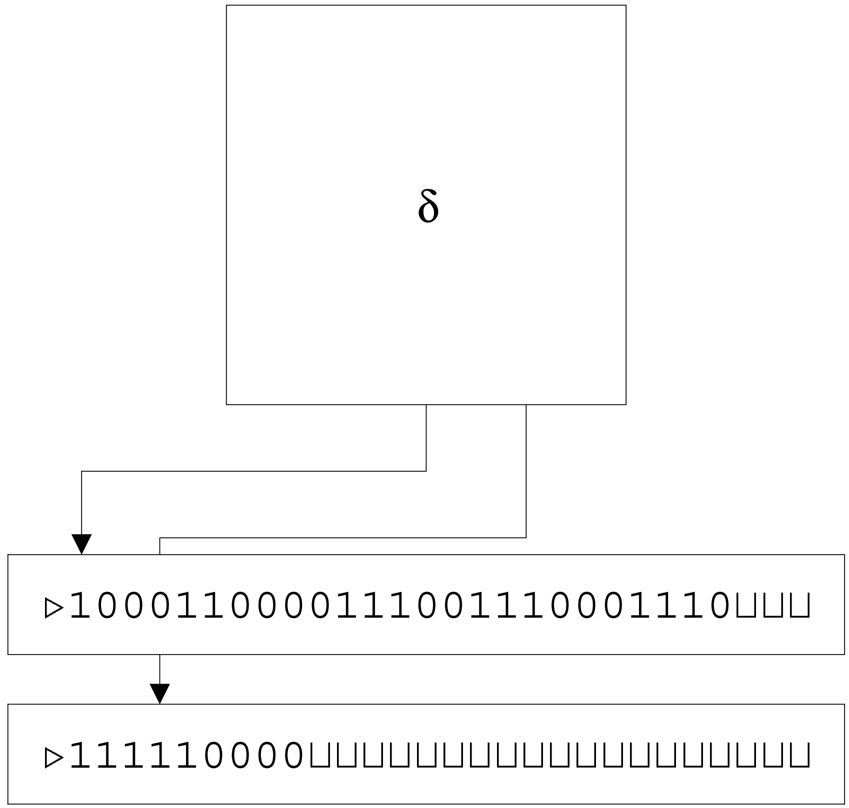
[a] Shor (1994).

[b] Contributed by Mr. Kai-Yuan Hou (B99201038, R03922014) on September 22, 2015.

Turing Machines with Multiple Strings

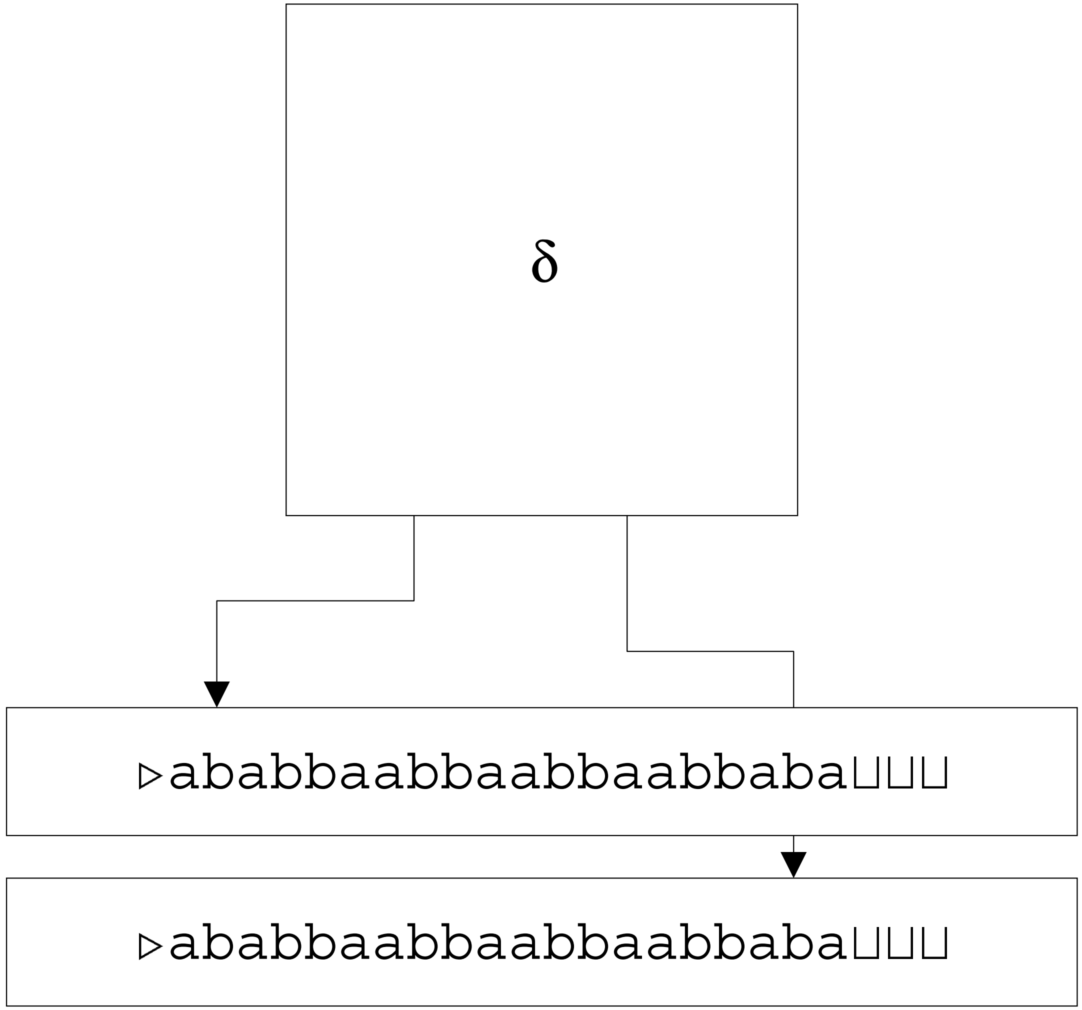
- A k-string Turing machine (TM) is a quadruple $M = (K, \Sigma, \delta, s)$.
- K, Σ, s are as before.
- $\delta : K \times \Sigma^k \to (K \cup \{h, \text{“yes”}, \text{“no”}\}) \times (\Sigma \times \{\leftarrow, \to, -\})^k$.
- All strings start with a $\triangleright$.
- The first string contains 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.
  - "Reasonable" models are related polynomially in running times.

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)

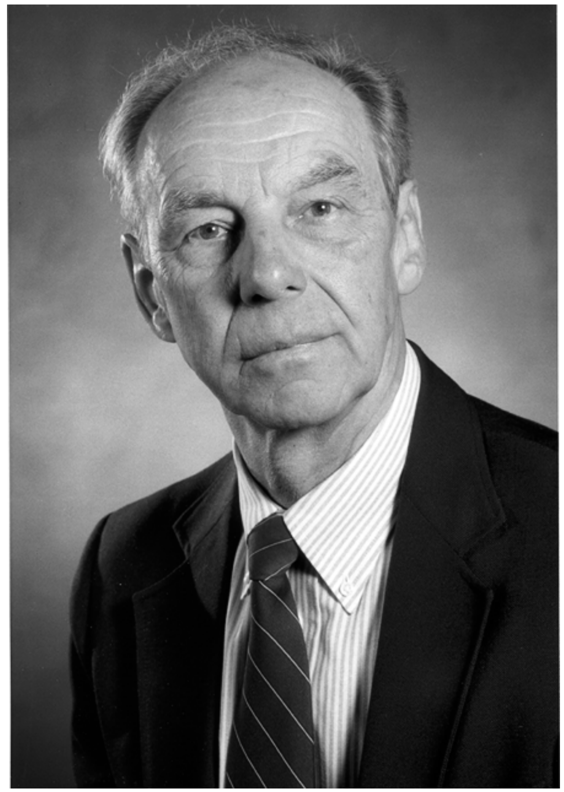
- Machine M **operates within time** $f(n)$ for $f : \mathbb{N} \to \mathbb{N}$ if for any input string x, the time required by M on x is at most $f(|\,x\,|)$.
  - $|\,x\,|$ is the length of string x.
- Function $f(n)$ is a **time bound** for M.

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] Hartmanis & Stearns (1965); Hartmanis, Lewis, & Stearns (1965).

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 string 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 $\rhd$. It must be changed; otherwise, our TM would never move to its left again by our convention on p. 23.

[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 "priming" of the last symbol of each w_i ensures that M' knows which symbol is under each cursor of M.[a]
- The first symbol of w_i is the primed version of $\triangleright$: $\triangleright'$.
  - Recall TM cursors are not allowed to move to the left of $\triangleright$ (p. 23).
  - Now the cursor of M' can move *between* the simulated strings of M.[b]

---

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

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

The Proof (continued)

- The initial configuration of M' is

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

  - $\triangleright''$ 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.

---

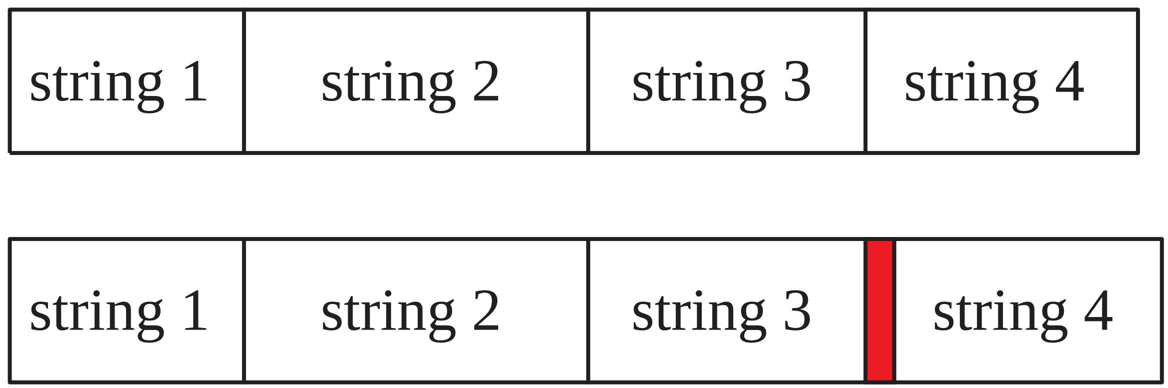
[a]Recall the TM program on p. 31.

The Proof (continued)

- It is possible that some strings of M need to be lengthened (see next page).
  - The linear-time algorithm on p. 37 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] Because whatever appears on the string of M' will be considered the output. So $\triangleright'$s and $\triangleright''$s need to 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).

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)

- M' takes $O(k^2 f(|x|))$ steps to simulate each step of M because there are k strings.
- 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 $O(k f(|x|)^2)$ if the interleaving data structure is adopted?

Simulation with Two-String TMs

We can do better with two-string 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 $f'(n) = \epsilon f(n) + n + 2$.*

[a] Hartmanis & Stearns (1965).

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 for some $k \geq 1$.
- If $L \in \mathrm{TIME}(n^k)$ for some $k \in \mathbb{N}$, it is a **polynomially decidable language**.
  - Clearly, $\mathrm{TIME}(n^k) \subseteq \mathrm{TIME}(n^{k+1})$.
- The union of all polynomially decidable languages is denoted by P:

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

- P contains problems that can be efficiently solved.

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 last string, the output string, is *write-only*.
    - So the cursor never moves to the left.
  - The cursor of the input string does not wander off into the $\sqcup$s.

[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 $f(n)$.

- $\text{SPACE}(f(n))$ is a set of languages.
  - PALINDROME $\in \text{SPACE}(\log n)$.[a]
- A linear speedup theorem similar to the one on p. 92 exists, so constant coefficients do not matter.

---

[a] Keep 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.

[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:

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

- But we cannot write

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

as in the deterministic case (p. 24) anymore.

Nondeterminism (concluded)

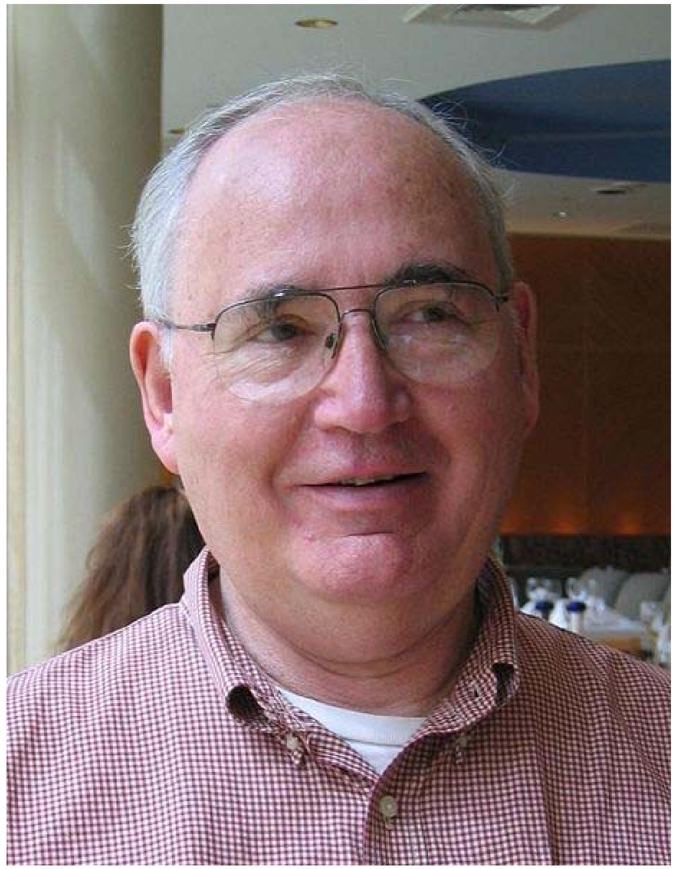
- A configuration yields another configuration in one step if there *exists* a rule in Δ that makes this happen.
- But only one will be taken.
- So there is only a single thread of computation.[a]
  - Nondeterminism is not parallelism, multiprocessing, multithreading, or quantum computation.

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

Michael O. Rabin[a] (1931–)

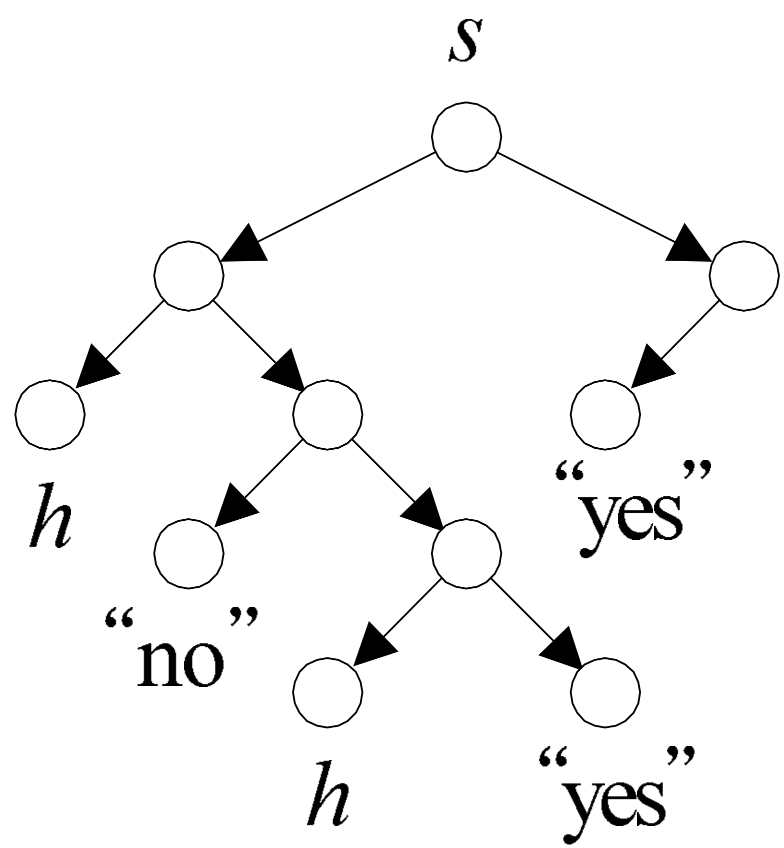
[a]Turing Award (1976).

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 (concluded)

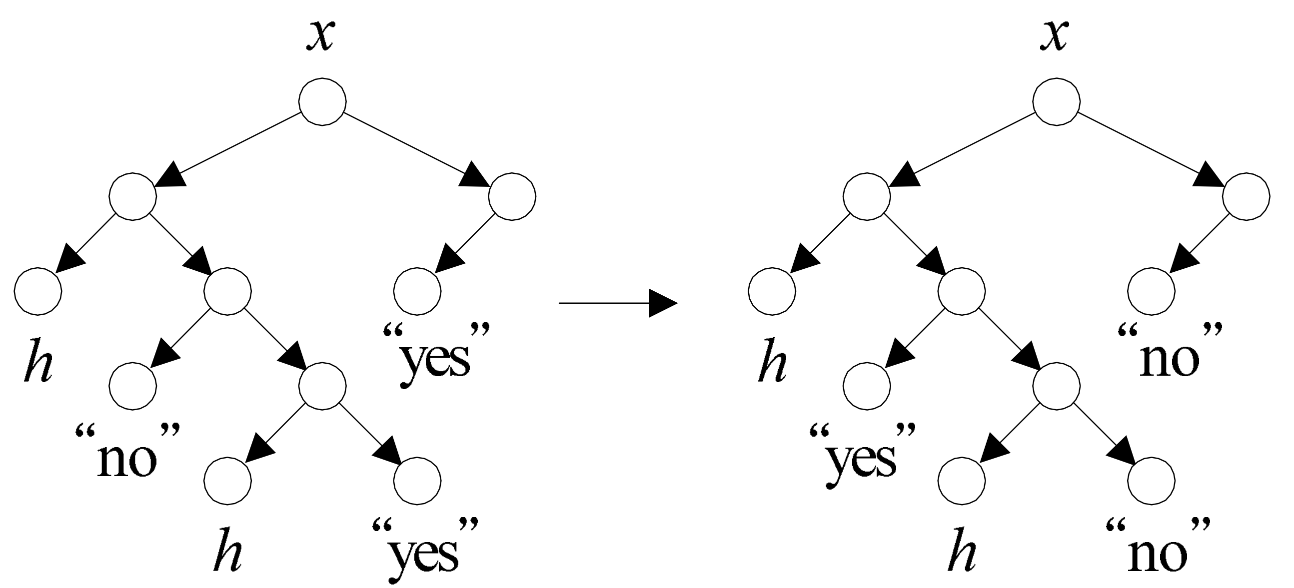
- It is not required that the NTM halts in all computation paths.[a]
- If $x \notin L$, no nondeterministic choices should lead to a "yes" state.
- The key is the algorithm's *overall* behavior not whether it gives a correct answer for each particular run.
- Note that determinism is a special case of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

---

[a] So "accepts" may be a more proper term. Some 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 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' accept languages that are *not* complements of each other.


x
h
"no"
"yes"
h
"yes"
x
h
"yes"
"no"
h
"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. 328).
  - Boolean satisfiability (p. 117 and p. 192).
- 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 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 $\text{NTIME}(f(n))) \subseteq \bigcup_{c>1} \text{TIME}(c^{f(n)})$.[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

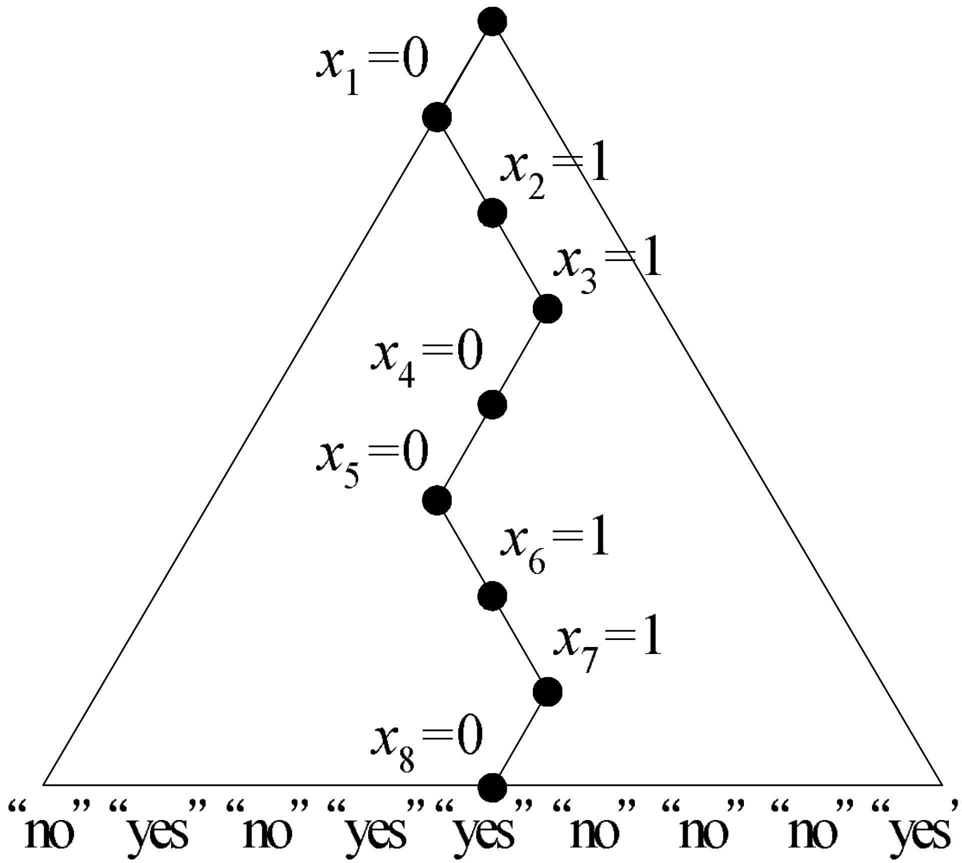
- Does converting an NTM into a TM require exploring all computation paths of the NTM as done in Theorem 6 (p. 114)?
- This is a key question in theory with important practical implications.

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.
- Recall that ϕ is satisfiable if and only if there is a truth assignment that satisfies ϕ.

---

[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.