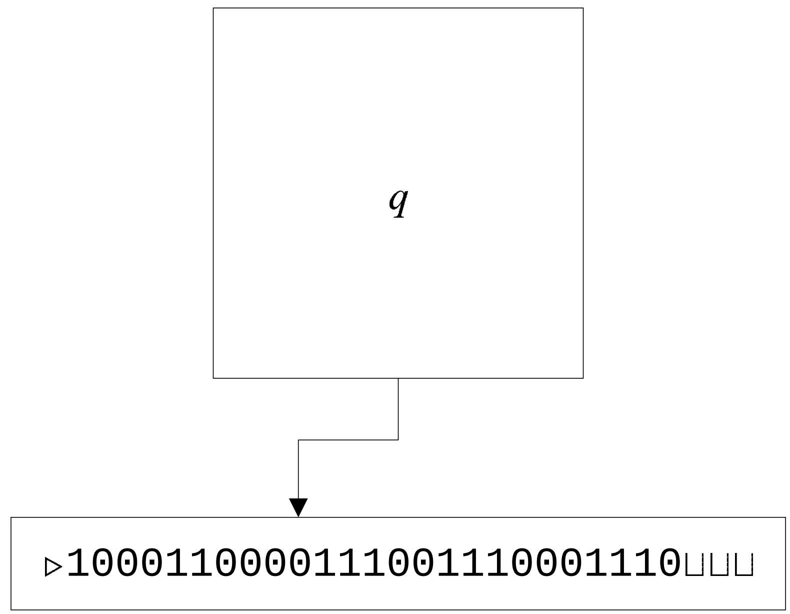
Programming TMs

- We will skip the details.
- It is not without loss of generality, in most cases, to describe a TM with pseudocode.
- They are equivalent anyway.
- Because of the simplicity of the TM (but not its programs), the model has the advantage of when it comes to complexity issues.

Configurations

- A **configuration** is a complete description of the current state of the computation.
- The specification of a configuration is sufficient for the computation to continue as if it had not been stopped.
  - What does your PC save before it enters the sleep mode?
- A configuration is a triple (q, w, u), where $q \in K$, $w \in \Sigma^*$ is the string to the left of the cursor (inclusive), and $u \in \Sigma^*$ is the string to the right of the cursor.



- $w = \triangleright$1000110000.
- $u =$ 11100111 0001110.

Yielding

- Fix a TM M.
- Configuration (q, w, u) **yields** configuration (q', w', u') in one step, denoted

$$(q, w, u) \xrightarrow{M} (q', w', u'),$$

  if a step of M from configuration (q, w, u) results in configuration (q', w', u').
- That configuration (q, w, u) yields configuration (q', w', u') in $k \in \mathbb{N}$ steps is denoted by $(q, w, u) \xrightarrow{M^k} (q', w', u')$.
- That configuration (q, w, u) yields configuration (q', w', u') is denoted by $(q, w, u) \xrightarrow{M^*} (q', w', u')$.

Inserting a Symbol

- We want to compute $f(x) = ax$.
- The TM moves the last symbol of x to the right by one position, it then moves the next to last symbol to the right, and so on.
- The TM finally writes a in the first position.
- The total number of steps is $O(n)$, where n is the length of x.

Palindromes

- A string is a **palindrome** if it reads the same forwards and backwards (e.g., 001100).
- A TM program can be written to recognize palindromes: "yes" for palindromes and "no" for nonpalindromes.
  - It matches the first character with the last character, the second character with the next to last character, etc.
  - This program takes $O(n^2)$ steps.
  - There is a matching lower bound of $\Omega(n^2)$.

Decidability and Recursive Languages

- Let $L \subseteq (\Sigma - \{\sqcup\})^*$ be a **language**, i.e., a set of strings of symbols with a finite length.
- 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 called a **recursive language**.
  - Palindromes over $\{0, 1\}^*$ constitute a recursive language.

Acceptability and Recursively Enumerable Languages

- Let $L \subseteq (\Sigma - \{\sqcup\})^*$ be a **language**, i.e., a set of strings of symbols with a finite length.
- 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.
- If L is accepted by some TM, then L is called a **recursively enumerable language**.

Recursive and Recursively Enumerable Languages

Proposition 1 *If L is recursive, then it is recursively enumerable.*

- Let TM M decides L.
- M' is identical to M except that when M is about to halt with a "no" state, M' moves its cursor to the right forever and never halts.
  - M' can be constructed by slightly modifying M's program.
- L is clearly accepted by M'.

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** if such an M exists.

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), various 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 to be Turing-uncomputable (yet).

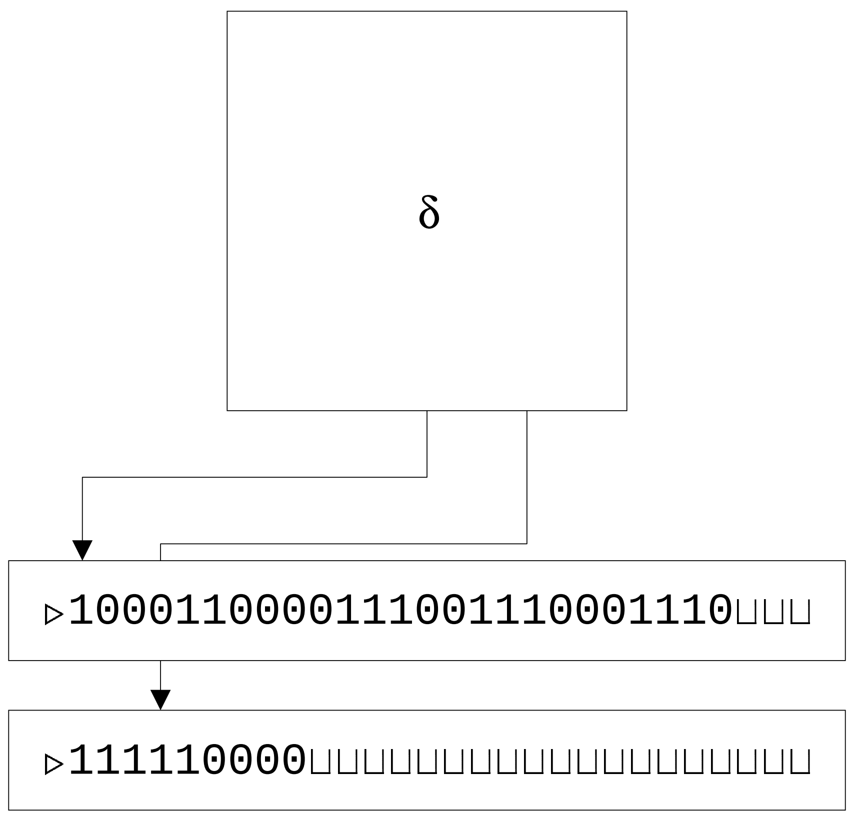
Extended Church's Thesis

- 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 will be binary from now on.

Turing Machines with Multiple Strings

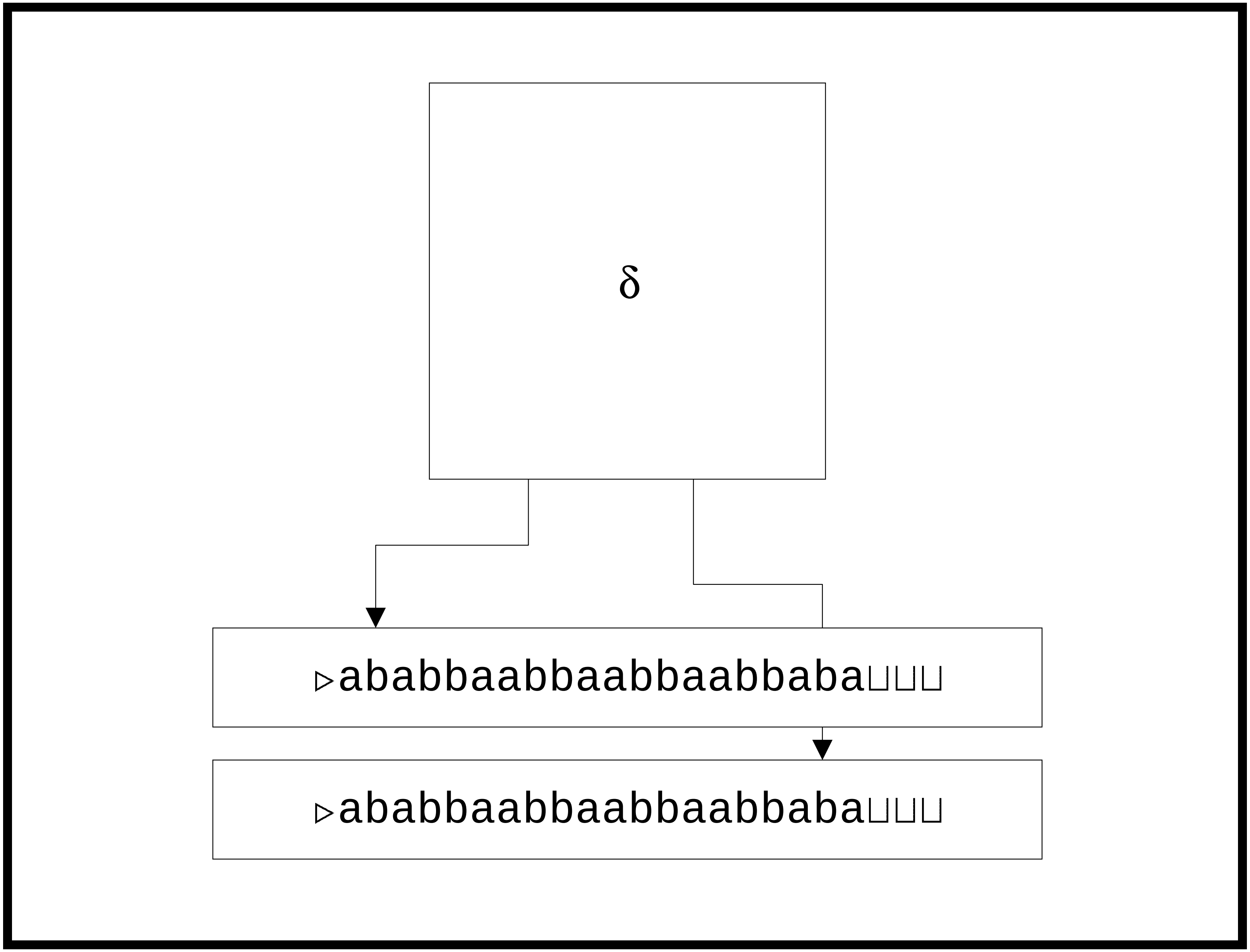
- A k-string Turing machine (TM) is a quadruple $M = (K, \Sigma, \delta, s)$.
- K, Σ, s are as before.
- $\delta : K \times \Sigma^k \to (K \cup \{h, \text{"yes"}, \text{"no"}\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$.
- All strings start with a $\triangleright$.
- The first string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



Palindromes Revisited

- A 2-string TM can decide palindromes 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 the input 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

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

where $w_i u_i$ is the ith string and the ith cursor is reading the last symbol of w_i.

  - Note 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 $f(n) = O(n^2)$.

---

[a] Hartmanis, Stearns, 1965, Hartmanis, Lewis, Stearns, 1965.

The Simulation Technique

Theorem 2 *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

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

- 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. 25 can be used for each such string.
- The simulation continues until M halts.
- M' erases all strings of M except the last one.

The Proof (continued)

- Since M halts within time $f(|x|)$, none of its strings ever becomes longer than $f(|x|)$.
- The total length of the string of M' at any moment is $O(kf(|x|))$.
- Simulating each step of M takes, *per string of M*, $O(kf(|x|))$ steps to collect information and $O(kf(|x|))$ steps to write and, if needed, to lengthen the string.
  - The total number of M' steps is hence $O(k^2 f(|x|))$.
- As there are $f(|x|)$ steps of M to simulate, M' operates within time $O(k^2 f(|x|)^2)$.

Linear Speedup

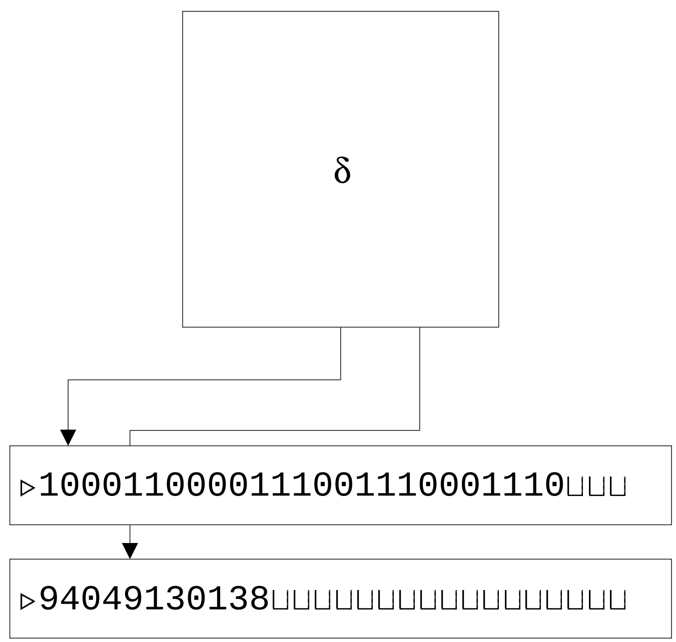
Theorem 3 *Let* $L \in \mathit{TIME}(f(n))$*. Then for any* $\epsilon > 0$*,* $L \in \mathit{TIME}(f'(n))$*, where* $f'(n) = \epsilon f(n) + n + 2$*.*

- 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 TM $M' = (K', \Sigma', \delta', s')$ operating within the time bound $f'(n)$ and which *simulates* M.
- Set $k' = \max(k, 2)$.
- We encode m symbols of M in *one* symbol of M' so that M' can simulate m steps of M within *six* steps.

The Proof (continued)

- $m \in \mathbb{Z}^+$ depend on M and ϵ alone.
- $\Sigma' = \Sigma \cup \Sigma^m$.
- Phase one of M':
  - M' has states corresponding to $K \times \Sigma^i$.
  - 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 takes $m\lceil |x|/m \rceil + 2$ steps.

Compression of Symbols; Enlarging the Word Length

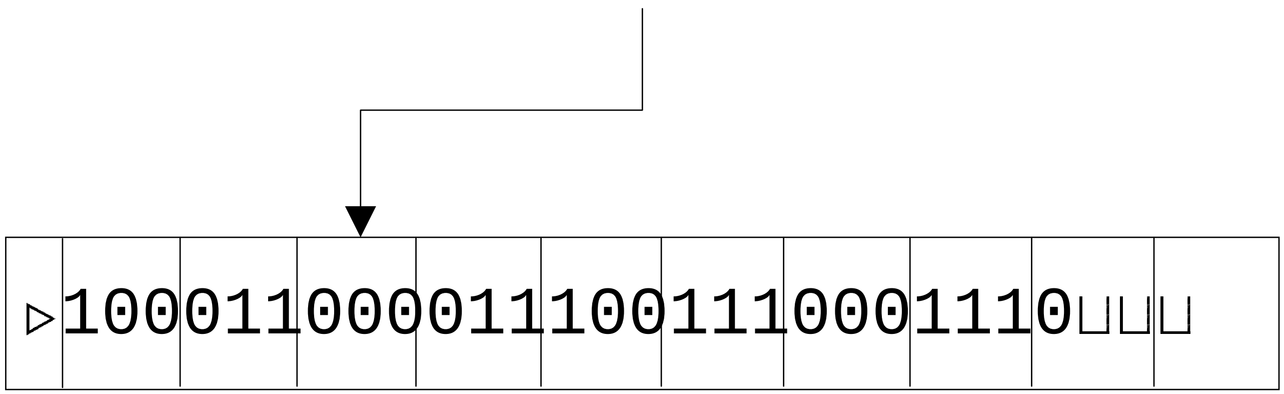


- $m = 3$.
- 3-ary representation, with $\sqcup \rightarrow 2$.

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' repeatedly 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)

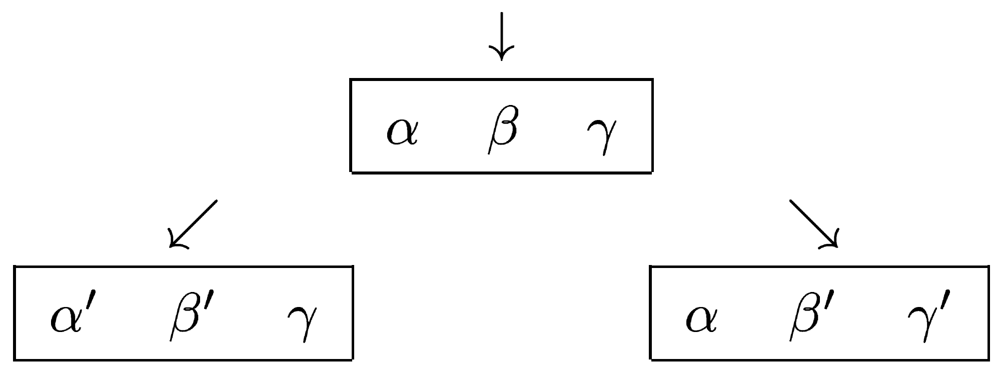


- $m = 3$.
- $\ell = 7$.
- $\lceil 8/3 \rceil = 3$.
- $j_i = ((8 - 1) \bmod 3) + 1 = 2$.

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.
- M' now "remembers" all Σ' symbols 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.
- Because no cursor of M can get out of the m-tuples scanned by M' above, M' has all the information to predict the next m moves of M!

The Proof (continued)



The Proof (continued)

- 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

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

- Choose $m = \lceil 6/\epsilon \rceil$ to complete the proof.

Implications of the Speedup Theorem

- We can trade state size for speed.
- 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 $k \in \mathbb{N}$.
- The union of all polynomially decidable languages is denoted by P, that is,

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

- Think of P as efficiently solvable problems.

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.
    - 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.
- 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 $f : \mathbb{N} \rightarrow \mathbb{N}$ 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

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

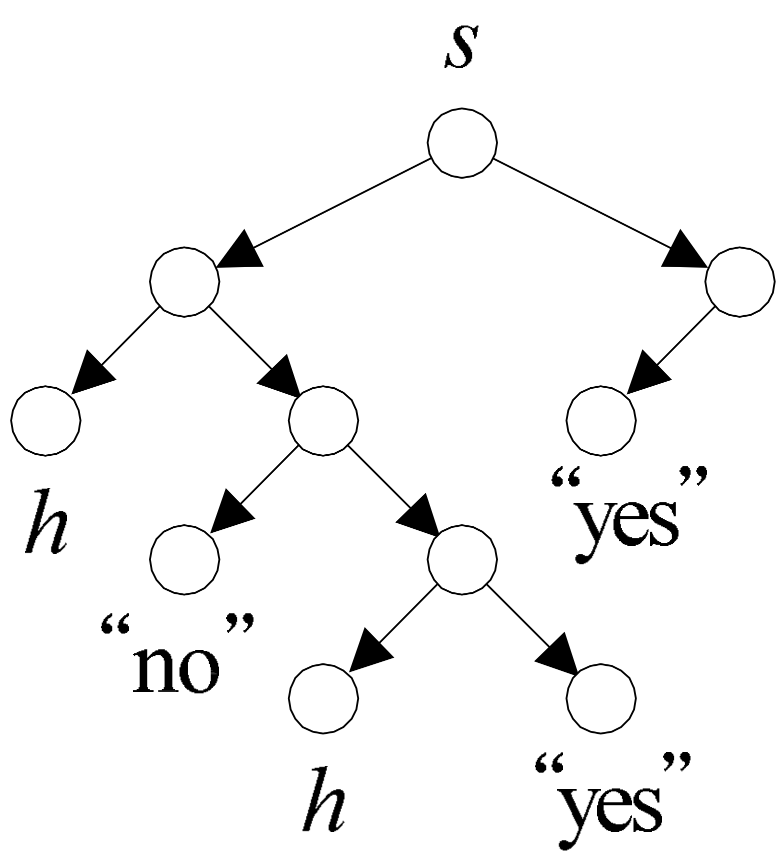
  if there is a TM with input and output that decides L and operates within space bound $f(n)$.

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

Nondeterminism

- 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.
- Determinism is a special case of nondeterminism.

Computation Tree and Computation Path

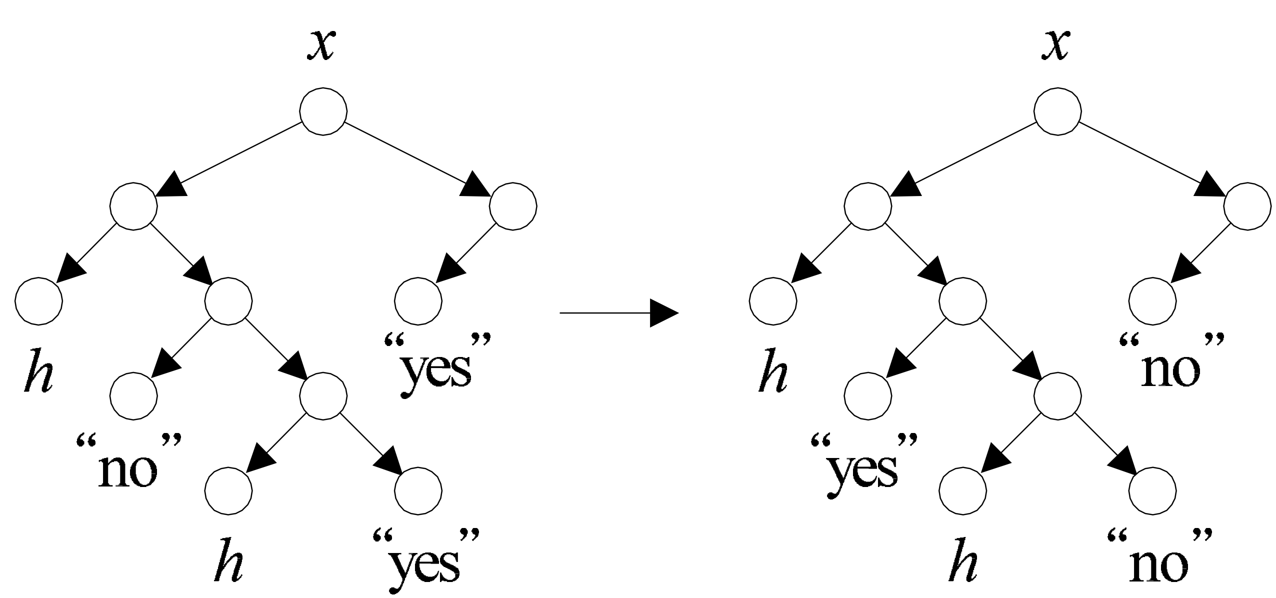


Decidability under Nondeterminism

- 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.

Complementing a TM's Halting States

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



A Nondeterministic Algorithm for Satisfiability

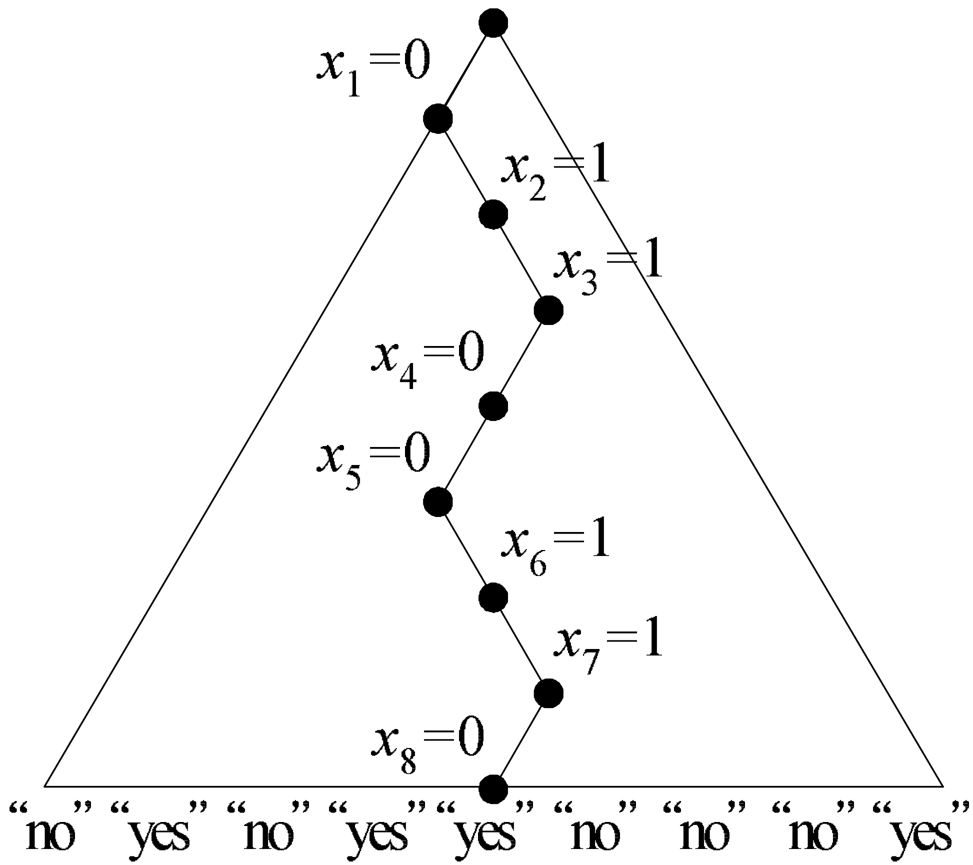
ϕ is a boolean formula with n variables.

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

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 the "yes" state.
- General paradigm: Guess a "proof" and verify it.

The Computation Tree for Satisfiability



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 $d_{ij} = d_{ji}$ (not essential here).
- The **traveling salesman problem** (TSP) asks for the total distance of the shortest tour of the cities.
- The decision version TSP (D) asks if therer is a tour with a total distance at most B, where B is an input.
- Both problems are extremely 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
$x_{n+1} := x_1$; {For convenience.}
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.
- Turning an NTM into a TM seems to require exploring all the computation paths of the NTM.

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).
  - Hamiltonian path.
  - Graph colorability.
  - TSP (D).
- The most important open problem in theoretical computer science is if $\mathrm{P} = \mathrm{NP}$.