

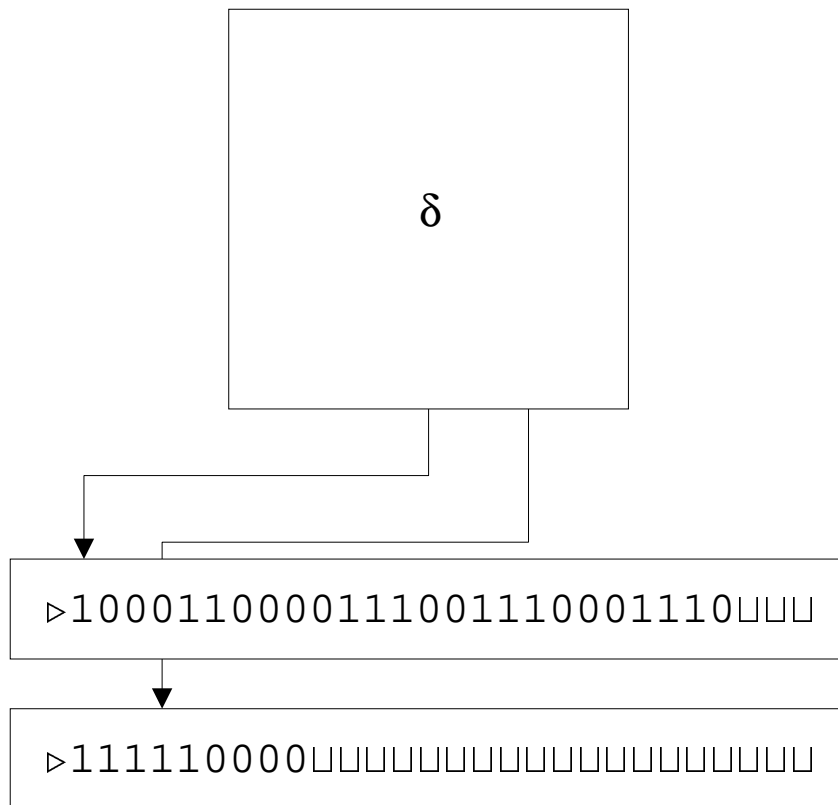
Extended Church's Thesis

- 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. 11111111.
- All numbers for TMs will be binary from now on.

Turing Machines with Multiple Strings

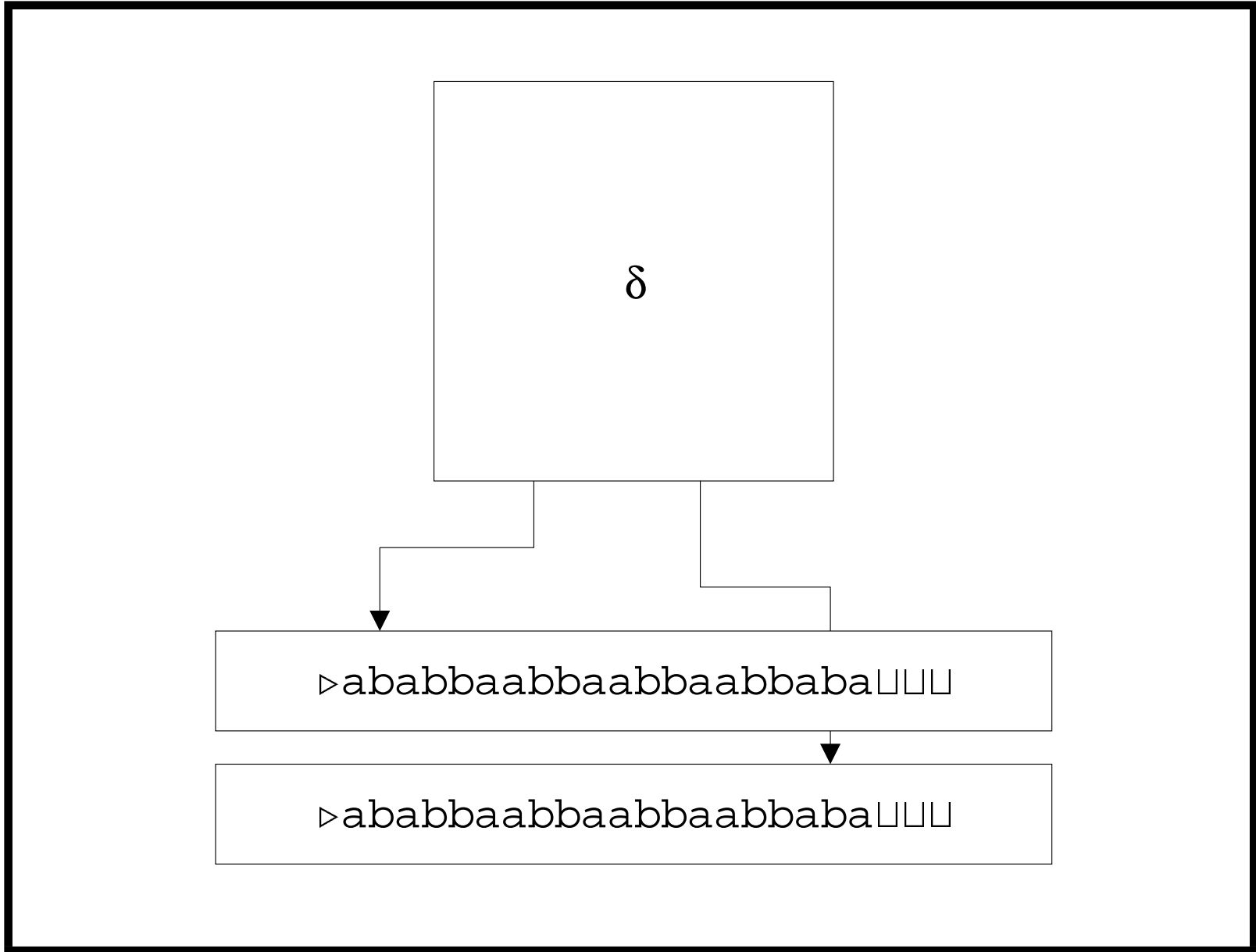
- A k -string Turing machine (TM) is a quadruple $M = (K, \Sigma, \delta, s)$.
- K, Σ, s are as before.
- $\delta : K \times \Sigma^k \rightarrow (K \cup \{h, \text{“yes”}, \text{“no”}\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$.
- All strings start with a \triangleright .
- The first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



Configurations and Yielding

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

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

- $w_i u_i$ is the i th string.
 - The i th 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

$$(s, \overbrace{\triangleright, x, \triangleright, \epsilon, \triangleright, \epsilon, \dots, \triangleright, \epsilon}^{2k}).$$

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 $f : \mathbb{N} \rightarrow \mathbb{N}$ 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 $f(n) = O(n)$.

^aHartmanis 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 $(q, w_1, u_1, w_2, u_2, \dots, w_k, u_k)$ 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 “priming” is to ensure that M' knows which symbol is under the cursor for each simulated string.^a
- The initial configuration of M' is

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

^aAdded 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 $O(kf(|x|))$.

^aWe 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 $O(k^2f(|x|))$ steps to simulate each step of M .
- As there are $f(|x|)$ steps of M to simulate, M' operates within time $O(k^2f(|x|)^2)$.

Linear Speedup^a

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

^aHartmanis 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:

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

- 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 \square 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 $(H, w_1, u_1, w_2, u_2, \dots, w_k, u_k)$, then the **space required by M on input x** is $\sum_{i=1}^k |w_i u_i|$.

^aCorrected 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 $f(n)$.

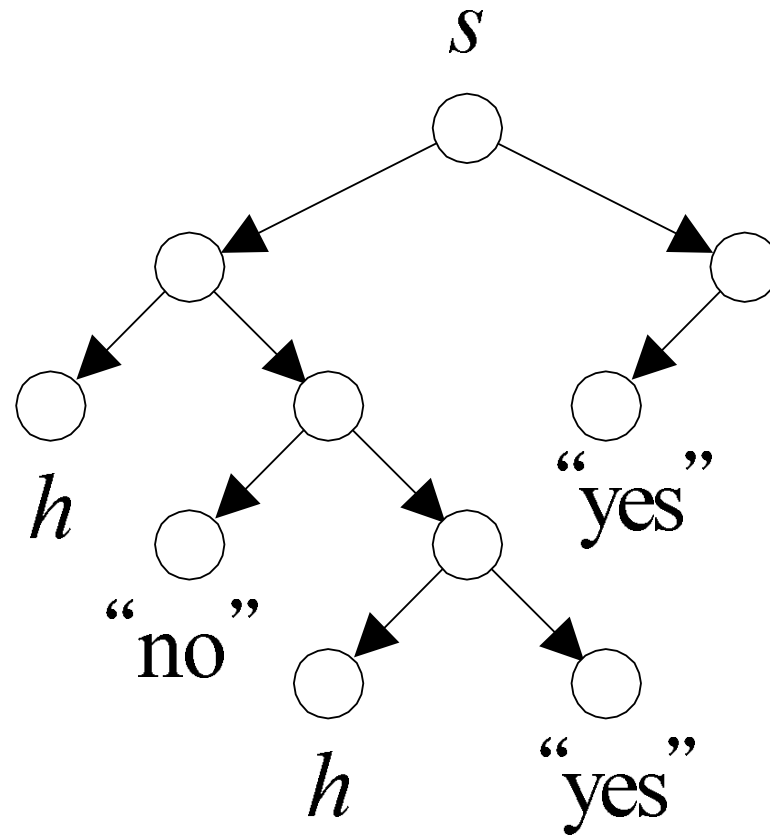
- $\text{SPACE}(f(n))$ is a set of languages.
 - $\text{PALINDROME} \in \text{SPACE}(\log n)$: Keep 3 counters.
- As in the linear speedup theorem (Theorem 4), constant coefficients do not matter.

Nondeterminism^a

- A **nondeterministic Turing machine (NTM)** is a quadruple $N = (K, \Sigma, \Delta, s)$.
- K, Σ, s are as before.
- $\Delta \subseteq K \times \Sigma \rightarrow (K \cup \{h, \text{“yes”}, \text{“no”}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ 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.

^aRabin 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.^a
 - 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.

^aSo “accepts” may be a more proper term here.

An Example

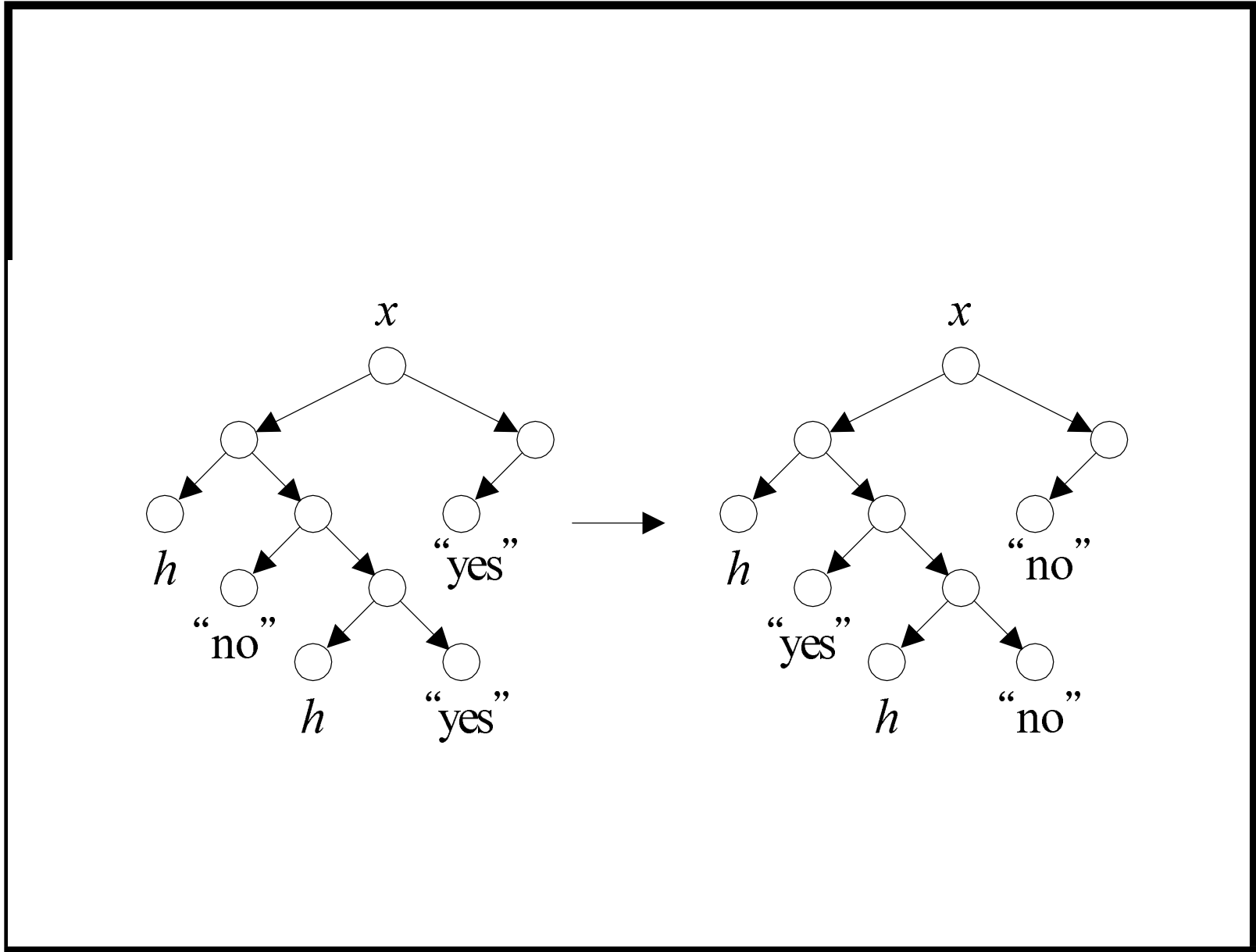
- Let L be the set of logical conclusions of a set of axioms.
 - Predicates not in L may be false under the axioms.
 - They may also be independent of the axioms.
 - * That is, they can be assumed true or false without contradicting the axioms.

An Example (concluded)

- Let ϕ be a predicate whose validity we would like to prove.
- Consider the nondeterministic algorithm:
 - 1: $b := \text{true};$
 - 2: **while** the input predicate $\phi \neq b$ **do**
 - 3: Generate a logical conclusion of b by applying some of the axioms; {Nondeterministic choice.}
 - 4: Assign this conclusion to b ;
 - 5: **end while**
 - 6: “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.

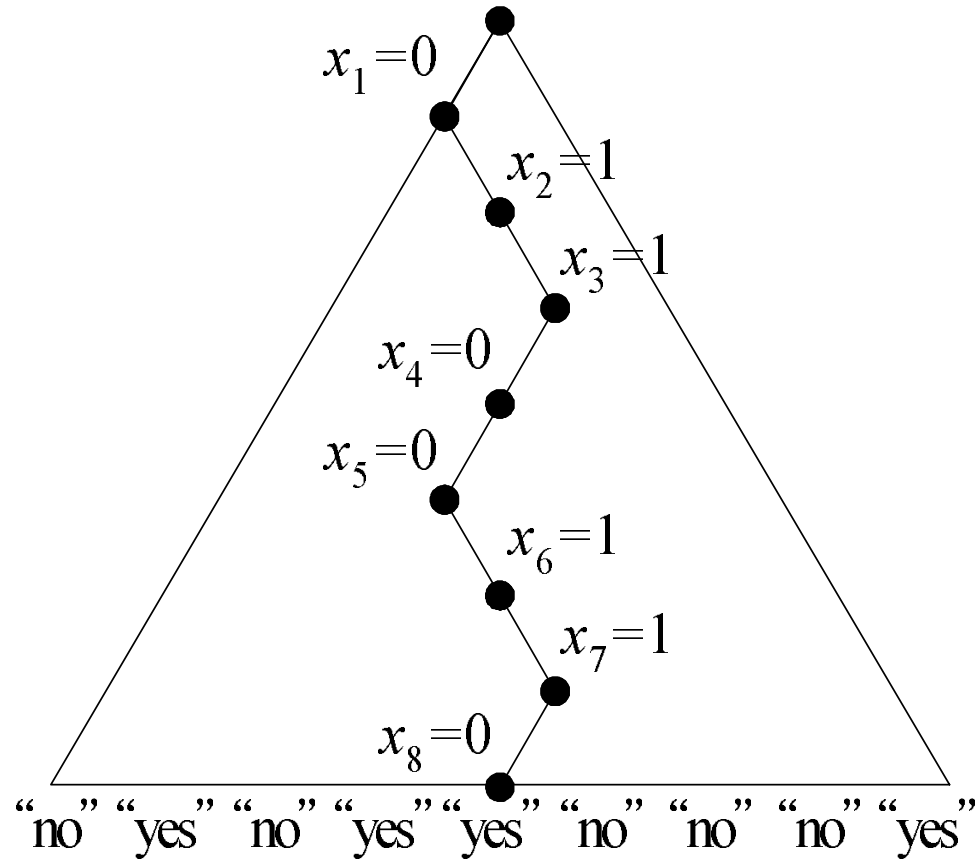


A Nondeterministic Algorithm for Satisfiability

ϕ is a boolean formula with n variables.

```
1: for  $i = 1, 2, \dots, n$  do  
2:   Guess  $x_i \in \{0, 1\}$ ; {Nondeterministic choice.}  
3: end for  
4: {Verification:}  
5: if  $\phi(x_1, x_2, \dots, x_n) = 1$  then  
6:   “yes”;  
7: else  
8:   “no”;  
9: 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 $1, 2, \dots, n$ 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. 316 and p. 395).

A Nondeterministic Algorithm for TSP (D)

```
1: for  $i = 1, 2, \dots, n$  do  
2:   Guess  $x_i \in \{1, 2, \dots, n\}$ ; {The  $i$ th city.}a  
3: end for  
4:  $x_{n+1} := x_1$ ;  
5: {Verification stage:}  
6: if  $x_1, x_2, \dots, x_n$  are distinct and  $\sum_{i=1}^n d_{x_i, x_{i+1}} \leq B$  then  
7:   “yes”;  
8: else  
9:   “no”;  
10: end if
```

^aCan 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 $f : \mathbb{N} \rightarrow \mathbb{N}$, 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

- $\text{NTIME}(f(n))$ is the set of languages decided by NTMs within time $f(n)$.
- $\text{NTIME}(f(n))$ is a complexity class.

NP

- Define

$$\text{NP} = \bigcup_{k>0} \text{NTIME}(n^k).$$

- Clearly $P \subseteq \text{NP}$.
- Think of NP as efficiently *verifiable* problems.
 - Boolean satisfiability (SAT).
 - TSP (D).
- The most important open problem in computer science is whether $P = \text{NP}$.

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 N 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 $\text{NTIME}(f(n)) \subseteq \bigcup_{c>1} \text{TIME}(c^{f(n)})$.

NTIME vs. TIME

- Does converting an NTM into a TM require exploring all of the computation paths of the NTM as done in Theorem 5 (p. 92)?
- 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 $f(n)$.

- $\text{NSPACE}(f(n))$ is a set of languages.
- As in the linear speedup theorem (Theorem 4 on p. 70), 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($n \log n$)

```
1:  $x_1 := a$ ; {Assume  $a \neq b$ .}
2: for  $i = 2, 3, \dots, n$  do
3:   Guess  $x_i \in \{v_1, v_2, \dots, v_n\}$ ; {The  $i$ th node.}
4: end for
5: for  $i = 2, 3, \dots, n$  do
6:   if  $(x_{i-1}, x_i) \notin E$  then
7:     “no”;
8:   end if
9:   if  $x_i = b$  then
10:    “yes”;
11:   end if
12: end for
13: “no”;
```

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

```
1:  $x := a$ ;  
2: for  $i = 2, 3, \dots, n$  do  
3:   Guess  $y \in \{v_1, v_2, \dots, v_n\}$ ; {The next node.}  
4:   if  $(x, y) \notin E$  then  
5:     “no”;  
6:   end if  
7:   if  $y = b$  then  
8:     “yes”;  
9:   end if  
10:   $x := y$ ;  
11: end for  
12: “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.
- $\text{REACHABILITY} \in \text{P}$ (p. 185).