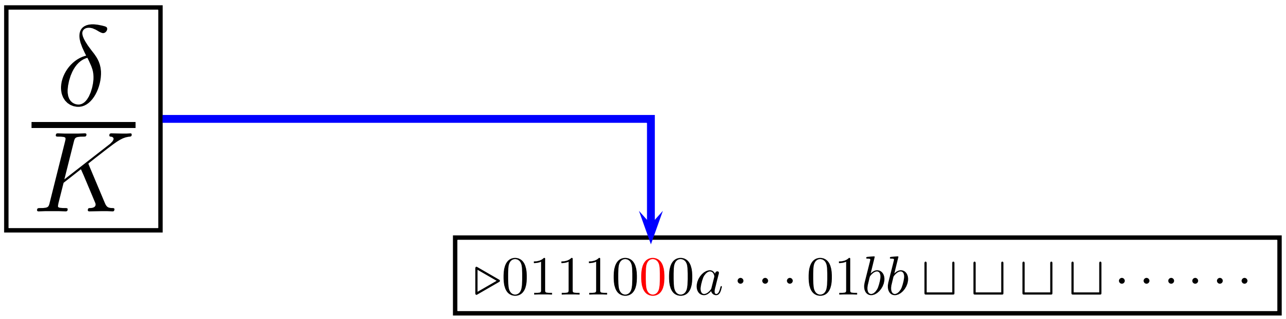
Theory of Computation
Chapter 2: Turing Machines

Guan-Shieng Huang

Feb. 24, 2003
Feb. 19, 2006

Turing Machine



Definition of TMs

A Turing Machine is a quadruple $M = (K, \Sigma, \delta, s)$, where

1. K is a finite set of states; (line numbers)
2. Σ is a finite set of symbols including $\sqcup$ and $\triangleright$; (alphabet)
3. $\delta : K \times \Sigma \to (K \cup \{\text{h}, \text{“yes”}, \text{“no”}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$, a transition function; (instructions)
4. $s \in K$, the initial state. (starting point)

- h: halt, "yes":accept, "no": reject
  (terminate the execution)
- $\rightarrow$: move right, $\leftarrow$: move left, $-$: stay
  (move the head)
- $\sqcup$: blank, $\triangleright$: the boundary symbol

- $\delta(q, \sigma) = (p, \rho, D)$
  While reading σ at line q, go to line p and write out ρ on the tape. Move the head according to the direction of D.

- $\delta(q, \triangleright) = (p, \rho, \rightarrow)$, to avoid crash.

Example 2.1

$p \in K,$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
$s,$	0	$(s, 0, \rightarrow)$
$s,$	1	$(s, 1, \rightarrow)$
$s,$	$\sqcup$	$(q, \sqcup, \leftarrow)$
$s,$	$\triangleright$	$(s, \triangleright, \rightarrow)$
$q,$	0	$(q_0, \sqcup, \rightarrow)$
$q,$	1	$(q_1, \sqcup, \rightarrow)$
$q,$	$\sqcup$	$(q, \sqcup, -)$
$q,$	$\triangleright$	$(h, \triangleright, \rightarrow)$
$q_0,$	0	$(s, 0, \leftarrow)$
$q_0,$	1	$(s, 0, \leftarrow)$
$q_0,$	$\sqcup$	$(s, 0, \leftarrow)$
$q_0,$	$\triangleright$	$(h, \triangleright, \rightarrow)$
$q_1,$	0	$(s, 1, \leftarrow)$
$q_1,$	1	$(s, 1, \leftarrow)$
$q_1,$	$\sqcup$	$(s, 1, \leftarrow)$
$q_1,$	$\triangleright$	$(h, \triangleright, \rightarrow)$

halt

0. $s,\ \underline{\triangleright}010$
1. $s,\ \triangleright\underline{0}10$
2. $s,\ \triangleright0\underline{1}0$
3. $s,\ \triangleright01\underline{0}$
4. $s,\ \triangleright010\underline{\sqcup}$
5. $q,\ \triangleright01\underline{0}\sqcup$
6. $q_0,\ \triangleright01\sqcup\underline{\sqcup}$
7. $s,\ \triangleright01\underline{\sqcup}0$
8. $q,\ \triangleright0\underline{1}\sqcup0$
9. $q_1,\ \triangleright0\sqcup\underline{\sqcup}0$
10. $s,\ \triangleright0\underline{\sqcup}10$
11. $q,\ \triangleright\underline{0}\sqcup10$
12. $q_0,\ \triangleright\sqcup\underline{\sqcup}10$
13. $s,\ \triangleright\underline{\sqcup}010$
14. $q,\ \underline{\triangleright}\sqcup010$
15. $h,\ \triangleright\underline{\sqcup}010$

Figure 2.1. Turing machine and computation.

Remark

x: input of M

$$M(x) = \begin{cases} \text{“yes”} \\ \text{“no”} \\ y \text{ if } M \text{ entered } h \\ \nearrow \text{ if } M \text{ never terminates} \end{cases}$$

Example 2.2

$(n)_2 \to (n+1)_2$ if no overflow happens.

$p \in K,$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
$s,$	0	$(s, 0, \rightarrow)$
$s,$	1	$(s, 1, \rightarrow)$
$s,$	$\sqcup$	$(q, \sqcup, \leftarrow)$
$s,$	$\triangleright$	$(s, \triangleright, \rightarrow)$
$q,$	0	$(h, 1, -)$
$q,$	1	$(q, 0, \leftarrow)$
$q,$	$\triangleright$	$(h, \triangleright, \rightarrow)$

Figure 2.2. Turing machine for binary successor.

Example 2.3 — Palindrome

$p \in K,$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
s	0	$(q_0, \triangleright, \rightarrow)$
s	1	$(q_1, \triangleright, \rightarrow)$
s	$\triangleright$	$(s, \triangleright, \rightarrow)$
s	$\sqcup$	$(\text{"yes"}, \sqcup, -)$
q_0	0	$(q_0, 0, \rightarrow)$
q_0	1	$(q_0, 1, \rightarrow)$
q_0	$\sqcup$	$(q_0', \sqcup, \leftarrow)$
q_1	0	$(q_1, 0, \rightarrow)$
q_1	1	$(q_1, 1, \rightarrow)$
q_1	$\sqcup$	$(q_1', \sqcup, \leftarrow)$

$p \in K,$	$\sigma \in \Sigma$	$\delta(p, \sigma)$
q_0'	0	$(q, \sqcup, \leftarrow)$
q_0'	1	$(\text{"no"}, 1, -)$
q_0'	$\triangleright$	$(\text{"yes"}, \sqcup, \rightarrow)$
q_1'	0	$(\text{"no"}, 1, -)$
q_1'	1	$(q, \sqcup, \leftarrow)$
q_1'	$\triangleright$	$(\text{"yes"}, \triangleright, \rightarrow)$
q	0	$(q, 0, \leftarrow)$
q	1	$(q, 1, \leftarrow)$
q	$\triangleright$	$(s, \triangleright, \rightarrow)$

Figure 2.3. Turing machine for palindromes.

Turing Machines as Algorithms

- $L \subseteq (\Sigma - \{\sqcup, \triangleright\})^*$, a language
- A TM M decides L if for all string x,

$$\begin{cases} x \in L \Rightarrow M(x) = \text{“yes”} \\ x \notin L \Rightarrow M(x) = \text{“no”}. \end{cases}$$

- A TM M accepts L if for all string x,

$$\begin{cases} x \in L \Rightarrow M(x) = \text{“yes”} \\ x \notin L \Rightarrow M(x) = \nearrow. \end{cases}$$

- If L is decided by some TM, we say L is recursive.
- If L is accepted by some TM, we say L is recursively enumerable.

Proposition 2.1

If L is recursive, then it is recursively enumerable.

Representation of mathematical objects: (data structure)

1. graphs, sets, numbers, ...
2. All acceptable encodings are polynomially related.

   (a) binary, ternary
   (b) adjacency matrix, adjacency list

However, unary representation of numbers is an exception.

k-string Turing Machines

A k-string Turing machine is a quadruple (K, Σ, δ, s) where

1. K, Σ, s are exactly as in ordinary Turing machines;
2. $\delta : K \times \Sigma^k \rightarrow (K \cup \{\text{h}, \text{“yes”}, \text{“no”}\}) \times (\Sigma \times \{\leftarrow, \rightarrow, -\})^k$;

An Example

$p \in K,$	$\sigma_1 \in \Sigma$	$\sigma_2 \in \Sigma$	$\delta(p, \sigma_1, \sigma_2)$
$s,$	0	$\sqcup$	$(s, 0, \rightarrow, 0, \rightarrow)$
$s,$	1	$\sqcup$	$(s, 1, \rightarrow, 1, \rightarrow)$
$s,$	$\triangleright$	$\triangleright$	$(s, \triangleright, \rightarrow, \triangleright, \rightarrow)$
$s,$	$\sqcup$	$\sqcup$	$(q, \sqcup, \leftarrow, \sqcup, -)$
$q,$	0	$\sqcup$	$(q, 0, \leftarrow, \sqcup, -)$
$q,$	1	$\sqcup$	$(q, 1, \leftarrow, \sqcup, -)$
$q,$	$\triangleright$	$\sqcup$	$(p, \triangleright, \rightarrow, \sqcup, \leftarrow)$
$p,$	0	0	$(p, 0, \rightarrow, \sqcup, \leftarrow)$
$p,$	1	1	$(p, 1, \rightarrow, \sqcup, \leftarrow)$
$p,$	0	1	$(\text{"no"}, 0, -, 1, -)$
$p,$	1	0	$(\text{"no"}, 1, -, 0, -)$
$p,$	$\sqcup$	$\triangleright$	$(\text{"yes"}, \sqcup, -, \triangleright, \rightarrow)$

Figure 2.5. 2-string Turing machine for palindromes.

1. If for a k-string Turing machine M and input x we have

$$(s, \triangleright, x, \triangleright, \epsilon, \ldots, \triangleright, \epsilon) \xrightarrow{M^t} (H, w_1, u_1, \ldots, w_k, u_k)$$

   for some $H \in \{\text{h}, \text{“yes”}, \text{“no”}\}$, then the time required by M on input x is t.

2. If for any input string x of length $|x|$, M terminates on input x within time $f(|x|)$, we say $f(n)$ is a time bound for M.

(worst case analysis)

TIME($f(n)$): the set of all languages that can be decided by TMs in time $f(n)$.

Theorem 2.1

Given any k-string TM M operating within time $f(n)$, we can construct a TM M' operating within time $O(f(n)^2)$ and such that, for any input x, $M(x) = M'(x)$.

(by simulation)

Linear Speedup

Theorem 2.2

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

Definition

$\mathcal{P} = \bigcup_{k \geq 1} \text{TIME}(n^k)$.

Space Bounds

A k-string TM with input and output is an ordinary k-string TM such that

1. the first tape is read-only;
   (Input cannot be modified.)
2. the last tape is write-only.
   (Output cannot be wound back.)

Proposition

For any k-string TM M operating with time bound $f(n)$ there is a $(k+2)$-string TM M' with input and output, which operates within time bound $O(f(n))$.

Space Bound for TM

Suppose that, for a k-string TM M and input x,

$$(s, \triangleright, x, \ldots, \triangleright, \epsilon) \xrightarrow{M^*} (H, w_1, u_1, \ldots, w_k, u_k)$$

where $H \in \{\text{h}, \text{"yes"}, \text{"no"}\}$ is a halting state.

1. The space required by M on input x is $\sum_{i=1}^{k} |w_i u_i|$.
2. If M is a machine with input and output, then the space required by M on input x is $\sum_{i=2}^{k-1} |w_i u_i|$.

1. We say that Turing machine M operates within space bound $f(n)$ if, for any input x, M requires space at most $f(|x|)$.
2. A language L is in the space complexity class SPACE$(f(n))$ if there is a TM with I/O that decides L and operates within space bound $f(n)$.
3. Define $\mathcal{L} = \text{SPACE}(\lg(n))$.

Theorem 2.3

Let L be a language in SPACE$(f(n))$. Then, for any $\epsilon > 0$, $L \in$ SPACE$(2 + \epsilon \cdot f(n))$.

Random Access Machines

Input: $(i_1, i_2, \ldots, i_n)$

Output: r_0 (accumulator)

Memory: $r_0, r_1, r_2, \ldots$ (infinite memory)

k: program counter

Three address modes: (for x)

1. j: direct;
2. $\uparrow j$: indirect;
3. $= j$: immediate.

(arbitrary large number)

Instruction	Operand	Semantics
READ	j	$r_0 := i_j$
READ	$\uparrow j$	$r_0 := i_{r_j}$
STORE	j	$r_j := r_0$
STORE	$\uparrow j$	$r_{r_j} := r_0$
LOAD	x	$r_0 := x$
ADD	x	$r_0 := r_0 + x$
SUB	x	$r_0 := r_0 - x$
HALF		$r_0 := \lfloor \frac{r_0}{2} \rfloor$
JUMP	j	$\kappa := j$
JPOS	j	**if** $r_0 > 0$ **then** $\kappa := j$
JZERO	j	**if** $r_0 = 0$ **then** $\kappa := j$
JNEG	j	**if** $r_0 < 0$ **then** $\kappa := j$
HALT		$\kappa := 0$

Theorem 2.5

If a RAM program Π computes a function ϕ in time $f(n)$, then there is a 7-string TM which computes ϕ in time $O(f(n)^3)$.
(by simulation)

Nondeterministic Machines

A nondeterministic TM is a quadruple $N = (K, \Sigma, \Delta, s)$, where

1. K, Σ, s are as in ordinary TM;
2. $\Delta \subseteq (K \times \Sigma) \times [(K \cup \{\text{h}, \text{“yes”}, \text{“no”}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}]$.

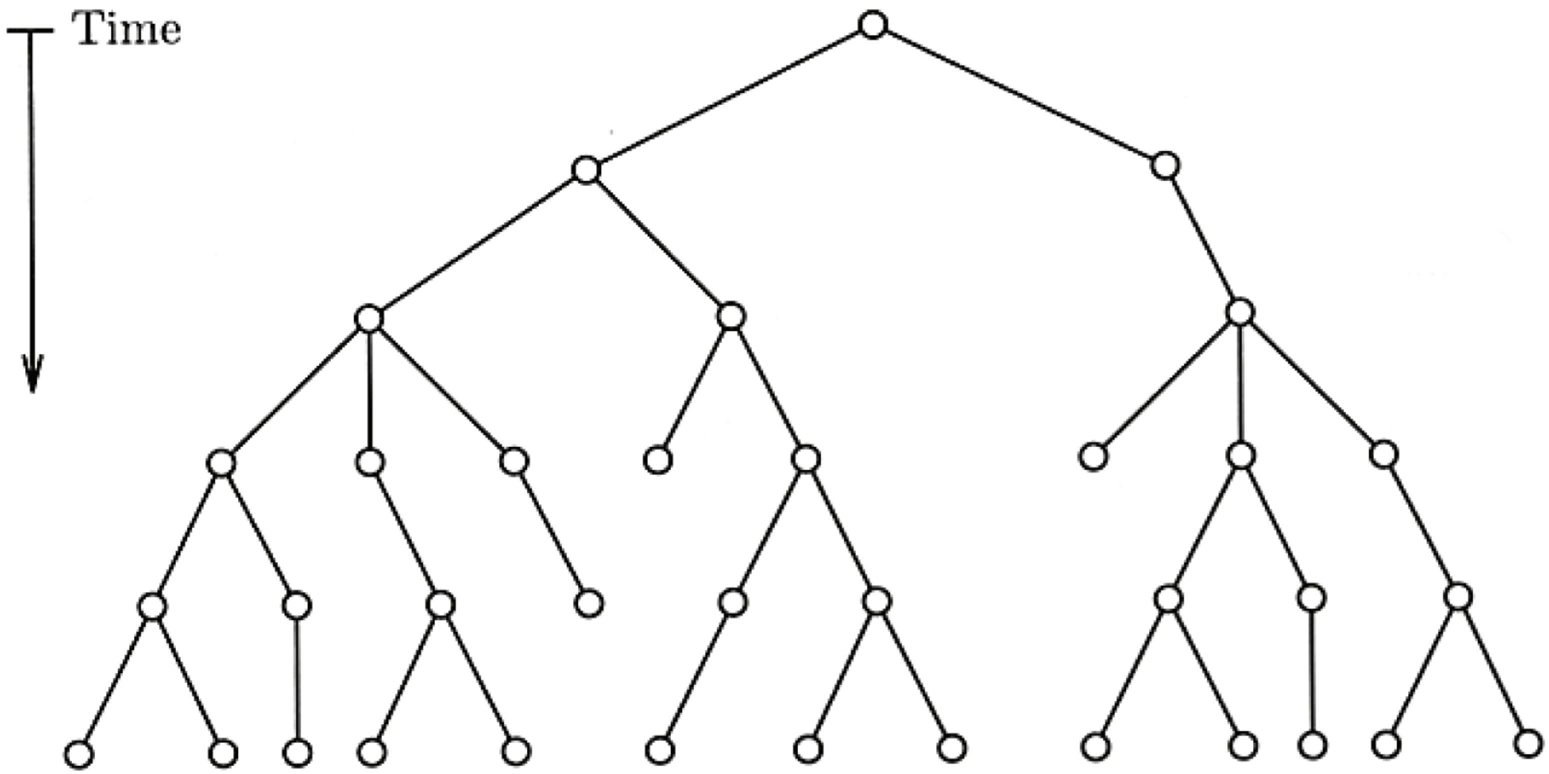


Figure 2-9. Nondeterministic computation.

1. N decides a language L if for any $x \in \Sigma^*$, $x \in L$ if and only if $(s, \triangleright, x) \xrightarrow{N^*} (\text{"yes"}, w, u)$ for some strings w and u.
2. An input is accepted if there is some sequence of nondeterministic choice that results in "yes".

N decides L in time $f(n)$ if

1. N decides L;
2. for any $x \in \Sigma^*$, if $(s, \triangleright, x) \xrightarrow{N^k} (\text{“yes”}, w, u)$, then $k \leq f(|x|)$.

Let NTIME($f(n)$) be the set of languages decided by NTMs within time f.

Let $\mathcal{NP} = \bigcup_{k \geq 1} \mathrm{NTIME}(n^k)$.
We have

$$\mathcal{P} \subseteq \mathcal{NP}.$$

Example 2.9

$TSP(D) \in \mathcal{NP}$

1. Write out arbitrary permutation of $1, \ldots, n$.
2. Check whether the tour indicated by this permutation is less than the distance bound.

Theorem 2.6

Suppose that language L is decided by an NTM N in time $f(n)$. Then it is decided by a 3-string DTM M in time $O(c^{f(n)})$, where $c > 1$ is some constant depending on N.

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

Example 2.10

- Reachability $\in$ NSPACE($\lg n$) (This is easy.)
- Reachability $\in$ SPACE($(\lg n)^2$) (In Chapter 7.)

Why employ nondeterminism?

Exercises

2.8.1, 2.8.4, 2.8.6, 2.8.7, 2.8.8, 2.8.9, 2.8.10, 2.8.11