CS 4104: Data and Algorithm Analysis

Clifford A. Shaffer

Department of Computer Science
Virginia Tech
Blacksburg, Virginia

Fall 2010

A General Model

Want a general model of computation that is as simple as possible.

- Wish to be able to reason about the model.
- “State machines” are simple.

Necessary features:

- Read
- Write
- Compute

Turing Machines (1)

A tape, divided into squares.

“States”

A single I/O head:

- Read current symbol
- Change current symbol

Control Unit Actions:

- Put the control unit into a new state.
- Either:
  1. Write a symbol in current tape square.
  2. Move I/O head one square left or right.

Turing Machines (2)

Tape has a fixed left end, infinite right end.

- Machine ceases to operate if head moves off left end.
- By convention, input is placed on left end of tape.

A **halt** state (*h*) signals end of computation.

"#" indicates a blank tape square.

Formal definition of Turing Machine

A **Turing Machine** is a quadruple (K, Σ, δ, s) where

- K is a finite set of **states** (not including h).
- Σ is an alphabet (containing #, not L or R).
- $s \in K$ is the **initial** state.
- δ is a function from $K \times \Sigma$ to $(K \cup \{h\}) \times (\Sigma \cup \{L, R\})$.

If $q \in K$, $a \in \Sigma$ and $\delta(q, a) = (p, b)$, then when in state q and scanning a, enter state p and

1. If $b \in \Sigma$ then replace a with b.
2. Else (b is L or R): move head.

Turing Machine Example 1

$M = (K, \Sigma, \delta, s)$ where

- $K = \{q_0, q_1\}$,
- $\Sigma = \{a, \#\}$,
- $s = q_0$,
- $\delta =$

q	σ	$\delta(q, \sigma)$
q_0	a	$(q_1, \#)$
q_0	$\#$	$(h, \#)$
q_1	a	(q_0, a)
q_1	$\#$	(q_0, R)

Turing Machine Example 2

$M = (K, \Sigma, \delta, s)$ where

- $K = \{q_0\}$,
- $\Sigma = \{a, \#\}$,
- $s = q_0$,
- $\delta =$

q	σ	$\delta(q, \sigma)$
q_0	a	(q_0, L)
q_0	$\#$	$(h, \#)$

Notation

Configuration: $(q, aaba\#\underline{\#}a)$

Halted configuration: q is h.

Hanging configuration: Move left from leftmost square.

A **computation** is a sequence of configurations for some $n \geq 0$. Such a computation is of **length** n.

Execution

Execution on first machine example.

$$
\begin{aligned}
(q_0, \underline{a}aaa) &\vdash_M (q_1, \underline{\#}aaa)\\
&\vdash_M (q_0, \#\underline{a}aa)\\
&\vdash_M (q_1, \#\underline{\#}aa)\\
&\vdash_M (q_0, \#\#\underline{a}a)\\
&\vdash_M (q_1, \#\#\underline{\#}a)\\
&\vdash_M (q_0, \#\#\#\underline{a})\\
&\vdash_M (q_1, \#\#\#\underline{\#})\\
&\vdash_M (q_0, \#\#\#\#\underline{\#})\\
&\vdash_M (h, \#\#\#\#\underline{\#})
\end{aligned}
$$

Computations

- M is said to **halt on input** w iff $(s, \#w\underline{\#})$ yields some halted configuration.
- M is said to **hang on input** w if $(s, \#w\underline{\#})$ yields some hanging configuration.
- Turing machines compute functions from strings to strings.
- Formally: Let f be a function from Σ_0^* to Σ_1^*. Turing machine M is said to **compute** f if for any $w \in \Sigma_0^*$, if $f(w) = u$ then

$$(s, \#w\underline{\#}) \vdash_M^* (h, \#u\underline{\#}).$$

- f is said to be a **Turing-computable function**.
- Multiple parameters: $f(w_1, ..., w_k) = u$, $(s, \#w_1\#w_2\#...\#w_k\underline{\#}) \vdash_M^* (h, \#u\underline{\#})$.

Functions on Natural Numbers

- Represent numbers in **unary** notation on symbol I (zero is represented by the empty string).
- $f : \mathbb{N} \rightarrow \mathbb{N}$ is computed by M if M computes $f' : \{I\}^* \rightarrow \{I\}^*$ where $f'(I^n) = I^{f(n)}$ for each $n \in \mathbb{N}$.
- Example: $f(n) = n + 1$ for each $n \in \mathbb{N}$.

q	σ	$\delta(q, \sigma)$
q_0	I	(h, R)
q_0	$\#$	(q_0, I)

$$(q_0, \#II\underline{\#}) \vdash_M (q_0, \#II\underline{I}) \vdash_M (h, \#III\underline{\#}).$$

- In general, $(q_0, \#I^n\underline{\#}) \vdash_M^* (h, \#I^{n+1}\underline{\#})$.
- What about $n = 0$?

Turing-decidable Languages

A language $L \subset \Sigma_0^*$ is **Turing-decidable** iff function $\chi_L : \Sigma_0^* \to \{\boxed{\text{Y}}, \boxed{\text{N}}\}$ is Turing-computable, where for each $w \in \Sigma_0^*$,

$$\chi_L(w) = \begin{cases} \boxed{\text{Y}} & \text{if } w \in L \\ \boxed{\text{N}} & \text{otherwise} \end{cases}$$

Ex: Let $\Sigma_0 = \{a\}$, and let $L = \{w \in \Sigma_0^* : |w| \text{ is even}\}$.

M erases the marks from right to left, with current parity encode by state. Once blank at left is reached, mark $\boxed{\text{Y}}$ or $\boxed{\text{N}}$ as appropriate.

Turing-acceptable Languages

M **accepts** a string w if M halts on input w.

- M accepts a language iff M halts on w iff $w \in L$.
- A language is **Turing-acceptable** if there is some Turing machine that accepts it.

Ex: $\Sigma_0 = \{a, b\}$, $L = \{w \in \Sigma_0^* : w$ contains at least one $a\}$.

q	σ	$\delta(q, \sigma)$
q_0	a	(h, a)
q_0	b	(q_0, L)
q_0	$\#$	(q_0, L)

Every Turing-decidable language is Turing-acceptable.

Combining Turing Machines

Lemma: If

$$(q_1, w_1\underline{a_1}u_1) \vdash_M^* (q_2, ww_2\underline{a_2}u_2)$$

for string w and

$$(q_2, w_2\underline{a_2}u_2) \vdash_M^* (q_3, w_3\underline{a_3}u_3),$$

then

$$(q_1, w_1\underline{a_1}u_1) \vdash_M^* (q_3, ww_3\underline{a_3}u_3).$$

Insight: Since $(q_2, w_2\underline{a_2}u_2) \vdash_M^* (q_3, w_3\underline{a_3}u_3)$, this computation must take place without moving the head left of w_2

- The machine cannot “sense” the left end of the tape

Combining Turing Machines (Cont)

Thus, the head won't move left of w_2 even if it is not at the left end of the tape.

This means that Turing machine computations can be combined into larger machines:

- M_2 prepares string as input to M_1.
- M_2 passes control to M_1 with I/O head at end of input.
- M_2 retrieves control when M_1 has completed.

Some Simple Machines

Basic machines:

- $|\Sigma|$ symbol-writing machines (one for each symbol).
- Head-moving machines R and L move the head appropriately.

More machines:

- First do M_1, then do M_2 or M_3 depending on current symbol.
- (For $\Sigma = \{a, b, c\}$) Move head to the right until a blank is found.
- Find first blank square to left: $L_{\#}$
- Copy Machine: Transform $\#w\underline{\#}$ into $\#w\#w\underline{\#}$.
- Shift a string left or right.

Extensions

The following extensions do not increase the power of Turing Machines.

- 2-way infinite tape
- Multiple tapes
- Multiple heads on one tape
- Two-dimensional “tape”
- Non-determinism