Examples of Turing Machines

Higher level descriptions

- We can give a formal description to a particular TM by specifying each of its seven components
- This way a TM can become cumbersome.

  Note: To avoid this we use higher level descriptions which are precise enough for the purpose of understanding
- However, every higher level description is actually just a short hand for its formal counterpart.

Contrast: identify a similar situation with real com

Example 1

Describe a TM M_2 that recognizes the language $A = \{0^{2^n} | n \geq 0\}$

M_2 = "On input string w:

1. Sweep left to right across the tape crossing off every other 0
2. If in stage 1 tape contained a single 0, *accept*
3. If in stage 1 tape contained more that a single 0 and the number of 0s was odd, *reject*
4. Return the head to the left-hand of the tape
5. Go to stage 1"

Analysis

- At each iteration, stage 1 cuts the number of 0s in half.
- If the resulting number of 0s is odd and greater than one, the original number could not have been a power of 2 and machine rejects
- If the number of 0 is one than the original number of zeros must have been a power of 2, so machine accepts.

Rationale: $\forall n \in N[(\ldots(n/2)/2\ldots/2)\ldots) = \frac{n}{2^n}]$

Hence, if $\frac{n}{2^n} = 1$ it means that $n = 2^n$.

Formal description of M_2

$M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$ where:

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, \sqcup\}$
- δ is described in Figure 1
- The start, accept, reject are q_1, q_{accept}, q_{reject} respectively

State diagram of M_2

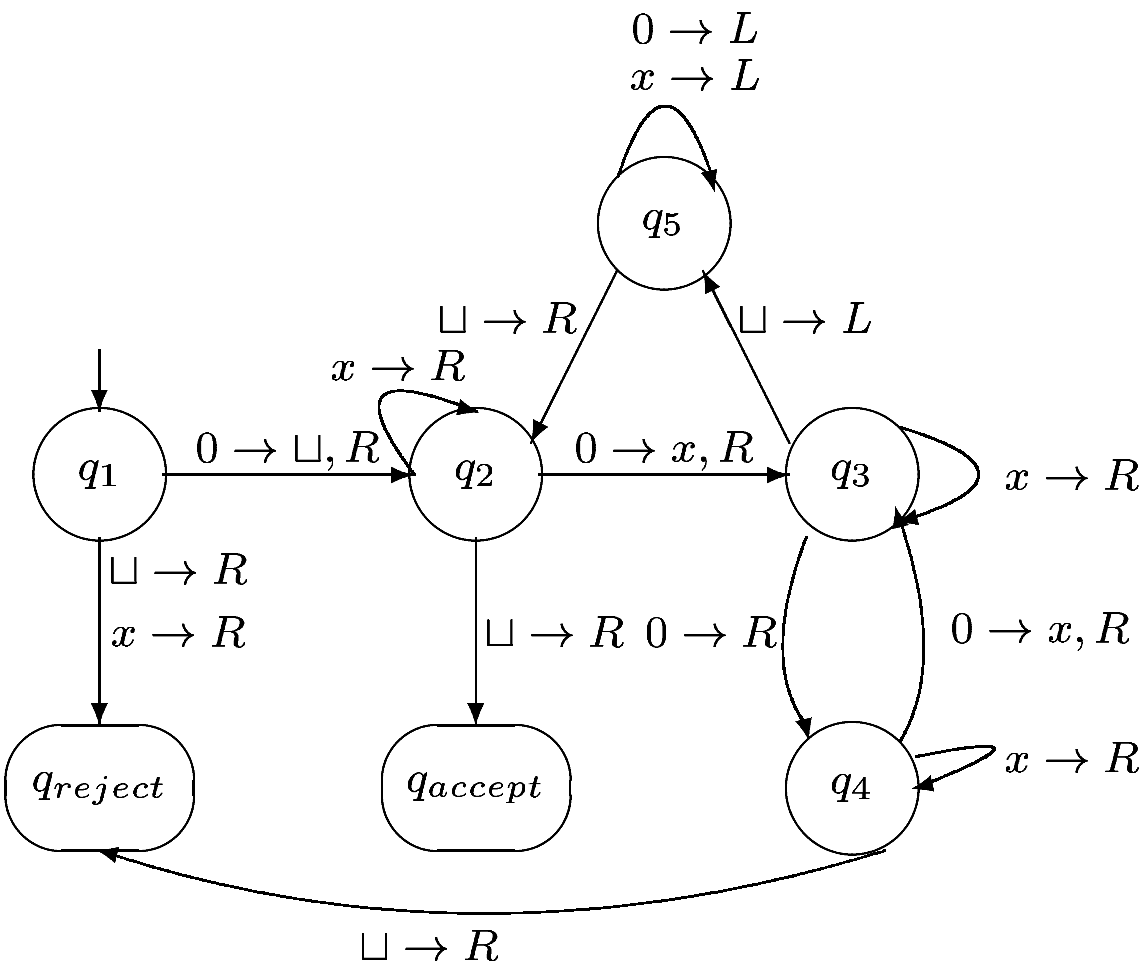

Figure 1: M_2's state transition diagram

Notations

- $\delta(q_i, a) = (q_j, b, R)$ is denoted by an arrow that starts at q_i, ends at q_j, and is labeled by $a \rightarrow b, R$
- $\delta(q_i, a) = (q_j, b, L)$ is denoted by an arrow that starts at q_i, ends at q_j, and is labeled by $a \rightarrow b, L$
- $\delta(q_i, a) = (q_j, a, R)$ is denoted by an arrow that starts at q_i, ends at q_j, and is labeled by $a \rightarrow R$
- $\delta(q_i, a) = (q_j, a, L)$ is denoted by an arrow that starts at q_i, ends at q_j, and is labeled by $a \rightarrow L$

Example run

On input $w = 0000$:

$q_1 0000$	$\sqcup q_2 000$	$\sqcup x q_3 00$	$\sqcup x 0 q_4 0$	$\sqcup x 0 x q_3 \sqcup$
$\sqcup x 0 q_5 x \sqcup$	$\sqcup x q_5 0 x \sqcup$	$\sqcup q_5 x 0 x \sqcup$	$q_5 \sqcup x 0 x \sqcup$	$\sqcup q_2 x 0 x \sqcup$
$\sqcup x q_2 0 x \sqcup$	$\sqcup x x q_3 x \sqcup$	$\sqcup x x x q_3 \sqcup$	$\sqcup x x q_5 x \sqcup$	$\sqcup x q_5 x x \sqcup$
$\sqcup q_5 x x x \sqcup$	$q_5 \sqcup x x x \sqcup$	$\sqcup q_2 x x x \sqcup$	$\sqcup x q_2 x x \sqcup$	$\sqcup x x q_2 x \sqcup$
$\sqcup x x x q_2 \sqcup$	$\sqcup x x x \sqcup q_a$			

Comments

- The arrow labeled $0 \to \sqcup, R$ in q_1 means $\delta(q_1, 0) = (q_2, \sqcup, R)$ i.e., in state q_1 with head reading 0, the machine goes to q_2, writes $\sqcup$, and moves to right
- The arrow labeled $0 \to R$ in q_3 means $\delta(q_3, 0) = (q_4, 0, R)$: M_2 moves to the right when reading a 0 without affecting the tape.

Note: This machines begins by writing a blank over the leftmost zero.

- This allows it to find the left-end of the tape in stage 4
- It also allows M_2 to identify the case when tape contains one zero only, in stage 2

Example 2

$M_1 = (Q, \Sigma, \Gamma, \delta, q_1, q_a, q_r)$ is the TM that decides the language $B = \{w\#w | w \in \{0, 1\}^*\}$

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_a, q_r\}$
- $\Sigma = \{0, 1, \#\}$, $\Gamma = \{0, 1, \#, x, \sqcup\}$
- δ is described in Figure 2
- Start, accept, and reject states are q_1, q_a, q_r, respectively

High-level description of M_1

M_1 = "On input w:

1. Scan the input tape to be sure that it contains a single $\#$. If not, *reject*
2. Zig-zag across the tape to corresponding positions on either side of $\#$ to check whether these positions contain the same symbol. If they do not, *reject*. Cross off the symbols as they are checked
3. When all symbols to the left of $\#$ have been crossed off, check for the remaining symbols to the right of $\#$. If any symbol remain, *reject*; otherwise *accept*"

Note: High-level descriptions of TM-s are also called *implementation descriptions*.

Turing machine M_1

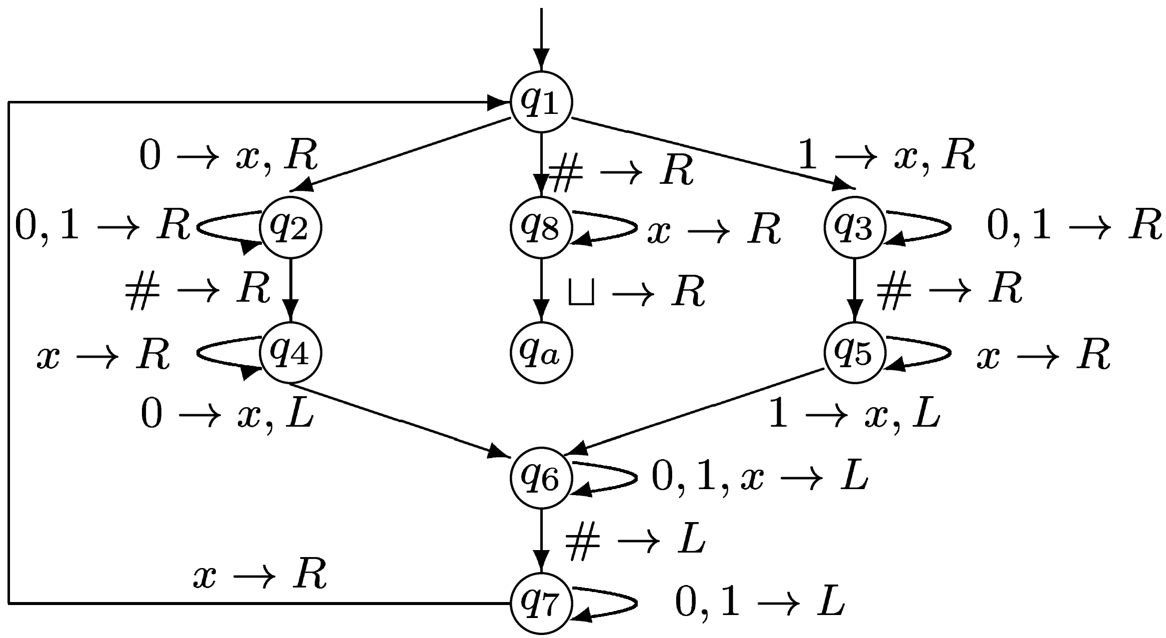

Figure 2: State diagram for TM M_1

More notations

- Transitions $0, 1 \to R$ in states q_2 and q_3 means that machines moves to the right as long as 0 or 1 is on the tape.
- The machine starts by writing a blank symbol to delimit the left-hand edge of the tape
- Stage 1 is implemented by states q_1 through q_7: q_2,q_4,q_6 if the first symbol of input is 0, and q_3, q_5, q_7 if the first input symbol was 1.
- To simplify the figure we don't show the reject state or transitions going to reject state. These transitions occur implicitly whenever a state lacks an outgoing transition for a particular symbol. Example, q_5 on # is such a transition

Note: using different states for input starting with 1 and 0 allows M_1 to implement the matching operation

Note

- The transition diagram in Figure 2 is rather complex.
- One can understand better what happens from the high-level description than from Figure 2.
- Therefore further we will replace transition diagrams by high-level descriptions, as initially suggested

Example 3

M_3 is a Turing machine that performs some elementary arithmetic. It decides the language $C = \{a^i b^j c^k | i \times j = k, i, j, k \geq 1\}$
M_3="On input string w

1. Scan the input from left to right to be sure that it is a member of $a^+b^+c^+$; reject if it is not
2. Return the head at the left-hand end of the tape
3. Cross off an a and scan to the right until a b occurs. Shuttle between the b's and c's crossing off one of each until all b's are gone. If all c's have been crossed of and some b's remain *reject*.
4. Restores the crossed off b's and repeat stage 3 if there is another a to cross off. If all a's are crossed off, determine whether all c's are crossed off. If yes *accept*, otherwise *reject*."

Analyzing M_3

- In stage 1 M_3 operates as a finite automaton; no writing is necessary as the head moves from left to right:
  1. $\delta(q_1, a) = (q_1, a, R)$,$\delta(q_1, b) = (q_2, b, R)$, $\delta(q_1, c) = (q_3, c, R)$
  2. $\delta(q_2, b) = (q_2, b, R)$, $\delta(q_2, a) = reject$, $\delta(q_2, c) = (q_2, c, R)$
  3. $\delta(q_3, c) = (q_3, c, R)$, $\delta(q_3, b) = reject$, $\delta(q_3, a) = reject$

Stage 2 finding the left-hand end

- Mark the left-hand end by writing a ⊔ before the input (this have been seen before)
- Note that if the machine tries to move the head to the left of the left-hand end of the tape the head remains in the same place. This feature can be made "the left-hand end detector" by:
  1. Write a special symbol over the current position, while recording the symbol that it replaced in the control
  2. Attempt to move to the left. If the head is still over the special symbol, the leftward move did not succeed, and the head must have been at the left-hand end. If the head is over a different symbol, some symbols are to the left of that position on the tape

Note

Stage 3 and stage 4 of M_3 have straightforward implementations

Element distinctness problem

Given a list of strings over $\{0, 1\}$ separated by $\#$, determine if all strings are different.
A TM that solves this problem accepts the language

$$E = \{\#x_1\#x_2\# \dots \#x_k | x_i \in \{0, 1\}^*, x_i \neq x_j \ for \ i \neq j\}$$

Example 4

$M_4 = (Q, \Sigma, \Gamma, \delta, q_s, q_a, q_r)$ is the TM that solves the *element distinctness problem*

M_4 works by comparing x_1 with $x_2, \ldots, x_k$, then by comparing x_2 with $x_3, \ldots, x_k$, and so on

Informal description

M_4="On input w:

1. Place a mark on top of the leftmost tape symbol. If that symbol was a blank, *accept*. If that symbol was a $\#$ continue with the next stage. Otherwise *reject*.
2. Scan right to the next $\#$ and place a second mark on top of it. If no $\#$ is encountered before a blank symbol, only x_1 was present, so *accept*.
3. By zig-zagging, compare the two strings to the right of the marked $\#$-s. If they are equal, *reject*
4. Move the rightmost of the two marks to the next $\#$ symbol to the right. If no $\#$ symbol is encountered before a blank symbol, move the leftmost mark to the next $\#$ to its right and the rightmost mark to the $\#$ after that. If no $\#$ is available for the rightmost mark, all strings have been compared, so *accept*.

Marking tape symbols

- In stage two the machine places a mark above a symbol, $\#$ in this case.
- In the actual implementation the machine has two different symbols, $\#$ and $\dot{\#}$ in the tape alphabet Γ
- Thus, when machine places a mark above symbol x it actually writes the marked symbol of x at that location
- Removing the mark means write the symbol at the location where the marked symbol was.