Recall that a finite state acceptor was given by (S, A, i, t, F)

set of states, alphabet, initial state, transition mapping, set of finishing states

with the transition mapping being given by $t : S \times A \to S$.

By contrast, for a Turing machine the transition mapping is of the form $t : S \times \tilde{A} \to S \times \tilde{A} \times \{L, R\}$

indicates the Turing machine can write

indicates the Turing machine's head can move left or right.

Def A Turing machine is a 7-tuple $(S, A, \tilde{A}, t, i, s_{accept}, s_{reject})$, where $S, A, \tilde{A}$ are finite sets and

(a) S is the set of states

(b) A is the input alphabet not containing the blank symbol $\sqcup$

(c) $\tilde{A}$ is the tape alphabet, where $\sqcup \in \tilde{A}$ and $A \subseteq \tilde{A}$.

(d) $t : S \times \tilde{A} \to S \times \tilde{A} \times \{L, R\}$ is the transition mapping

(e) i is the initial state of the machine.

(f) $s_{accept} \in S$ is the accept state.

(g) $s_{reject} \in S$ is the reject state and $s_{accept} \neq s_{reject}$.

Remarks about the definition

1) Since A does not contain the blank symbol $\sqcup$, the first blank on the tape marks the end of the input string.

2) If the Turing machine is instructed to move left, and it has reached the first cell of the tape, then it stays at the first cell.

3) The Turing machine continues to compute until it enters either the accept or reject states at which point it halts. If it doesnot enter either, then it goes on forever.

Example (considered again) $A = \{0, 1\}$ $L = \{0^m 1^m \mid m \in \mathbb{N}, m \geq 1\}$

We need to be able to write down the transition mapping hence the set of states S. Recall that what we gave was an algorithm, and using that algorithm we processed strings to convince ourselves that the corresponding Turing machine behaved correctly.

Here is the algorithm again:

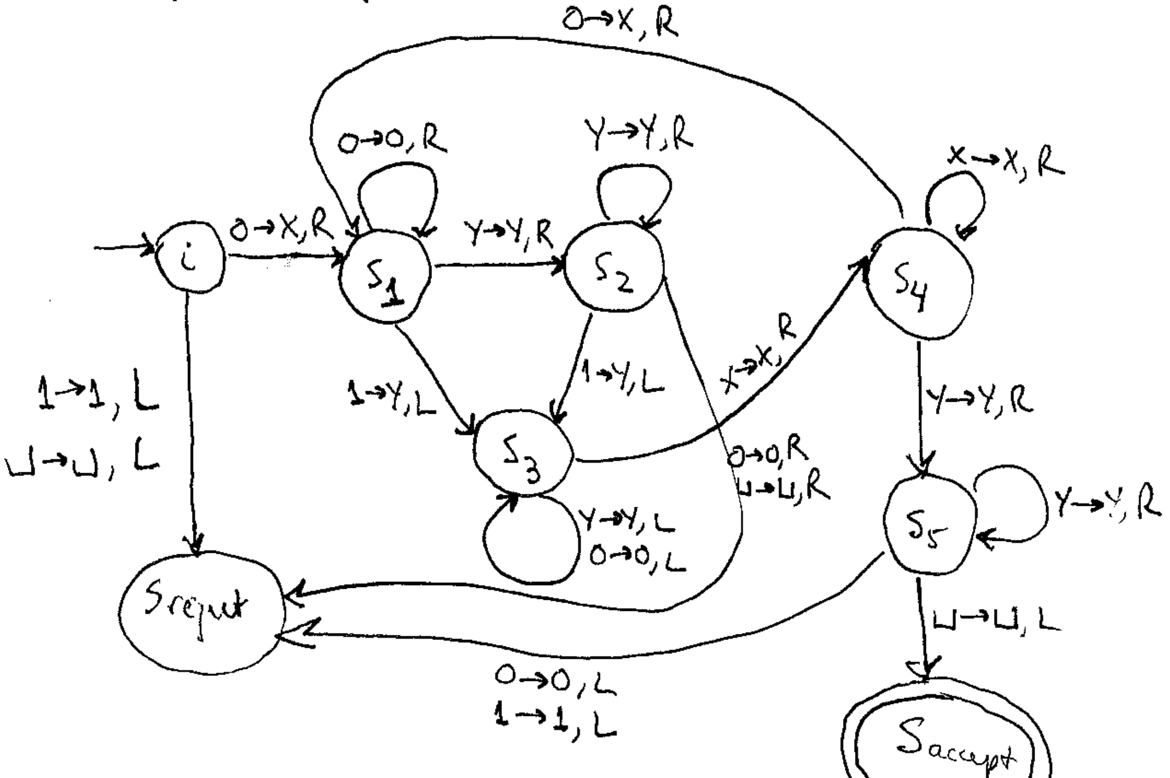
The tape head is initially positioned over the first cell.

1. If anything other than 0 is in the first cell, then REJECT.
2. If 0 is in the cell, then change 0 to X.
3. Move right to the first 1. If none, then REJECT.
4. Change 1 to Y.

5. Move left to the leftmost 0. If none, move right looking for either a 0 or a 1. If either 0 or 1 is found before the first blank symbol, then REJECT; otherwise, ACCEPT.

6. Go to step 2.

Before we can write down the set of states S or the transition mapping t, let us draw a transition diagram which is the Turing machine equivalent to drawing a finite state acceptor when we looked at regular languages.



$i \to S_{reject}$ represents step 1 of the algorithm

$i \to S_1$ and $S_4 \to S_1$ represent step 2 of the algorithm ($i \to S_1$ at the first pass through the string; $S_4 \to S_1$ at subsequent passes)

$S_1 \to S_1$, $S_1 \to S_2$, $S_2 \to S_2$ represent the first part of step 3.

$S_2 \to S_{reject}$ represents the second part of step 3.

$S_1 \to S_3$ and $S_2 \to S_3$ represent step 4.

$S_3 \to S_3$ and $S_3 \to S_4$ represent the first sentence in step 5.

$S_4 \to S_4$, $S_4 \to S_5$, $S_5 \to S_5$ represent the second sentence in step 5.

$S_5 \to S_{reject}$ is the first half of the third sentence in step 5.

$S_5 \to S_{accept}$ is the second half of the third sentence in step 5.
$S_4 \to S_1$ represents step 6.
We have accounted for all pieces of our algorithm. Therefore, we have written down a Turing machine where $A = \{0,1\}$, $\tilde{A} = \{0,1,x,y,\sqcup\}$
blank symbol

$S = \{i, s_{accept}, s_{reject}, s_1, s_2, s_3, s_4, s_5\}$
i is the initial state; $s_{accept} \in S$ is the accept state; $s_{reject} \in S$ is the reject state.
We just have to write down the transition mapping $t: S \times \tilde{A} \to S \times \tilde{A} \times \{L, R\}$

$t(i,0) = (s_1, X, R)$
$t(i,1) = (s_{reject}, 1, L)$
$t(i,\sqcup) = (s_{reject}, \sqcup, L)$

These are the only 3 transitions possible out of state i, but $t: S \times \tilde{A} \to S \times \tilde{A} \times \{L,R\}$ so technically, to write down the full transition mapping, we must assign triplets in $S \times \tilde{A} \times \{L,R\}$ even to inputs from $\tilde{A}$ that cannot occur when in i:

$t(i,x) = (s_{reject}, x, L)$ ← we assign s_{reject}, same element of $\tilde{A}$, and one of the allowable tape head directions
$t(i,y) = (s_{reject}, y, L)$

Technically, the Turing machine halts when it enters either an accepting state (s_{accept}) or a rejecting state (s_{reject}), so in practice we can define $\tilde{S} = \{i, s_1, s_2, s_3, s_4, s_5\} = \underbrace{S \setminus \{s_{accept}, s_{reject}\}}_{\text{set of nonhalting states}}$

and $t: \tilde{S} \times \tilde{A} \to S \times \tilde{A} \times \{L,R\}$, so we avoid writing down the transitions from s_{accept} and s_{reject}.
We only have states s_1, s_2, s_3, s_4, and s_5 left.

$t(s_1, 0) = (s_1, 0, R)$
$t(s_1, Y) = (s_2, Y, R)$
$t(s_1, 1) = (s_3, Y, L)$

on the diagram

$t(s_1, X) = (s_{reject}, X, R)$
$t(s_1, \sqcup) = (s_{reject}, \sqcup, R)$

not on the diagram; cannot occur, so added for completeness

$$\left.\begin{aligned}
t(s_2, Y) &= (s_2, Y, R)\\
t(s_2, 1) &= (s_3, Y, L)\\
t(s_2, 0) &= (s_{reject}, 0, R)\\
t(s_2, \sqcup) &= (s_{reject}, \sqcup, R)
\end{aligned}\right\}\ \text{on the diagram; can occur}$$

$$t(s_2, X) = (s_{reject}, X, R) \leftarrow \text{not on the diagram; cannot occur; added for completeness}$$

$$\left.\begin{aligned}
t(s_3, Y) &= (s_3, Y, L)\\
t(s_3, 0) &= (s_3, 0, L)\\
t(s_3, X) &= (s_4, X, R)
\end{aligned}\right\}\ \text{on the diagram; can occur}$$

$$\left.\begin{aligned}
t(s_3, \sqcup) &= (s_{reject}, \sqcup, R)\\
t(s_3, 1) &= (s_{reject}, 1, R)
\end{aligned}\right\}\ \text{not on the diagram; cannot occur; added for completeness}$$

$$\left.\begin{aligned}
t(s_4, X) &= (s_4, X, R)\\
t(s_4, Y) &= (s_5, Y, R)\\
t(s_4, 0) &= (s_1, X, R)
\end{aligned}\right\}\ \text{on the diagram; can occur}$$

$$\left.\begin{aligned}
t(s_4, 1) &= (s_{reject}, 1, R)\\
t(s_4, \sqcup) &= (s_{reject}, \sqcup, R)
\end{aligned}\right\}\ \text{not on the diagram; cannot occur; added for completeness}$$

$$\left.\begin{aligned}
t(s_5, Y) &= (s_5, Y, R)\\
t(s_5, \sqcup) &= (s_{accept}, \sqcup, L)\\
t(s_5, 0) &= (s_{reject}, 0, L)\\
t(s_5, 1) &= (s_{reject}, 1, L)
\end{aligned}\right\}\ \text{on the diagram; can occur}$$

$$t(s_5, X) = (s_{reject}, X, L) \leftarrow \text{not on the diagram; cannot occur; added for completeness.}$$

Moral of the story

The transition mapping is a very inefficient way of specifying a Turing machine as a lot of transitions cannot occur unlike what we saw for a finite state acceptor, where the input alphabet was exactly the alphabet of the language. Here $A \subset \tilde{A}$. Therefore, we will specify a Turing machine via either an algorithm or the transition diagram <u>only</u>.

To figure out which languages are recognized by a Turing machine, we need to introduce the notion of a configuration. As a Turing machine goes through its computations, changes take place in

1. The state of the machine
2. the tape contents
3. the tape head location

} a setting of these three items is called a <u>configuration</u>

<u>Representing configurations</u> We represent a configuration as us_iv, where u, v are strings in the tape alphabet $\tilde{A}$ and s_i is the current state of the machine. The tape contents are then the string uv and the current location of the tape head is on the first symbol of v. The assumption here is that the tape contains only blanks after the last symbol in v.