Turing Machines

Task: Take a more realistic look at the model of a computer than a finite state acceptor.

Turing machines were first proposed by Alan Turing in 1936 in order to explore the theoretical limits of computation. We shall see that certain problems cannot be solved even by a Turing machine and are thus beyond the limits of computation.

A Turing machine is similar to a finite state acceptor but has **unlimited memory** given by an **infinite tape** (countably infinite). The tape is divided into **cells** each of which contains a character of a **tape alphabet**.

The Turing machine is equipped with a **tape head** that can read and write symbols on the tape and move left (back) or right (forward) on the tape. Initially, the tape contains only the input string and is blank everywhere else. To store information, the Turing machine can write this information on the tape. To read information that it has written, the Turing machine can move its head back over it.

The Turing machine continues computing until it decides to produce an output. The outputs "**accepts**" and "**rejects**" are obtained by entering accepting or rejecting states respectively. It is also possible for the Turing machine to go on forever if it does not enter either an accepting nor rejecting state.

[0][1][0][_][_][... The **blank symbol** (_) is part of the tape alphabet.

Example: Let A = {0, 1} and L = $\{0^m1^m \mid m \in N, m \geq 1\}$
We know L is not a regular language, so there is no finite state acceptor that can recognise it, but there is a Turing machine that can.

Initial State of the Tape:
Input string of 0s and 1s, then infinitely many blanks.

Idea of this Turing machine:
Change a 0 to an X, and a 1 to a Y until either:

a) All 0s and 1s have been matched - ACCEPT
b) The 0s and 1s do **not** match or the string does not have the form 0*1* - REJECT

Algorithm:
The tape head is initially positioned over the first cell.

1. If anything other than 0 is in the first cell - REJECT
2. If 0 is in the cell, then change 0 to X.
3. Move right to the first 1. If none - REJECT
4. Change 1 to Y.
5. Move left to the leftmost 0. If none, move right looking for either a 0 or 1.
   If either a 0 or 1 is found before the first blank symbol - REJECT
   Otherwise - ACCEPT
6. Go to STEP 2.

Examples of processing strings: (Tape Head)

0011_ X011_ X011_ X0Y1_ X0Y1_ XXY1_ XXY1_ XXYY_ XXYY_ XXYY_ ACCEPT (STEP 5)	001_ X01_ X01_ X0Y_ X0Y_ XXY_ XXY_ REJECT (STEP 3)	011_ X11_ X11_ XY1_ XY1_ XY1_ REJECT (STEP 5)	010_ X10_ X10_ XY0_ XY0_ XY0_ REJECT (STEP 5)

Note that we have the following:

- A = {0, 1} is the **input alphabet**.
- _ ∉ A, where _ is the **blank symbol**.
- Ã = {0, 1, X, Y, _) is the **tape alphabet**.
- S is the set of **states**.

Note that the tape head is moving right or left, so we also need to have a set {L, R} with L for left and R for right for specifying where the tape head goes.

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

- S = Set of States
- A = Alphabet
- i = Initial State
- t = Transition Mapping
- F = State of Finishing States

The transition mapping is given by $t : S \times A \rightarrow S$

By contrast, a Turing machine's transition mapping is of the form $t : S \times \tilde{A} \rightarrow S \times \tilde{A} \times \{L, R\}$

- $\tilde{A}$ = Indicates what the Turing machine can write
- {L, R} = Indicates the Turing machine can move left or right

Definition: A Turing machine is a 7-tuple $(S, A, \tilde{A}, t, i, S_{acc}, S_{rej})$ where S, A, $\tilde{A}$ are finite sets.

- S = Set of States
- A = Input Alphabet **not** containing the blank symbol _
- $\tilde{A}$ = Tape Alphabet where $\_ \in \tilde{A}$ and $A \subseteq \tilde{A}$
- t = Transition Mapping
- i = Initial State
- S_{acc} = Accept State
- S_{rej} = Reject State

Remarks:

1. Since A does not contain the blank symbol _, 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 does not enter either state, it goes on forever.

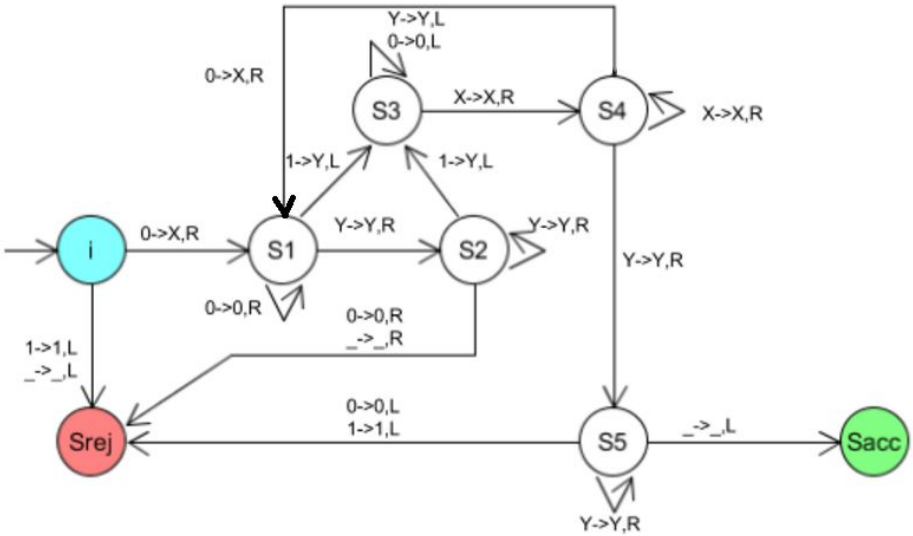
Example: (Same example as previous)
$A = \{0, 1\}$ and $L = \{0^m1^m \mid m \in N, m \geq 1\}$

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.

Algorithm:
The tape head is initially positioned over the first cell.

1. If anything other than 0 is in the first cell - REJECT
2. If 0 is in the cell, then change 0 to X.
3. Move right to the first 1. If none - REJECT
4. Change 1 to Y.
5. Move left to the leftmost 0. If none, move right looking for either a 0 or 1.
   If either a 0 or 1 is found before the first blank symbol - 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 of drawing a finite state acceptor.



- $i \Rightarrow S_{rej}$ represents step 1 of the algorithm.
- $i \Rightarrow S_1$ and $S_4 \Rightarrow S_1$ represent step 2 of the algorithm.
  ($i \Rightarrow S_1$ at the first pass through the string, $S_4 \Rightarrow S_1$ at subsequent passes)
- $S_1 \Rightarrow S_1$, $S_1 \Rightarrow S_2$ and $S_2 \Rightarrow S_2$ represent the first part of step 3.
- $S_2 \Rightarrow S_{rej}$ represents the second part of step 3.
- $S_1 \Rightarrow S_3$ represents step 4.
- $S_3 \Rightarrow S_3$ and $S_3 \Rightarrow S_4$ represent the first sentence in step 5.
- $S_4 \Rightarrow S_4$ and $S_4 \Rightarrow S_5$, $S_5 \Rightarrow S_5$ represent the second sentence in step 5.
- $S_5 \Rightarrow S_{rej}$ represents the first half of the third sentence in step 5.
- $S_5 \Rightarrow S_{acc}$ represents the second half of the third sentence in step 5.
- $S_4 \Rightarrow S_1$ represents the step 6.

We have accounted for all pieces of our algorithm, therefore we have written down a Turing machine when A = {0, 1}, Ã = {0, 1, X, Y, _ }