Turing Machines

Introduction

TASK: look at a more realistic model of a computer than a finite state acceptor.

Turing Machines were first proposed by Allen Turing in 1936, in order to explore the theoretical limits of computation. We'll see an example of a problem that cannot be solved by a Turing machine, i.e beyond the limits of computation.

Turing machines are similar to a finite state acceptor, but with unlimited memory given by an infinite tape (countably infinite). The tape is divided into cells, each of which holds a character of the tape alphabet *(radix)*. 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)*. Initially the tape contains only the input string and is blank everywhere else. To store information, the turing machine can write this information on to the tape.

To read information that it has written, the turing machine can move its head back over the tape. 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 *(infinite loop)* if it does not enter either an accepting or rejecting state.

Illustration of a turing machine

[0][1][0][_][_][_] <- Tape
------^ <- Tape head\

Example let A = {0,1} and let L = $\{0^m 1^m | m \in N, m \geq 1\}$

We know L is not a regular language, so there is no finite state acceptor that can recognize it, but there is a turing machine that can.

Initiated state of the Tape: Input string of 0's and 1's, then infinite blanks.

Idea of this Turing machine: Change a 0 to an X, and a 1 to a Y until either all 0's and 1's have been matched hence **ACCEPT**, or the 0's and 1's do not match or the string does not have the form 0*1* hence **REJECT**

Algorithm

The tape head is initially positioned over the first cell *(Left most cell)*.

1. If anything other than zero is in the first cell, then **REJECT**.
2. If Zero is in the cell, then change zero to X.
3. Move right, to the first 1. If there is no 1, **REJECT**.
4. Change 1 to Y.
5. - Move to the left most 0.
   - If none, move right looking for either a 0 or 1.
   - If either 0 or 1 is found before the blank symbol, then **REJECT**, otherwise, **ACCEPT**.
6. Go to step 2.

Let's process some strings.

INPUT - 0011_

[0][0][1][1][_]
^
[x][0][1][1][_]
^
[x][0][1][1][_]
------^
[x][0][y][1][_]
------^
[x][0][y][1][_]
---^
[x][x][y][1][_]
---^

[x][x][y][1][_]
---------^
[x][x][y][y][_]
---------^
[x][x][y][y][_]
^
[x][x][y][y][_]
----------^\

Outcome - **ACCEPT**: Step 5

INPUT - 001_

[0][0][1][_]
^
[x][0][1][_]
^
[x][0][1][_]
------^
[x][0][y][_]
------^
[x][0][y][_]
---^
[x][x][y][_]
---^
[x][x][y][_]
---------^\

Outcome - **REJECT**: Step 3

Input - 011_

[0][1][1][_]
^
[x][1][1][_]
^
[x][y][1][_]
---^
[x][y][1][_]
^
[x][y][1][_]
------^
[x][y][1][_]
------^\

Outcome - **REJECT**: Step 5

Input - 010_

[0][1][0][_]
^
[x][1][0][_]
^
[x][1][0][_]
---^
[x][y][0][_]
---^
[x][y][0][_]
^
[x][y][0][_]
------^\

Outcome - **REJECT**: Step 5

NOTE the following: A = {0, 1}, the input alphabet. ”_” $\notin A$ where "_" is the blank symbol. $A\prime$ = {0,1,_,x,y} is the tape alphabet. S a set of states.

NOTE also 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 specifying where the tape head goes.