Monash University
Faculty of Information Technology
2nd Semester 2023

FIT2014
Exercises 7
Turing Machines

ASSESSED PREPARATION: Question 4.
You must provide a serious attempt at this entire question at the start of your class.

1.

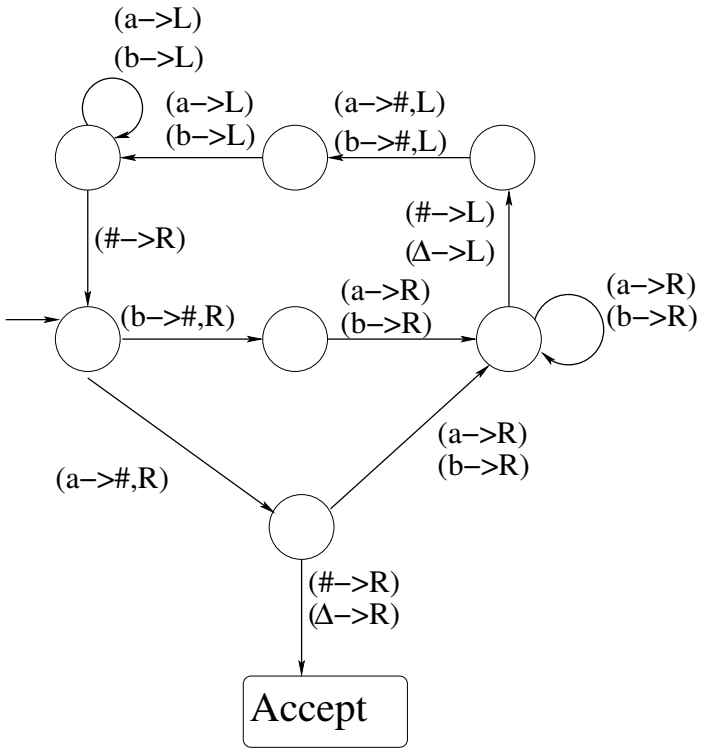
Trace the execution of the following input strings on the machine shown.

i) *aaa*

ii) *aba*

iii) *baaba*

iv) *ababb*



2. For the following inputs trace the execution of the Turing Machine built in Lectures to compute the function $f(n) = 2n$.

i) Δ

ii) a

iii) aa

3. What is the main difference between Final States in Finite Automata and Accept States in Turing machines?

4. In this question, you will build a Turing machine that interchanges the first and last letter of the input string.

Some examples:

input		output
ε	$\longmapsto$	ε
`a`	$\longmapsto$	`a`
`b`	$\longmapsto$	`b`
`ab`	$\longmapsto$	`ba`
`baa`	$\longmapsto$	`aab`
`abba`	$\longmapsto$	`abba`
`babbb`	$\longmapsto$	`babbb`

(a)

First, design it on paper, and write your Turing machine as a table or a diagram.

(b)

Then build it in Tuatara v2.1 and demonstrate it there.

Some things to consider:

- Develop your TM in small stages. Here's one approach. To begin with, try writing a TM that just replaces the last letter by `a`. Then modify it so that it replaces the last letter by the first letter. Then try to extend it to do the whole task.
- Once you've built your TM, think about what it does when it returns to the first tape cell. Does it try to move left from that first cell? Our TMs in lectures crash if that happens, but Tuatara TMs don't crash in that situation: instead, they just stay still at the first tape cell. If you Tuatara TM exploits this slackness of Tuatara, modify it so that it behaves properly.

(c) (optional)

If your Tuatara TM diagram is particularly beautiful and elegant, ask your tutor's opinion of its aesthetic and artistic qualities.

(d)

Let $t(n)$ be the maximum number of steps your TM takes, over all input strings of length n. Find an expression for $t(n)$, as a function of n.

If finding an exact expression is too hard, find the best upper bound expression that you can. (Your expression should still be a function of n, and it must be $\geq t(n)$ for all n, but as close to $t(n)$ as you can get.)

5. Build a Turing Machine that accepts the language $\{a^n b^{n+1}\}$.

6. Build a Turing Machine which computes the function, $f(n) = n/2$.

7.

Describe the classes of languages recognised by each of the following:

(i) TMs which cannot move on the tape.

(ii) TMs which only move to the right.

(iii) TMs which only stay still or move to the right (i.e., cannot move to the left).

(iv) TMs which operate like normal TMs except that, if they attempt to move Left when in the first tape cell, they stay still instead of crashing.

8. Explain how a Turing machine can be simulated by a generalised Pushdown Automaton with two stacks (2PDA).

(This is like an ordinary PDA except that a transition is specified by: the input character, the characters at the top of each of the two stacks (which are both popped), and the characters that are then pushed onto each of the two stacks. Any of these characters may be replaced by ε, with the usual meaning.)

9.

Suppose you have a Turing machine M. In this question, we will use the following propositions about M.

- For each time t, each tape position i and each symbol s, the propostion $A_{t,i,s}$ means:

  At time t, tape cell i contains symbol s.

- For each time t and each state q, the propostion $B_{t,q}$ means:

  At time t, the machine is in state q.

- For each time t and each tape position i, the propostion $C_{t,i}$ means:

  At time t the Tape Head is at tape cell i.

The time t can be any non-negative integer (with start time being $t = 0$). The tape cell i can be any positive integer. The state q can be any number in $\{1, 2, \ldots, N\}$, where N is the number of states. The symbol s is a member of $\{\mathbf{a},\mathbf{b},\Delta\}$, where as usual Δ denotes the blank symbol.

Use the above propositions to construct logical expressions with the following meanings.

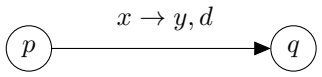
(a) At time t, tape cell i has at least one symbol in it.

(b) At time t, tape cell i has at most one symbol in it.

(c) At time t, tape cell i has exactly one symbol in it.

(d) At time t, and with Tape Head at tape cell i, **if** the current state is p and the symbol in cell i is x, **then** at time $t + 1$ the state is q, the symbol in cell i is y, and the Tape Head is at cell $i + d$, where $d \in \{\pm 1\}$.

This describes the following transition:



10.

Suppose that a particular Turing machine T always stops in $\leq n^2$ steps, where n is the length of the input string.

Let U be a Universal Turing Machine (UTM), which takes as input a pair (M, x) where M is any Turing machine and x is an input string for M.

Suppose that any computation by any M that takes t steps can be simulated by U in $\leq t^3$ steps.

Derive an upper bound for the time taken by U to simulate T on an input of length n.

Supplementary exercises

11. Build a TM that accepts the language of all words that contain the substring *bbb*.

12. Build a TM that accepts the language of all words that **do not** contain the substring *bbb*.

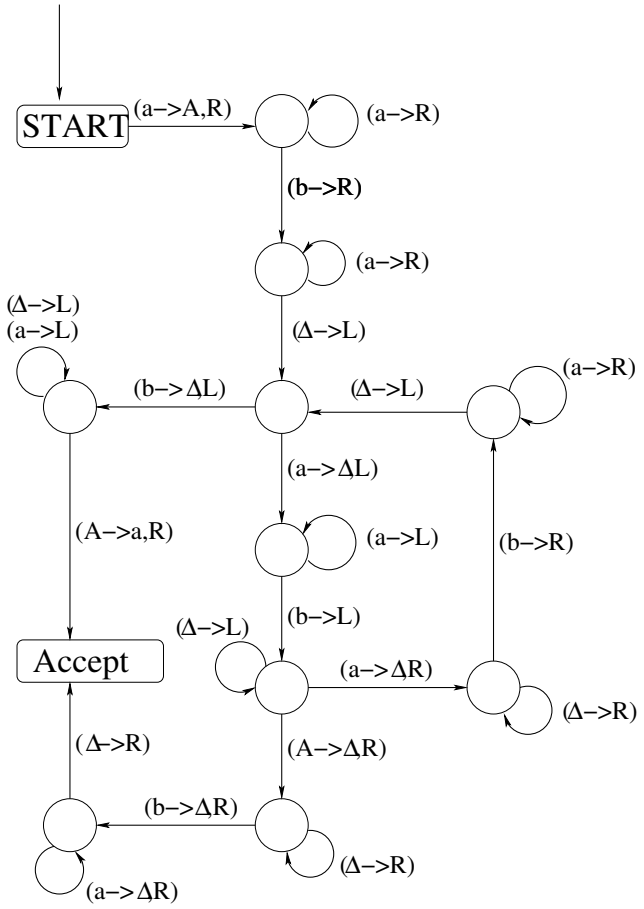
13. Build a Turing Machine that accepts all strings with more a's than b's.

14. The Turing Machine below computes something related to unary subtraction. Run the following inputs on this machine and interpret the results.

i) *aaaba*

ii) *baaa*

iii) *aabaa*



15.

Explain how a Turing machine can be simulated by a Queue Automaton.

(This is superficially like a PDA, except that instead of a stack, it has a queue. A transition $x, y \to z$ means that x is the current input character, y is at the head of the queue and is removed

from it (i.e., served), and z is then appended to the queue. Any of these characters may be replaced by ε, with the usual meaning.)

16. Build a Turing machine that takes as its input a string of the form $x\#b$ where $x \in \{0,1\}^*$ is a binary string and $b \in \{0,1\}$ is a single bit, and:

- if $b = 0$, just outputs x;
- if $b = 1$, flips every bit of x (i.e., changes every 0 to 1, and every 1 to 0).

Before stopping, your TM must erase $\#b$, since the output does not include these characters.

See also Q17.

17. Extend your TM from Q17 so that it now has two bits, $b_0 b_1$, after the `#`, and the decision of whether or not to flip a bit of x depends on the *parity* of the position of the bit on the tape: for bits of x in even-numbered tape cells, we flip or not according to b_0, while for bits of x in odd-numbered tape cells, we flip or not according to b_1. (We imagine the tape cells to be numbered $0, 1, 2, 3, \ldots$, from left to right.)

As in Q16, your TM must erase `#` and everything beyond it before stopping.

You can use states of the TM to remember whether you currently want to flip a bit or not. But remembering whether the next bit to be flipped is on an even-numbered cell or an odd-numbered cell is best done using appropriate marks on the tape, rather than choice of state.