Monash University
Faculty of Information Technology
2^{nd} Semester 2022

FIT2014
Tutorial 6
Turing Machines

ASSESSED PREPARATION: Question 6.
You must submit a serious attempt at this entire question to your tutor by the Sunday evening prior to your tutorial.

1.

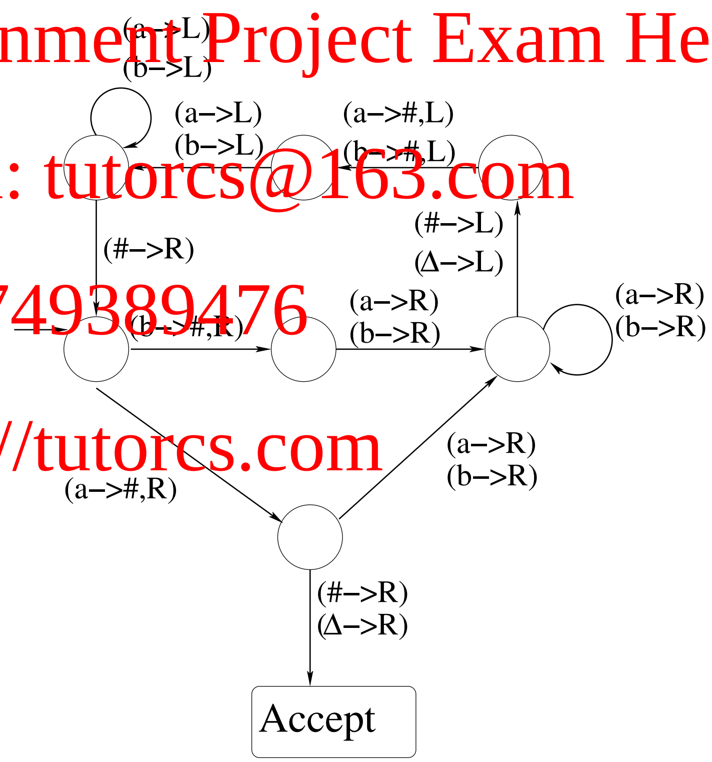
Trace the execution of the following input strings on the machine shown.

i) *aaa*

ii) *aba*

iii) *baaba*

iv) *ababb*



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

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

4. 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

5.

Describe the classe[...]ed by each of the following:
(i) TMs which can[...]
(ii) TMs which on[...]
(iii) TMs which on[...] the right (i.e., cannot move to the left).
(iv) TMs which op[...] except that, if they attempt to move Left when in the first tape cell, they st[...]ng.

6. First, read about *quaternions* in Assignment 2, including the use of quaternions to represent 3D rotations.

(a) For each of the fundamental unit quaternions i, j, k, state the *angle* and *axis* of the rotation they represent.

(b) If $v = xi + yj + zk$ is any pure quaternion, what can you say about each of iv/i, jv/j, kv/k?

(c) Prove, by induction on n, that for all n, a product of n fundamental unit quaternions represents a rotation by either 0° (in which case it does nothing) or 180°.

7. Suppose we have a sequence of fundamental unit quaternions $q_1, q_2, \ldots, q_n$, with each $q_i \in \{i, j, k\}$. The quaternion expression

$$q_n \cdots q_2 q_1 v / q_1 / q_2 / \cdots / q_n \tag{1}$$

represents the pure quaternion obtained from v by first applying the rotation represented by q_1, then applying the rotation represented by q_2, and so on, finally applying the rotation represented by q_n.

Build a Turing machine in Tuatara that takes as its input a string over the alphabet {`i`, `j`, `k`, `v`, `d`} (where `d` stands for <u>di</u>vision and is used to represent /, since alphabets in Tuatara can only contain alphanumeric characters), and

- *accepts* if the input string is of the form described above (see equation (1));
- *rejects*, by crashing, if the input string is *not* of this form.

See also Q15.

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, [illegible]d each symbol s, the propostion $A_{t,i,s}$ means:

  At time [illegible]ymbol s.

- For each time t [illegible]opostion $B_{t,q}$ means:

  At time [illegible]te q.

- For each time t [illegible]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:

$$p \xrightarrow{x \to y, d} q$$

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 Turing Machine that accepts the language of all words that contain the substring bbb.

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

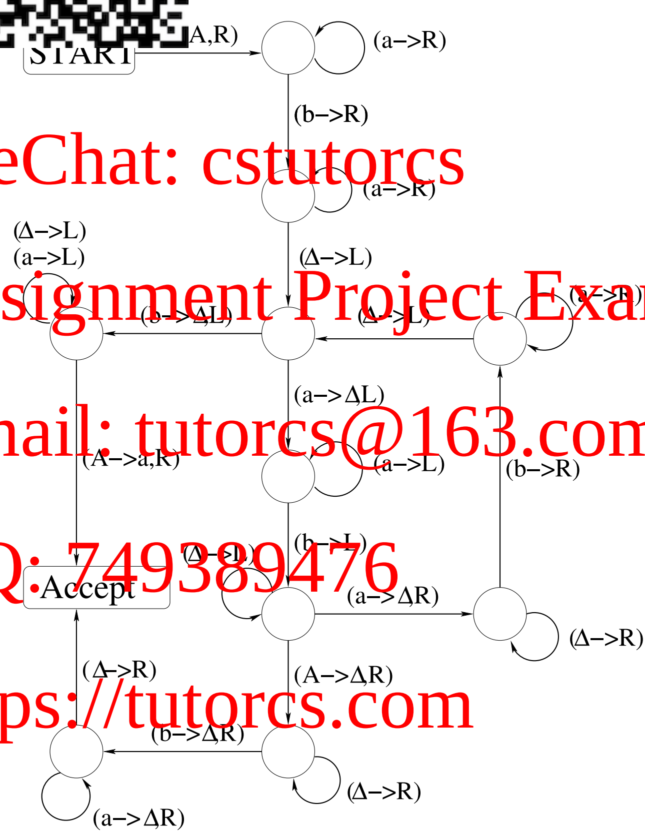
13. Build a Turing Machine that accepts all strings with more a's than b's.

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.
(a) Prove or disprove: the language of quaternion expressions over {i, j, k, v, /} described in Q7 is regular.
(b) Prove or disprove: the language of quaternion expressions over {i, j, k, v, /} described in Q7 is context-free.

16.
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.)

17. 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 out[illegible]
- if $b = 1$, flips eve[illegible]s every 0 to 1, and every 1 to 0).

Before stopping, your [illegible]e the output does not include these characters.

See also Q18.

18. 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 Q17, 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.