CITS2211 Discrete Structures
Week 12 Exercises – Turing Machines

2022

For an *implementation-level description* of a Turing Machine you do not have to give a formal specification of the machine's moves, but only to explain the algorithm in English prose in terms of the series of steps the machine would use for this calculation.

1. Give a brief explanation for the following terms in the context of Turing Machines: state, input tape, move, halt, halting problem, recognise (a language), compute (a function).

SOLUTION:

(a) State: As in a finite state machine, a value from a finite set of state identifiers that defines part of the current context of the TM.

(b) Input tape: An infinite tape on which is written a sequence of blanks and symbols from the given alphabet for the TM.

(c) Move (of the TM): Given a transition specified by (s,i,o,s',D) a TM in state s looking at input symbol i will write o to the tape, changes to state s' and move in direction D (either one space left or right).

(d) Halt: The TM reaches a state and input tape symbol for which there is no move defined

(e) Halting Problem: Is there an algorithm that given a string s_T representing a Turing machine T, and an input string α, that can determine whether T will halt if it is given the string α as input?

(f) Recognise: A TM recognises a language L if given a word $w \in L$ on its input tape, the TM will halt in an accepting state.

(g) Compute: A TM computes a function $f(x) = y$ if given an input tape with a representation of x it produces an output tape with a representation of y (and usually then halts).

2. Outline an *implementation-level description* of a Turing machine that recognises the language $(10)^*$. State any assumptions you make.

SOLUTION: Start at the leftmost input symbol on the tape. Assume we can stop at the right hand end. Or you can choose to move back to the left hand end of the string after reaching the rightmost blank (i.e. the start position). Assume that if the machine encounters an input symbol with no move then it halts and rejects the string; we do not explicitly show transitions to a reject state.

(a) If blank then accept else continue

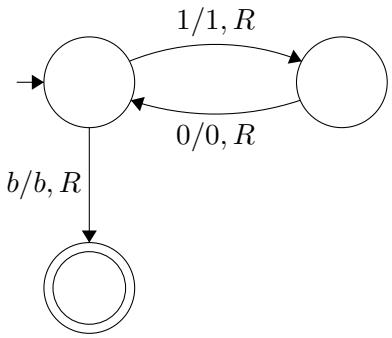
(b) Check for 1 and move R

(c) Check for 0 and move R

(d) Repeat from step (a)

3. Write a state machine version of your Turing machine to recognise the language $(10)^*$ (as in the previous question). Show all the moves of this machine.

SOLUTION:



4. Outline an *implementation-level description* of a Turing machine that computes the function $f(n) = n - 2$ where a natural number n is represented in unary. (That means 1 represents the natural number 0, and 11 represents the natural number 1, and 111 represents the natural number 2, 1111 represents 3 and so on.)

SOLUTION: Assume the number is represented in unary with 1 representing 0, 11 for 1, 111 for 2 etc.

(a) Machine starts at the left of the tape over the start symbol to the left of the first 1.

(b) Move to the right over 1s until you reach first blank (not 1)

(c) Move left, replace 1 with a blank

(d) Move left again, replace 1 with a blank

(e) Move back to left hand start symbol - the number remaining on the tape is n-2

(f) OR if either steps 3,4 failed then reject.

5. Outline an *implementation-level description* of a Turing machine that acts as a "doubler". For Input: A string of 1s of length n and for Output: A string of 1s of length $2n$. Note, this machine calculates the function $f(n) = n + n$ where a natural number n is represented in unary.

SOLUTION:

(a) Find left most 1 and change it to a 0

(b) Go right past next blank and any 1s on the tap and write 11 to the right hand end of the tape (that is write 1, go right and write another 1)

(c) Go left to the first 0, then right and return to step 1.

(d) When you run out of 1s to change to 0s, then go right to the summed 1s, blanking all 0s as you go.

(e) Delete the first 1 from the sum group (because unary n has n+1 1s, so the doubling had one too many 1 in it)

Here is another strategy (there are many more):

(a) Read leftmost 1 on the tape and change to 0.

(b) Skip to the blank at the right hand of n, skip over that, and write a 1 (ie copy the first number)

(c) Return to the leftmost 0, go right to next 1 and return to step 1.

(d) Continue this until all the original 1s are 0s.
Now you have n 0s, a blank and n 1s on the tape so need to

move up the 0s and change them back to 1s, to be left with n+n 1s. Do this as follows:

(e) Skip left over 0s until you reach a blank.

(f) Move one right and change this to a blank.

(g) Move next right and change 0 to 1, repeat until you reach the separator blank

(h) Change the separator blank to a 1.

(i) Move to the end of the string and change the last 1 to a blank, so you end up with $n + n - 1$ 1s.

6. Outline an *implementation-level description* of a Turing machine that computes the function $f(x, y) = max(x, y)$ where x and y are represented in unary number notation separated by a separator symbol $*$. The machine starts at the leftmost non-blank cell. It should leave the tape containing z in unary, where z is the maximum of x and y. Do not use any symbols other than 1, the separator $*$ and blank at any time.

SOLUTION:

Assume that one of the numbers is strictly larger than the other i.e. they are not equal. The algorithm used for the machine relies of the fact that max(x, y) = 1 + max(x - 1, y - 1). Move left and right crossing off matching 1s from the ends of the tape. Eventually there is a 1 on the left or the right, but no more 1s on the other side.

This problem is easier if you allow extra symbols. But since we only have $*$ and 1 we must first mark the edges of the input tape so that we are able to restore the 1s for the maximum number side once we know it.

The strategy is:

Part 1: mark the edges of the inputs on the tape

(a) Machine starts at the left of the tape; Change the first 1 to a $*$ to mark the left hand end.

(b) Move to the far right hand end and change the last 1 to a $*$ to mark the right hand end.

(c) Move back left over the middle $*$ until the left hand $*$.

Part 2: cross off matching 1s either side of the middle until you run out of 1s on one side

(a) Move right until the first 1, change it to a b and move left back to the middle $*$.

(b) If you find only blanks before the $*$ (no more 1s) then the *right* hand side is smaller and max is the left hand side. Change state to L, move left back to the middle and transition to part 3.

(c) Move left until the first 1, change it to a b and move left back to the middle $*$.

(d) If you find only blanks before the $*$ (no more 1s) then the *left* hand side is smaller and max is the righthand side. Change state to R, move right back to the middle and transition to part 3.

Part 3: restore all the 1s on the maximum number side

(a) If state is L then move left replacing every b or 1with 1 until you reach the leftmost $*$. Replace that with 1 and halt.

(b) If state is R then move right replacing every b or 1 with 1 until you reach the rightmost $*$. Replace that with 1 and halt.