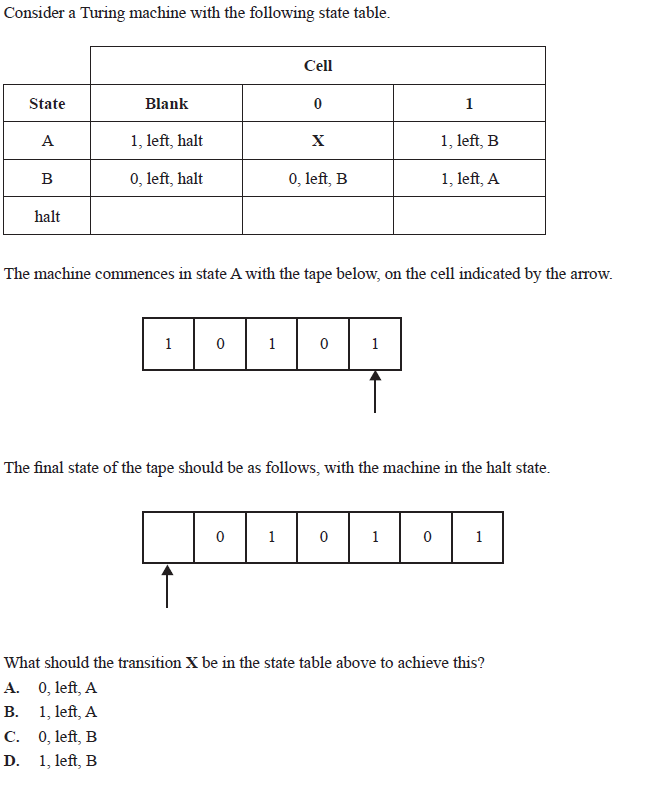
Week 9 – Submit Task

Question 1

(VCAA Algorithmics Examination 2021)

You must also explain your answer below.



Iteration 1:

* State: A
* Head: On 1 (last 1 on the right)
* Action: According to the table, write 1, move left, and transition to state B.
* Tape: 10101
* Next state: B
* Next head position: Now on the second 0 from the right.

Iteration 2:

* State: B
* Head: On 0 (second 0 from the right)
* Action: Write 0, move left, stay in state B.
* Tape: 10101
* Next state: B
* Next head position: Now on the middle 1.

Iteration 3:

* State: B
* Head: On 1 (middle 1)
* Action: Write 1, move left, transition to state A.
* Tape: 10101
* Next state: A
* Next head position: Now on the first 0 from the left.

Iteration 4 (with correct X):

* State: A
* Head: On 0 (first 0 from the left)
* Action: X=0,L,A
* Tape: 10001
* Next state: A
* Next head position: Now on the first 1.

Iteration 5:

* State: A
* Head: On 1 (first 1)
* Action: Write 1, move left, and transition to B.
* Tape: 01001
* Next state: B
* Next head position: Now on the first blank.

Iteration 6:

* State: B
* Head: On blank (leftmost)
* Action: Write 0, move left, and transition to A.
* Tape: 001001
* Next state: A
* Next head position: Still on the blank.

Each of the following questions is worth 2-3 marks. Provide a short written response.

Question 2

Explain the concept of a Turing machine and its components. How does it simulate the behaviour of a computer algorithm?

It’s a computational model made by Alan Turing in ’36 that provides a framework to understand what it means for a problem to be computable. It is a decision problem that helps determine whether a computational model can conclusively be answered with a “True” or “False”. It can read, write and erase symbols on an infinitely long tape. Its behaviour is determined by a finite state machine, which consists of a finite set of states, a transition function that defines actions to be taken based on current state and symbol being read.

Question 3

Define undecidability in the context of computer science. Provide an example of an undecidable problem and explain why it is impossible to solve algorithmically.

Undecidability refers to a class of problems for which no algorithm can be constructed that will always lead to a correct yes-or-no answer where no Turing machine can be designed that always halts with a correct solution for every possible input.

1. Assume there exists a Turing machine H that solves the halting problem.
2. Construct a new Turing machine H+ that, given an input x, runs H on x and itself.
3. If H predicts that H(x+) halts, H enters an infinite loop; if H predicts that H+(x) will run indefinitely, H halts.

This construction creates a contradiction, showing that H cannot exist because it would lead to an impossible situation where H+ both halts and does not halt.

Question 4

Define the halting problem and explain why it is considered an undecidable problem. What are the consequences of its undecidability for the limits of computation?

One of the most famous undecidable problems is the halting problem. The halting problem asks whether a given Turing machine will eventually halt (stop running) when started on a specific input, or whether it will run indefinitely. Alan Turing proved in 1936 that there is no general algorithm that can solve the halting problem for all possible Turing machine-input pairs. It shows that there are problems that no algorithm, no matter how powerful, can solve and hence there are inherent limits to algorithmic reasoning and that some problems are beyond the reach of mechanical computation

Question 5

Discuss the relationship between the halting problem and Gödel's incompleteness theorems. How do they both relate to the concept of undecidability?

Gödel's Incompleteness Theorems and the halting problem are both connected to the concept of undecidability, though they apply to different domains: Gödel's theorems apply to formal mathematical systems, while the halting problem applies to computation.

* Gödel's First Incompleteness Theorem states that in any sufficiently powerful formal system, there are statements that are true but cannot be proven within the system.
* Gödel's Second Incompleteness Theorem states that no such system can prove its own consistency.

Both theorems show the existence of limits on formal systems and computation, showing that there are true mathematical statements (or halting Turing machines) that cannot be resolved by any algorithmic process within the system.