

AQA Computer Science A-Level

4.4.5 A model of computation

Intermediate Notes



Specification:

4.4.5.1 Turing machine:

Be familiar with the structure and use of Turing machines that perform simple computations.

Know that a Turing machine can be viewed as a computer with a single fixed program, expressed using:

- A finite set of states in a state transition diagram
- A finite alphabet of symbols
- An infinite tape with marked-off squares
- A sensing read-write head that can travel along the tape, one square at a time.

One of the states is called a start state and states that have no outgoing transitions are called halting states.

Understand the equivalence between a transition function and a state transition diagram.

Be able to:

- Represent transition rules using a transition function
- Represent transition rules using a state transition diagram
- Hand-trace simple Turing machines

Be able to explain the importance of Turing machines and the Universal Turing machine to the subject of computation.



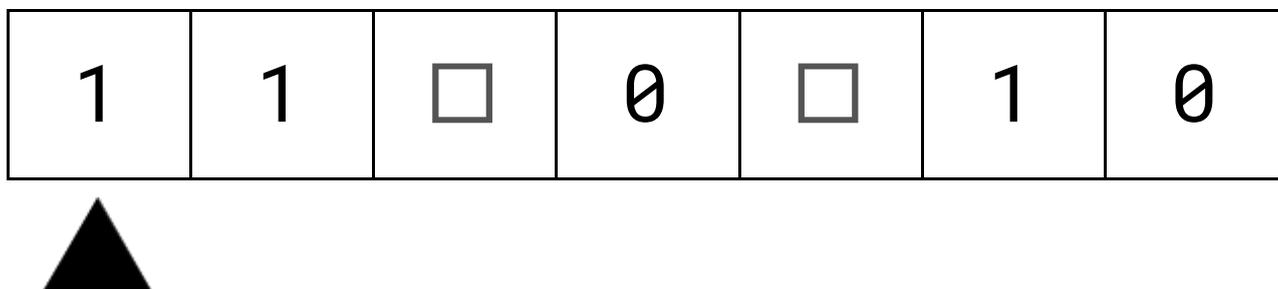
Turing Machines

A Turing Machine is a **model of computation** that consists of a **finite state machine**, a **read/write head** and a tape that is **infinitely long** in one direction.

The tape is divided into small sections called **cells**, each of which can be left **blank** or contain a **symbol**. Symbols are written to and removed from cells on the tape by the Turing machine's read/write head. The set of symbols that a Turing machine uses is called its **alphabet**.

A Turing machine can be viewed as a computer which runs only a **single program** which is represented by a **finite state machine**. The finite state machine will have a **start state** and may have a number of states called **halting states** from which the Turing machine can't change state.

Turing machines stop after reaching their halting state. This state can be entered **at any point** in the machine's execution of its input data and is entered once all of the input data has been processed.



A Turing machine can be represented graphically as a series of cells, each containing a **symbol**, and a triangular pointer which represents the position of the machine's **read/write head**. A □ symbol signifies an **empty cell**.

As a model of computation, Turing machines are **more powerful** than finite state machines because their tape is **infinitely long** in one direction.

Synoptic Link

Finite state machines are covered under **regular languages**.

It's worth reading the notes on **finite state machines** before reading these notes.



Transition Functions

The rules that a Turing machine follows can be laid out using [transition functions](#). These are written in the form:

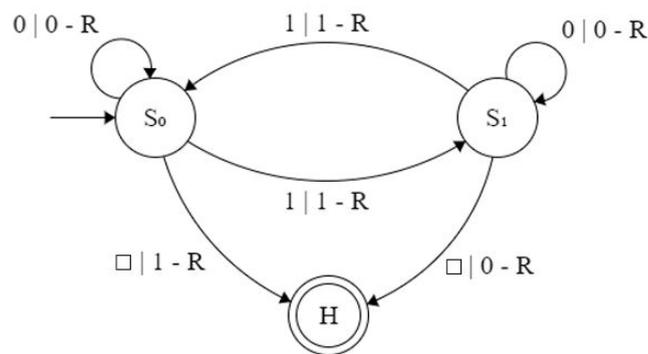
$$\delta(\text{current state, read}) = (\text{new state, write, move})$$

where δ is the Greek letter delta. For example:

$$\delta(S_0, \square) = (S_1, 1, R)$$

means if the machine is in state S_0 and reads an empty cell, the machine should write a 1, move to state S_1 , and move its read/write head to the right.

There is a link between [transition functions](#) and transition rules in a [state transition diagram](#).



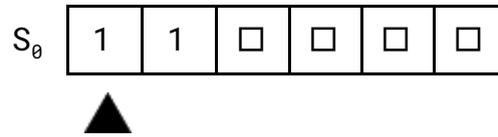
For example, the transition between state S_0 and state S_1 in the state transition diagram above could be written as a [transition function](#) like so:

$$\delta(S_0, 1) = (S_1, 1, R)$$

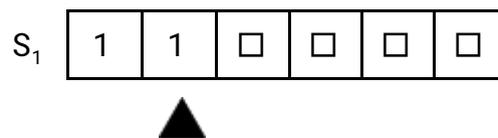


Example - Tracing a Turing machine

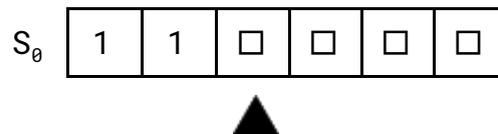
This example uses a Turing machine following the finite state machine in the state transition diagram above. Starting in S_0 (the **start state**) and with the data 110 on the tape.



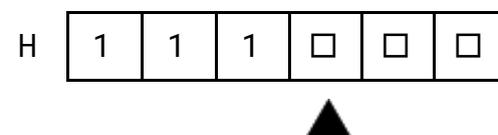
The Turing machine is in state S_0 and the read/write head reads a 1. In accordance with the state transition diagram, the Turing machine writes a 1, changes to state S_1 and moves to the right.



Now in state S_1 , the Turing machine reads a 1. As specified by the state transition diagram, the machine writes a 1 to the tape, moves to the right and changes to state S_0 .



Now the Turing machine reads an empty cell and is in state S_0 . The machine writes a 1, moves to the right and changes to state H, the **halting state**.



Universal Turing Machines



Turing machines are limited to following **just one** finite state machine, making them **specific** to the problem they need to solve.

Universal Turing machines can represent **any finite state machine**. A description of the finite state machine to be used is read off of **the same tape** as the input data and then used to process the input data as usual.

Synoptic Link

Universal Turing machines are an example of the **stored program concept**.

The stored program concept is covered in more detail under **computer organisation and architecture**.

The importance of Turing machines

Turing machines provide a **definition of what is computable** and can prove that there are problems which **cannot be solved by computers**.

