CS 162FZ: Introduction to Computer Science II

Lecture 14

Turing Machines

Dr. Chun-Yang Zhang

Introduction

- A Turing Machine is a theoretical (universal) computer.
- It is a **mathematical model** of computation that can be used to **simulate any computer algorithm**, no matter how complicated it is.
- The Turing machine was invented in 1936 by British Computer Scientist **Alan Turing.**

Alan Turing's Bio

Alan Turing, in full Alan Mathison Turing, (born June 23, 1912, London, England—died June 7, 1954, Wilmslow, Cheshire), British mathematician and logician who made major contributions to **mathematics, cryptanalysis, logic, philosophy, and mathematical biology and also to the new areas later named computer science, cognitive science, artificial intelligence, and artificial life.**

Alan Turing’s Bio

- Decrypt German cipher during the WWII, see the movie The Imitation Game

No.164 豆瓣电影Top250

模仿游戏 The Imitation Game (2014)

导演: 莫滕·泰杜姆
编剧: 格拉汉姆·摩尔 / 安德鲁·霍奇斯
主演: 本尼迪克特·康伯巴奇 / 凯拉·奈特莉 / 马修·古迪 / 罗里·金尼尔 / 艾伦·里奇 / 更多...
类型: 剧情 / 同性 / 传记 / 战争
制片国家/地区: 英国 / 美国
语言: 英语 / 德语
上映日期: 2015-07-21(中国大陆) / 2014-11-14(英国) / 2014-12-25(美国)
片长: 114分钟
又名: 解码游戏(港) / 模拟游戏
IMDb: tt2084970

豆瓣评分

8.7

504139人评价

5星	47.2%
4星	42.3%
3星	9.9%
2星	0.5%
1星	0.1%

好于 94% 传记片
好于 97% 同性片

Alan Turing’s Bio

- He committed suicide with a poison apple.

- He was condemned to homosexuality, and pardoned in 2012 by the Queen Elizabeth.

- The greatest prize in computer sciences is named by his name.

What is Turing Machine?

A Turing machine is a hypothetical machine thought of by the mathematician Alan Turing in 1936. Despite its simplicity, the machine can simulate ANY computer algorithm, no matter how complicated it is!

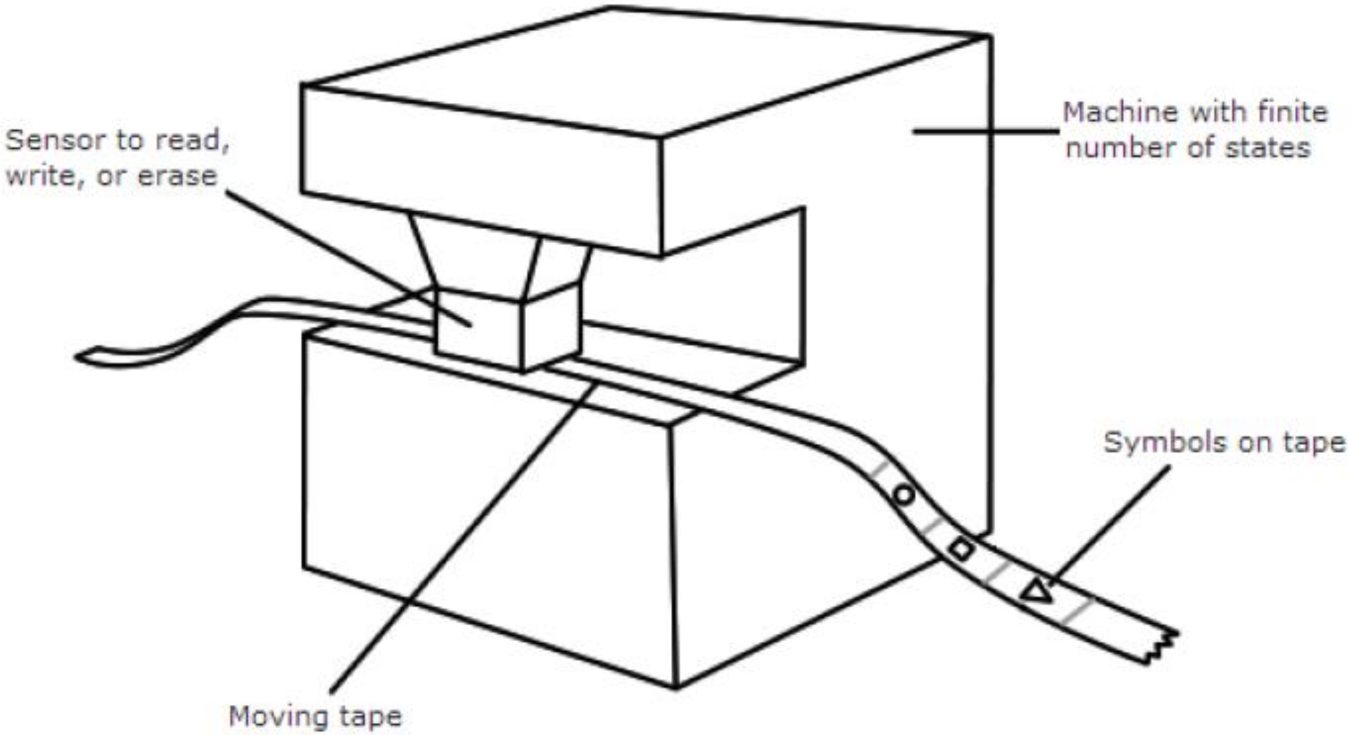
It consists of an infinitely-long tape which acts like the memory in a typical computer, or any other form of data storage. The squares on the tape are usually blank at the start and can be written with symbols. In this case, the machine can only process the symbols 0 and 1 and " " (blank), and is thus said to be a 3-symbol Turing machine.

Turing Machine

At any one time, the machine has a head which is positioned over one of the squares on the tape. With this head, the machine can perform three very basic operations:

- Read the symbol on the square under the head.
- Edit the symbol by writing a new symbol or erasing it.
- Move the tape left of right by one square so that the machine can read and edit the symbol on a neighboring square.

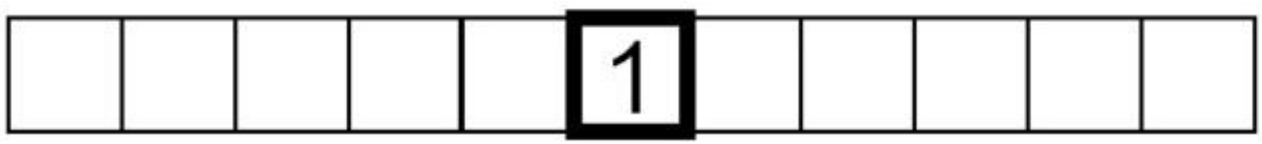
Example of Turing Machine



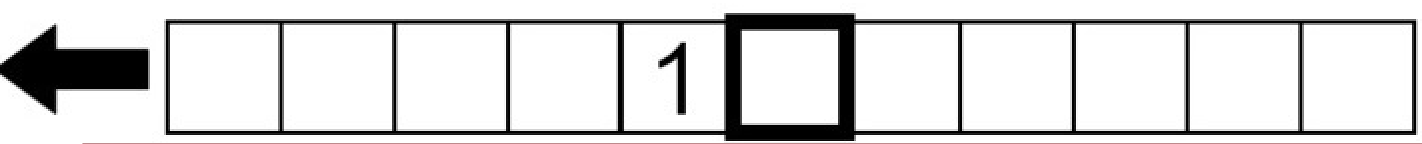
A simple demonstration

As a trivial example to demonstrate these operations, let's try printing the symbols "1 1 0" on an initially blank tape:

First, we write a 1 on the square under the head:



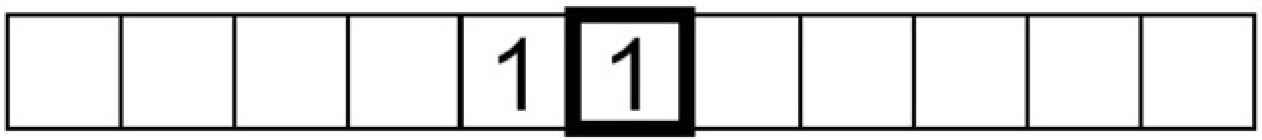
Next, we move the tape left by one square:



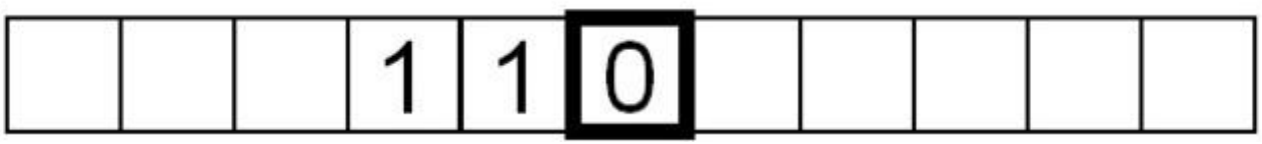
Maynooth University
National University of Ireland Maynooth

A simple demonstration

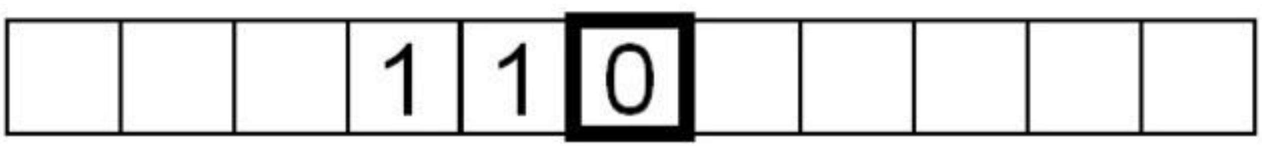
Now, write a 1 on the new square under the head:



We then move the tape left by one square again:



Finally, write a 0 and that's it!



A simple program

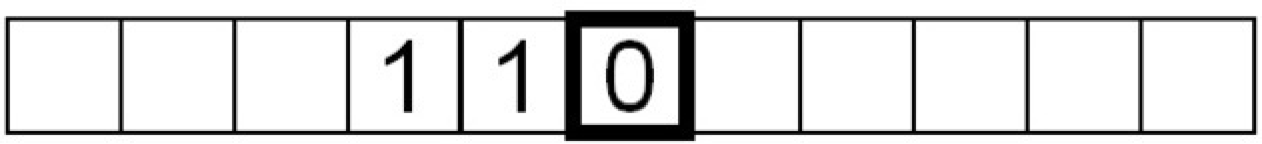
With the symbols "1 1 0" printed on the tape, let's attempt to convert the 1s to 0s and vice versa. This is called bit inversion, since 1s and 0s are bits in binary.

This can be done by passing the following instructions to the Turing machine, utilizing the machine's reading capabilities to decide its subsequent operations on its own. These instructions make up a simple program.

Symbol read	Write instruction	Move instruction
Blank	None	None
0	Write 1	Move tape to the right
1	Write 0	Move tape to the right

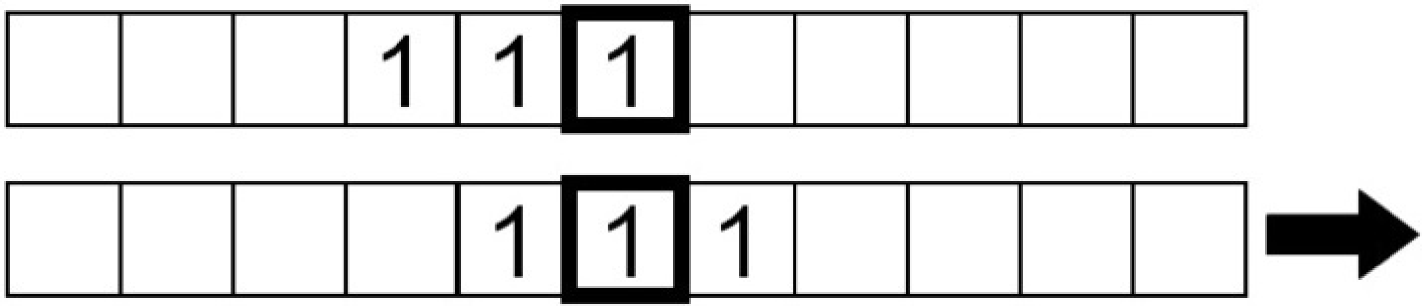
A simple program

- The machine will first read the symbol under the head, write a new symbol accordingly, then move the tape left or right as instructed, before repeating the read-write-move sequence again.
- Let's see what this program does to our tape from the previous end point of the instructions:

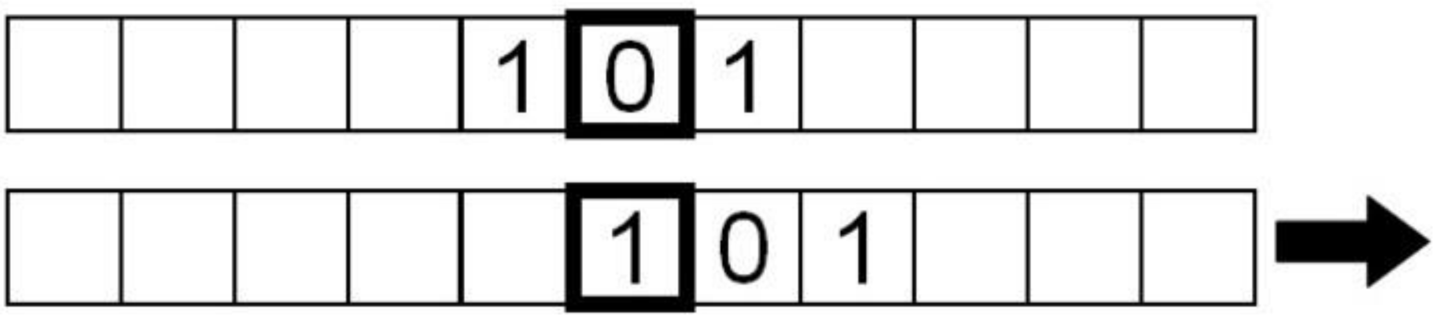


The current symbol under the head is 0, so we write a 1 and move the tape right by one square.

A simple program

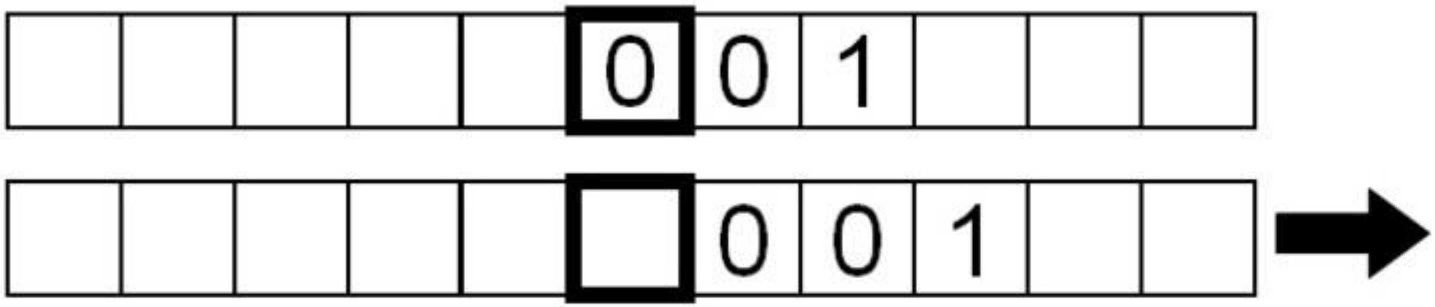


The symbol being read is now 1, so we write a 0 and move the tape right by one square:



A simple program

Similarly, the symbol read is a 1, so we repeat the same instructions.



Finally, a 'blank' symbol is read, so the machine does nothing apart from read the blank symbol continuously since we have instructed it to repeat the read-write-move sequence without stopping.

In fact, the program is incomplete. How does the machine repeat the sequence endlessly, and how does the machine stop running the program? The program tells it to with the concept of a **machine state**.

Overview of a Turing Machine

- A Turing Machine (TM) is an idealized computing device consisting of a read/write head with a paper tape passing through it.
- The tape is of **unbounded length.**
- The tape is divided into squares, each square bearing a single symbol - '0' or '1', for example.
- The tape acts as the machine's general purpose storage medium, serving both as the **means of input and output** and also as a **working memory** for storing the results of intermediate steps of the computation
- The machine needs to keep **track of the previous state** it was in when it moves to a new state.

Maynooth University
National University of Ireland Maynooth

Overview of a Turing Machine

- There must be **a finite number of symbols** used in the alphabet that the Turing machine can recognize.
- The read/write head is **programmable.**
- To compute with the device, **you program it**, write the input on the tape, place the head over the square containing the **leftmost input symbol**, and set the machine in motion.
- Once the **computation is completed**, the machine will come to a halt with the head positioned over the square containing the **leftmost symbol** of the output (or elsewhere if so programmed).

Overview of a Turing Machine

There are just **six types of fundamental operation** that a Turing machine performs in the course of a computation. These are to:

- **read the symbol** that the head is currently over
- **write a symbol** on the square the head is currently over it will need to **clear** the symbol currently here, if any
- **move the tape left one position**
- **move the tape right one position**
- **change state**
- **halt**

Overview of a Turing Machine

- A program or 'instruction table' for a Turing machine is a **finite collection of instructions**, each calling for certain operations to be performed **if certain conditions** are met.

- Every instruction is of the form:

  If the current state is *n* and the symbol under the head is *x*, then write *y* on the square under the head, go to state *m*, and move one square **left or right**

Example of Instruction Table

An example of one such table might be:

Current State	Current Symbol	Print Symbol	Move Tape	Next State
a	1	1	L	B
a	0	*	R	S
b	1	0	R	A
...				

Example of Instruction Table

- There are **three special states**: **start state, accept state** and **reject state.**
- The Turing Machine computes until it produces an output:
- It either **accepts or rejects** by entering **designated halt states.**
- If it never enters an accepting or rejecting state the Turing Machine **goes on forever, never halting.**

Current State	Current Symbol	Print Symbol	Move Tape	Next State
a	1	1	L	B
a	0	*	R	S
b	1	0	R	A
...				

Example Turing Machine

- Describe a TM *M1* that multiplies an integer number by 10. If the input on the tape is:

...	0	1	2	Δ	Δ	...

The output should be:

...	0	1	2	0	Δ	...

- The alphabet of this TM is: $\Gamma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \Delta\}$ where **Δ is the empty symbol.**

- The input alphabet (what the TM **can write** on the tape) is: $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Example Turing Machine

- The instruction table for this TM might be:

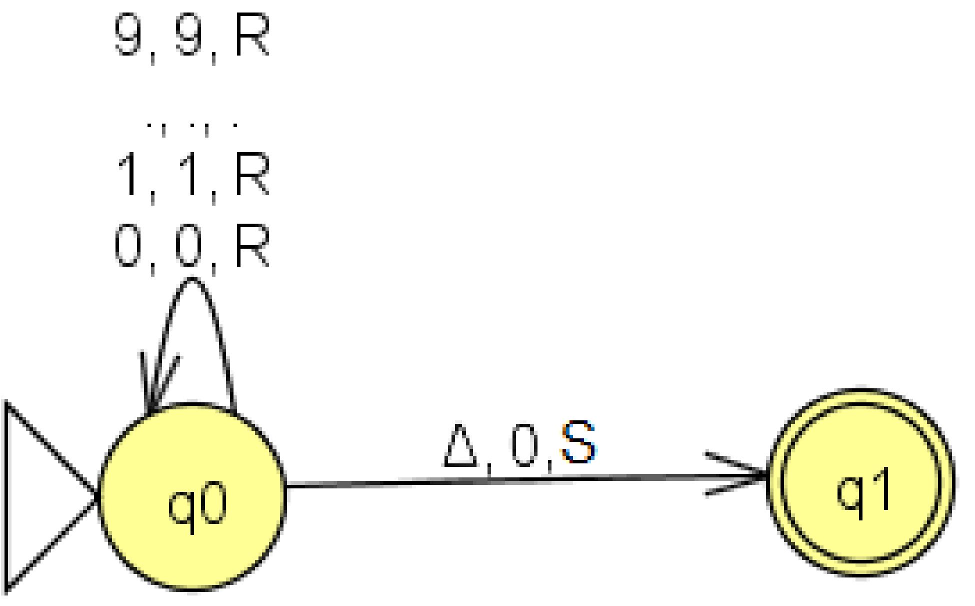
Current State	Current Symbol	Print Symbol	Move Tape	Next State
q_0	0	0	R	q_0
q_0	1	1	R	q_0
q_0	2	2	R	q_0
q_0	3	3	R	q_0
q_0	4	4	R	q_0
q_0	5	5	R	q_0
q_0	6	6	R	q_0
q_0	7	7	R	q_0
q_0	8	8	R	q_0
q_0	9	9	R	q_0
q_0	Δ	0	S	q_1

Example Turing Machine

- **q0** is the starting state for this TM.
- We will stay in this state reading symbols and moving right until we come across the first empty symbol Δ.
- Once we encounter this, we want to change this Δ symbol to a 0 to represent multiplying the number by 10
- We move to state **q1** which is the accepting state for this TM and halt (represented by the movement **S (Stop)**).

Example Turing Machine

The graphical representation of this TM is:



Example Turing Machine

- Design a TM that will perform the unary addition of two numbers. Unary representation can be defined as follows: 1 = 1, 2= 11, 3 = 111, 4 = 1111, 5 = 1111, …
- Unary represents a number x by using x 1’s – written as 1^x.
- The unary addition of 2(11) and 4 (1111) is:

11 + 1111 = 111111 (Equivalent to: $1^2 + 1^4 = 1^6$)

A sample input for the TM is:

…	1	+	1	1	Δ	…

The output should be:

…	1	1	1	Δ	Δ	…

Example Turing Machine

- What is the alphabet? What is the input alphabet? What is the instruction table? Can you design the graphical representation?
- The simplest way to solve this is to find the + symbol and change it to a 1.
- We must then move to the last 1 to the right and replace it with a Δ.
- The alphabet is: $\Gamma = \{1, +, \Delta\}$
- The input alphabet (what the TM **can write** on the tape) is: $\Sigma = \{1, \Delta\}$

Example Turing Machine

- Let us assume that we start at the first one of the first number to the left.
- Let us assume that this is state q0 – we will stay in q0 while we keep encountering 1's, writing 1's to the tape and moving right with each step.
- When we encounter the + symbol we need to first change this symbol from a + to a 1, move right to the next symbol and move to state q1.
- We move to this new state as we no longer need to worry about the first number or the + symbol.

Example Turing Machine

- We now know we are at the start of the second number.
- We will keep reading 1's, writing 1's to the tape and moving **right** (all the time staying in q1) until we encounter a Δ symbol.
- When we encounter the Δ symbol we write a Δ to the tape and move back **left** – we need to get to the last 1 to remove it from the tape.
- We will change state again to q2.
- We now know that we should encounter a 1 which needs to be overwritten with a Δ.
- This is the unary addition complete and we move to state q3 which is the **halting state.**

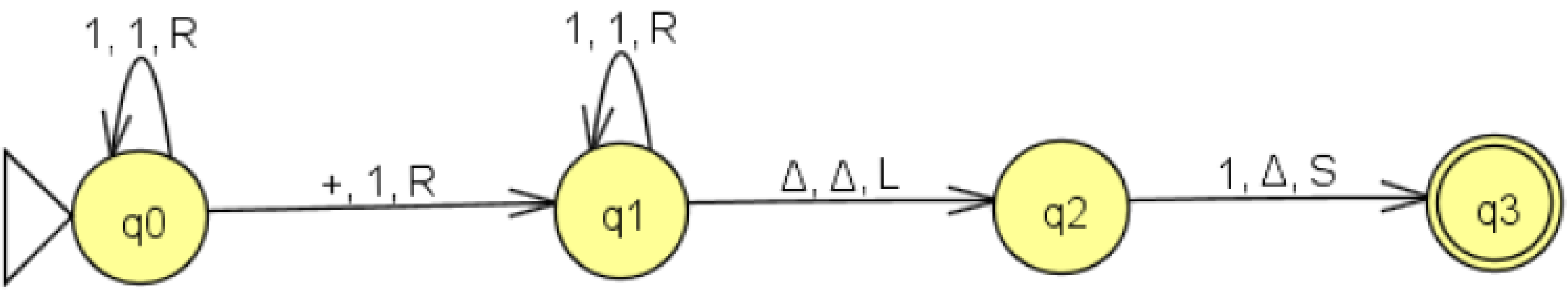
Example Turing Machine

- The instruction table for this TM might be:

Current State	Current Symbol	Print Symbol	Move Tape	Next State
q_0	1	1	R	q_0
q_0	+	1	R	q_1
q_1	1	1	R	q_1
q_1	Δ	Δ	L	q_2
q_2	1	Δ	S	q_3

Example Turing Machine

The graphical representation of this TM is:



Church-Turing Thesis

Any real-world computer can be simulated by a Turing machine.

- Proposed independently by Alonzo Church and Alan Turing.
- “Everything computable is computable by a Turing Machine”.

Neural Turing Machine

- Three components
  - Memory
  - Controller
  - Read/Write Heads

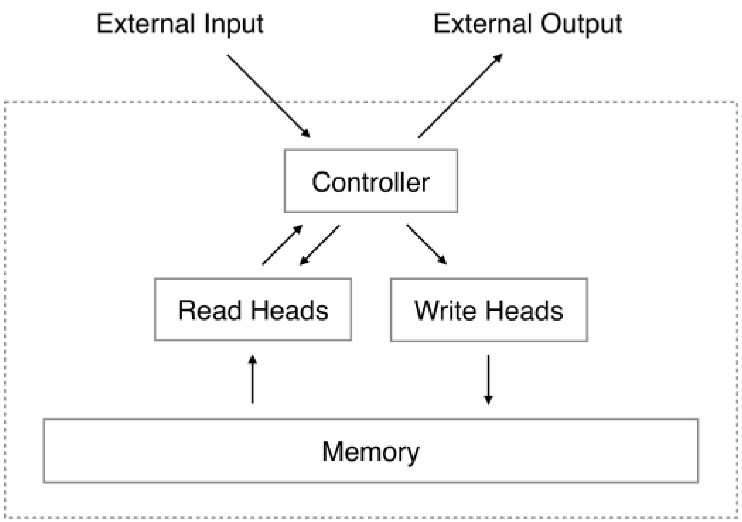


Figure 1: Neural Turing Machine Architecture. During each update cycle, the controller network receives inputs from an external environment and emits outputs in response. It also reads to and writes from a memory matrix via a set of parallel read and write heads. The dashed line indicates the division between the NTM circuit and the outside world.

Turing Machine: Summary

- A Turing Machine (TM) is a theoretical (universal) computer. It is a mathematical model of computation that can be used to simulate any computer algorithm
- There are six types of fundamental operation that a TM performs in the course of a computation.
- These are to:
  - Read
  - Write
  - Move the tape to one left position
  - Move the tape to one right position
  - Change state
  - Halt
- Every TM can be represented by an 'instruction table' which is a finite collection of operating instructions

Turing Machine: Summary

- There are three special states of a TM: start state, accept state and reject state
- The TM computes until it produces an output: it either accepts or rejects by entering designated halt states.
- If a TM never enters an accepting or rejecting state the TM goes on forever, never halting.
- Any real-world computer can be simulated by a Turing machine