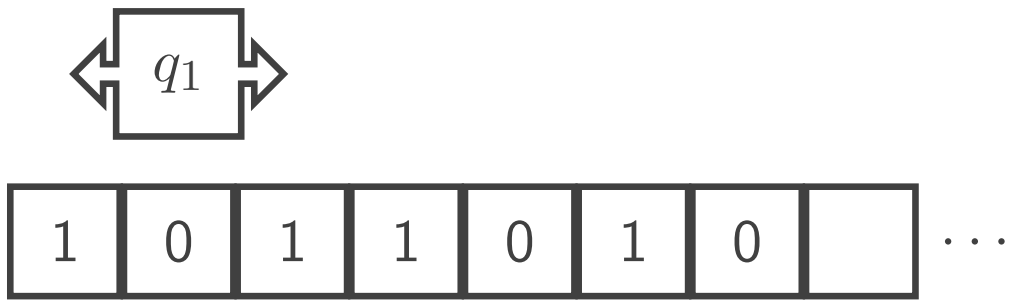
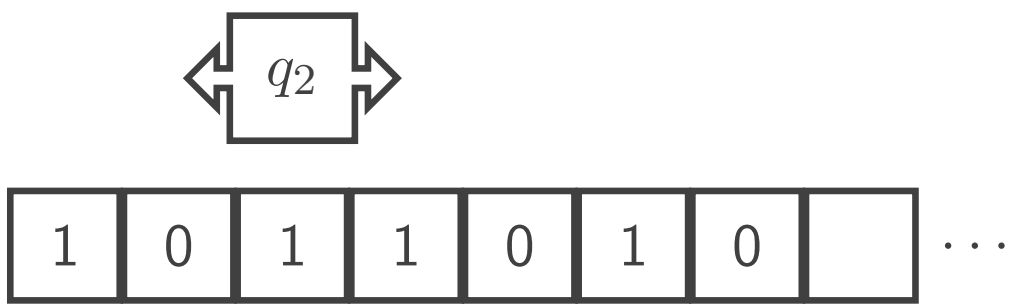
Turing Machines

ian.mcloughlin@gmit.ie

Visualisation





Turing's Second Example

Turing's own words...

As a slightly more difficult example we can construct a machine to compute the sequence $001011011101111011111\ldots$. The machine is to be capable of five m-configurations, viz. $\mathfrak{o}$, $\mathfrak{q}$, $\mathfrak{p}$, $\mathfrak{f}$, $\mathfrak{b}$ and of printing e, x, 0, 1. The first three symbols on the tape will be $ee0$; the other figures follow on alternate squares. On the intermediate squares we never print anything but x. These letters serve to keep the place for us and are erased when we have finished with them. We also arrange that in the sequence of figures on alternate squares there shall be no blanks.

www.cs.virginia.edu/ robins/Turing_Paper_1936.pdf

Turing's second example: state table

m-config.	symbol	operations	final m-config.
$\mathfrak{b}$		$Pe, R, Pe, R, P0, R, R, P0, L, L$	$\mathfrak{o}$
$\mathfrak{o}$	1	R, Px, L, L, L	$\mathfrak{o}$
	0		$\mathfrak{q}$
$\mathfrak{q}$	Any (0 or 1)	R, R	$\mathfrak{q}$
	None	$P1, L$	$\mathfrak{p}$
$\mathfrak{p}$	x	E, R	$\mathfrak{q}$
	e	R	$\mathfrak{f}$
	None	L, L	$\mathfrak{p}$
$\mathfrak{f}$	Any	R, R	$\mathfrak{f}$
	None	$P0, L, L$	$\mathfrak{o}$

Modern State Table

State	Input	Write	Move	Next
q_0	_	_	L	q_a
q_0	0	0	R	q_0
q_0	1	1	R	q_1
q_1	_	_	L	q_f
q_1	0	0	R	q_1
q_1	1	1	R	q_0

$$\delta(q_i, \gamma_n) \rightarrow (q_j, \gamma_m, L/R)$$

Running an input

Tape input: 101101

q_0	1		0		1		1		0		1		_
	1	q_1	0		1		1		0		1		_
	1		0	q_1	1		1		0		1		_
	1		0		1	q_0	1		0		1		_
	1		0		1		1	q_1	0		1		_
	1		0		1		1		0	q_1	1		_
	1		0		1		1		0		1	q_0	_
	1		0		1		1		0	q_a	1		_

Tape output: 101101

Final state: q_a (the accept state)

Outcomes

Running a Turing machine on a given tape input has two effects.

Accept or Fail

The Turing machine halts by ending in the accept or reject state. The machine accepts or rejects its input. Note a third possibility: the machine doesn't stop.

Tape content

Sometimes we're interested in what is left on the tape when the Turing machine finishes. Even for Turing machines that run forever, we can be interested in the tape output. Some of Turing's examples focused on this.

Notation

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$$

- Q Set of states (finite)
- Σ Input alphabet, subset of $\Gamma \setminus \{\_\}$
- Γ Tape alphabet (finite)
- $\_$ Blank symbol, element of Γ
- δ Transition function, $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- q_0 Start state, $\in Q$
- q_a Accept state, $\in Q$
- q_r Reject state, $\in Q$, $\neq q_a$

The blank symbol

_ is generally the only difference between the tape alphabet and the input alphabet.

Some definitions of Turing machines permit other symbols in the difference.

Empty cells of the tape in the machine are said to contain the blank symbol.

Important because the blank symbol marks the end of the input on the tape, so blank can't form part of the input.

Recap: Alphabets

Sets are collections of objects. An object is either in the set or not, and elements are distinct.

Alphabets are sets, strings are tuples over alphabets.

ϵ is the empty string.

A^* is the set of all strings over alphabet A, including ϵ.

$|w|$ denotes the length of a string w, e.g. $|001110| = 6$.

Recap: Strings

Examples

The following are examples of strings over the alphabet $\{0, 1\}$:

$$100110, 111, 0, \epsilon, 0101010, 1, 11$$

Single character strings

Note the distinction between a symbol in an alphabet and the string containing a single symbol. They look the same, but one is a symbol and one is a string. This is akin to the distinction in C between the character `'a'` and the string literal `"a"`.

Decidable languages

Language is a set of strings.

TM accepts a subset of Σ^*, the language of the TM.

Halting – a TM will accept an input, reject it, or never halt.

Decider – a TM that halts on all inputs *decides* its language.

Decidable language – some TM decides it.

Counting steps

n the length of the input, irrespective of the alphabet.

How many times do we look-up the state table (steps) before we accept a string that is accepted?

Assume the number of steps is always finite — machine is a decider.

Set $f(n)$ to the highest number of steps for strings of length n.

What does $f(n)$ look like? A polynomial, like n^2?

Non-deterministic Turing machine

Deterministic Turing machines have exactly one row in their state table for every combination of (non-terminal) state and tape symbol.

Non-deterministic Turing machines can have any number of rows for each state/symbol (including none).

Key difference: deterministic machines follow one computational path only, whereas non-deterministic machines can branch/fork into many computational paths.

Note: all deterministic machines are essentially non-deterministic machines.

Non-deterministic Turing machine and languages

Strings are accepted by non-deterministic Turing machines if any branch ends in the accept state.

Any language that is accepted (or decided) by a non-deterministic Turing machine has some deterministic Turing machine that accepts (or decides) it.

However, the non-deterministic machine may accept strings in less steps.