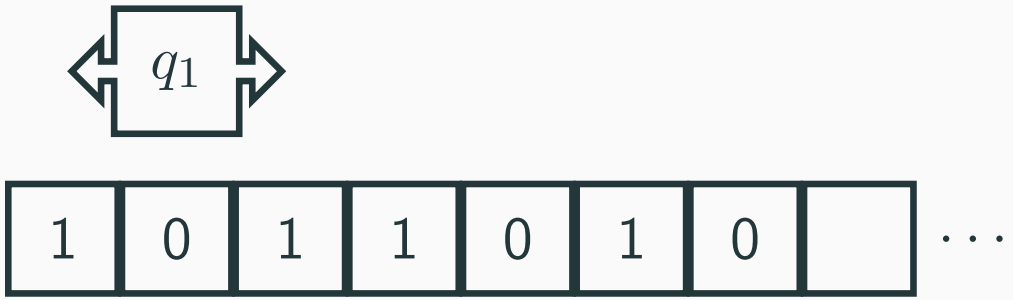
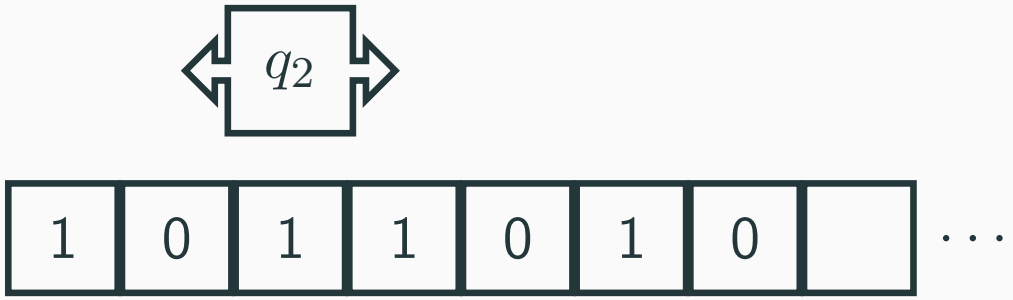
Turing machines

ian.mcloughlin@gmit.ie

Visualisation





State Table

State	Input	Write	Move	Next
q_0	$\sqcup$	$\sqcup$	L	q_a
q_0	0	0	R	q_0
q_0	1	1	R	q_1
q_1	$\sqcup$	$\sqcup$	L	q_f
q_1	0	0	R	q_1
q_1	1	1	R	q_0

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

Notation

M Turing Machine: $(Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$

Q Set of states (finite)

Σ Input alphabet, subset of $\Gamma \setminus \{\sqcup\}$

Γ Tape alphabet (finite)

$\sqcup$ 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, $\in Q$, $\neq q_a$

Blank symbol

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

Definitions of Turing machines generally don't disallow other symbols in the difference, but there's always at least a blank.

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

Importantly the blank symbol marks the end of the input on the tape.

That's why the input cannot contain the blank symbol.

Sets and alphabets

Recall sets are just collections of objects called elements.

Sets have two important attributes – an object is either in the set or not, and all elements are distinct.

Alphabets in the definition of Turing machines are just sets, and their elements are called symbols.

Strings are finite sequences (i.e. ordered lists) of symbols over alphabets.

ϵ is the empty string.

$\mathbb{A}^*$ is the set containing all strings over the alphabet A, including the empty string.

$|w|$ denotes the length of a string w, e.g. if $w = xxyzxy$ then $|w| = 6$.

Alphabets and strings

Examples

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

- 100110
- 111
- 0
- ϵ

Single character strings

Note the distinction between a symbol in an alphabet and the string containing a single string. 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".

Languages

Language is a set of strings.

Turing machines accept some strings as inputs.

Accepted language of a Turing machine is the set of strings it accepts.

Halting – given a string, a Turing machine will either accept it, reject it, or never stop (fail to halt).

Decide – a Turing machine that halts on all inputs is called a decider for the language it accepts.

Turing-decidable – a language that some Turing machine decides.

Turing's Second Example

As a slightly more difficult example we can construct a machine to compute the sequence $0010110111011111011111\ldots$. *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: 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	$P0, L, L$	$\mathfrak{o}$

Turing's Second Example: Initial state and tape

```
// The contents of the tape.
var tape = [];
// The current position of the machine on the tape.
var pos = 0;
// The current state.
var state = b;
```

Turing's Second Example: Reading and writing

```
// The blank symbol.
var blank = undefined;

// Write symbol sym to the current cell on the tape.
function write(sym) {
        tape[pos] = sym;
}

// Return true iff the symbol in the current cell is sym.
function read(sym) {
        return sym == tape[pos] ? true : false;
}
```

Turing's Second Example: Moving the machine

```
// Move the machine head right.
function right() {
        pos++;
}

// Move the machine head left.
function left() {
        pos--;
}
```

Turing's Second Example: State b

m-config.	*symbol*	*operations*	*final m-config.*
$\mathfrak{b}$		$Pe, R, Pe, R, P0, R, R, P0, L, L$	$\mathfrak{o}$

```
function b() {
        write('e'); right();
        write('e'); right();
        write('0'); right(); right();
        write('0'); left(); left();
        state = o;
}
```

Turing's Second Example: State q

m-config.	*symbol*	*operations*	*final m-config.*
$\mathfrak{q}$	Any (0 or 1)	R, R	$\mathfrak{q}$
	None	$P1, L$	$\mathfrak{p}$

```
function q() {
        if (read('0') || read('1')) {
                right(); right(); state = q;
        }
        else if (read(blank)) {
                write('1'); left(); state = p;
        }
}
```