1 Universal Turing Machines

Here is an encoding to represent an arbitrary Turing machine over an arbitrary alphabet as a string over a fixed alphabet.

state	q followed by an i digit binary number
symbol in Σ	a followed by a j digit binary number
$\sqcup$	$a0^j$
$\triangleright$	$a0^{j-1}1$
$\leftarrow$	$a0^{j-2}10$
$\rightarrow$	$a0^{j-2}11$
start state	$q0^i$

Let *encode*(M) denote the encoding of a Turing machine M. The book uses "M" for this. It consists in a sequence of (q, a, p, b) strings, where

$$\delta(q, a) = (p, b),$$

thus representing the transition table. The 4-tuples (q, a, p, b) are represented by encoding $q, a, p,$ and b as indicated above, and including left and right parentheses and commas. So a possible encoding of a 4-tuple would be

$$(q00, a100, q01, a000).$$

Here

- i would be 2,
- j would be 3,
- $q00$ would be the start state,
- $a100$ would represent a symbol,
- $q01$ would be another state, and
- $a000$ would represent $\sqcup$, that is, blank.

This encoding uses the symbols "(", ")", "q", "a", "0", "1", ",", and blank, so it uses a fixed number of symbols to encode a Turing machine having an arbitrary number of symbols in its alphabet.

1.1 Encoding of an example Turing machine

Here's an example Turing machine from section 4.1, with the names of states changed, and its encoding:

$M = (K, \Sigma, \delta, s, \{h\})$, $K = \{s, q, h\}$,$\Sigma = \{a, \sqcup, \triangleright\}$.

	q'	σ	$\delta(q', \sigma)$	4-tuple
	s	a	$(q, \sqcup)$	$(s, a, q, \sqcup)$
	s	$\sqcup$	$(h, \sqcup)$	$(s, \sqcup, h, \sqcup)$
δ:	s	$\triangleright$	$(s, \rightarrow)$	$(s, \triangleright, s, \rightarrow)$
	q	a	(s, a)	(q, a, s, a)
	q	$\sqcup$	$(s, \rightarrow)$	$(q, \sqcup, s, \rightarrow)$
	q	$\triangleright$	$(q, \rightarrow)$	$(q, \triangleright, q, \rightarrow)$

Here is the encoding of states and symbols:

States

s	$q00$	(start state is 00)
q	$q01$	
h	$q10$	(book has $q11$ here)

Symbols

$\sqcup$	$a000$	0 for blank
$\triangleright$	$a001$	1 for end marker
$\leftarrow$	$a010$	2 for left arrow
$\rightarrow$	$a011$	3 for right arrow
a	$a100$	another symbol

Then the Turing machine as a whole is encoded by concatenating the encoding of the 4-tuples in lexicographic order, according to their encodings, so

- the states are in the order $q00$, $q01$, $q10$ and
- the symbols are in the order $a000$, $a001$, $a010$, $a011$, $a100$.

Thus the 4-tuples are in the order

$$(s, \sqcup, h, \sqcup),\ (s, \triangleright, s, \rightarrow),\ (s, a, q, \sqcup),\ (q, \sqcup, s, \rightarrow),\ (q, \triangleright, q, \rightarrow),\ (q, a, s, a).$$

This gives the encoding

$$(q00, a000, q10, a000), (q00, a001, q00, a011), (q00, a100, q01, a000), \ldots, (q01, a100, q00, a100).$$

(The book gives a different order.) From this encoding all the components of the Turing machine can be obtained.

- The states are the items that appear in the first and third components,
- the halting states are the states that do not appear in the first components,
- the start state, left end marker, et cetera are given by the encoding,
- the input alphabet consists of the items appearing in the second component, and
- the transition function is given by the 4-tuples.

The encoding of a Turing machine M is denoted by $encode(M)$.

1.2 Encoding of strings

Strings are encoded by concatenating the encodings of their symbols. Thus the string

$$\rhd \sqcup aa$$

would be represented by

$$a001a000a100a100.$$

The encoding of a string x is denoted by $encode(x)$.

1.3 Encoding inputs to a universal Turing machine

The input to a universal Turing machine U would be the concatenation of $encode(M)$ and $encode(x)$ for some M and x, which would be written as

$$encode(M)encode(x).$$

Given such an encoding, U would halt if and only if M halts on input x.

1.4 Encoding in Binary

Of course, the final encoding of a Turing machine can be converted to a binary string by converting each symbol to a binary sequence. This binary string can be seen as a binary integer, so each Turing machine can be represented by a nonnegative integer.