Turing Machines

Sections 17.6 – 17.7

The Universal Turing Machine

Universal Turing Machine:

A programmable TM that accepts as input:

program   input string

It executes the program, and produces the output:

output string

The Universal Turing Machine

To formally define the Universal Turing Machine *U* we need to:

1. Define an encoding operation for TMs.

2. Describe the operation of *U* given input <*M*, *w*>, the encoding of:

    - a TM *M*, and

    - an input string *w*.

Encoding a Turing Machine *M*

We need to describe $M = (K, \Sigma, \Gamma, \delta, s, H)$ as a string:

- The states
- The tape alphabet
- The transitions

Encoding the States

- Let i be $\lceil \log_2(|K|) \rceil$.

- Number the states from 0 to $|K|$-1 in binary:
  - Number s, the start state, 0.
  - Number the others in any order.

- If t' is the binary number assigned to state t, then:
  - If t is the halting state y, assign it the string yt'.
  - If t is the halting state n, assign it the string nt'.
  - If t is any other state, assign it the string qt'.

Example of Encoding the States

Suppose *M* has 9 states.

i = 4

s = `q0000`,

Remaining states (where *y* is 3 and *n* is 4):

`q0001`, `q0010`, `y0011`, `n0100`,
`q0101`, `q0110`, `q0111`, `q1000`

Encoding a Turing Machine *M* (cont'd)

The tape alphabet:

$ax \quad : x \in \{0, 1\}^+$,
$|x| = j$, and
j is the smallest integer such that $2^j \geq |\Gamma|$.

Example: $\Sigma = \{❑, a, b, c\}$. $\quad j = 2$.

❑ = a00
a = a01
b = a10
c = a11

Encoding a Turing Machine *M* (cont'd)

The transitions: (state, input, state, output, move)

Example: (`q000`,`a000`,`q110`,`a000`,→)

An Encoding Example

Consider $M = (\{s, q, h\}, \{a, b, c\}, \{❑, a, b, c\}, \delta, s, \{h\})$:

state	symbol	δ
s	❑	$(q,❑, \rightarrow)$
s	a	$(s,b,\rightarrow)$
s	b	$(q,a, \leftarrow)$
s	c	$(q,b, \leftarrow)$
q	❑	$(s,a, \rightarrow)$
q	a	$(q,b,\rightarrow)$
q	b	$(q,b, \leftarrow)$
q	c	$(h,a, \leftarrow)$

state/symbol	representation
s	q00
q	q01
h	h10
❑	a00
a	a01
b	a10
c	a11

<*M*> = (q00,a00,q01,a00,→), (q00,a01,q00,a10,→),
(q00,a10,q01,a01,←), (q00,a11,q01,a10,←),
(q01,a00,q00,a01,→), (q01,a01,q01,a10,→),
(q01,a10,q01,a11,←), (q01,a11,h11,a01,←)

Enumerating Turing Machines

Theorem: There exists an infinite lexicographic enumeration of:

(a) All syntactically valid TMs.

(b) All syntactically valid TMs with specific input alphabet Σ.

(c) All syntactically valid TMs with specific input alphabet Σ and specific tape alphabet Γ.

Enumerating Turing Machines

Proof: Fix Σ = {(,), a, q, y, n, 0, 1, comma, $\rightarrow$, $\leftarrow$}, ordered as listed. Then:

1. Lexicographically enumerate the strings in Σ^*.
2. As each string *s* is generated, check to see whether it is a syntactically valid Turing machine description. If it is, output it.

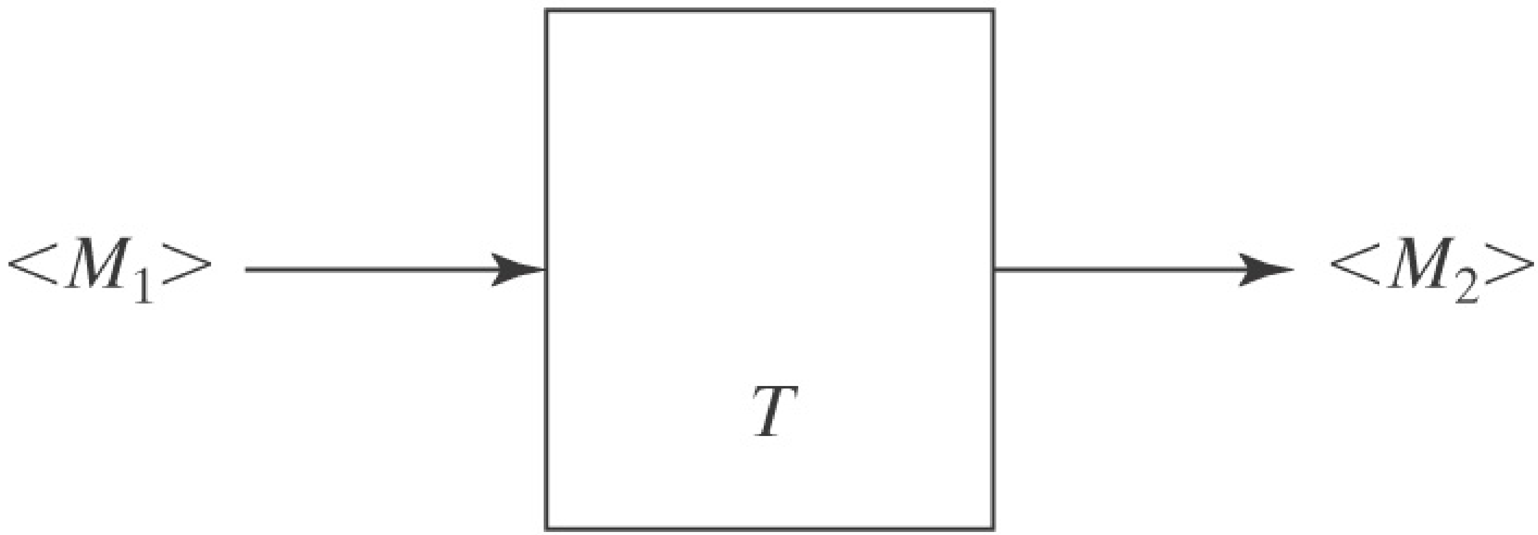
To restrict the enumeration to symbols in sets Σ and Γ, check, in step 2, that only alphabets of the appropriate sizes are allowed.

We can now talk about the *i*th Turing machine.

Another Win of Encoding

One big win of defining a way to encode any Turing machine *M*:

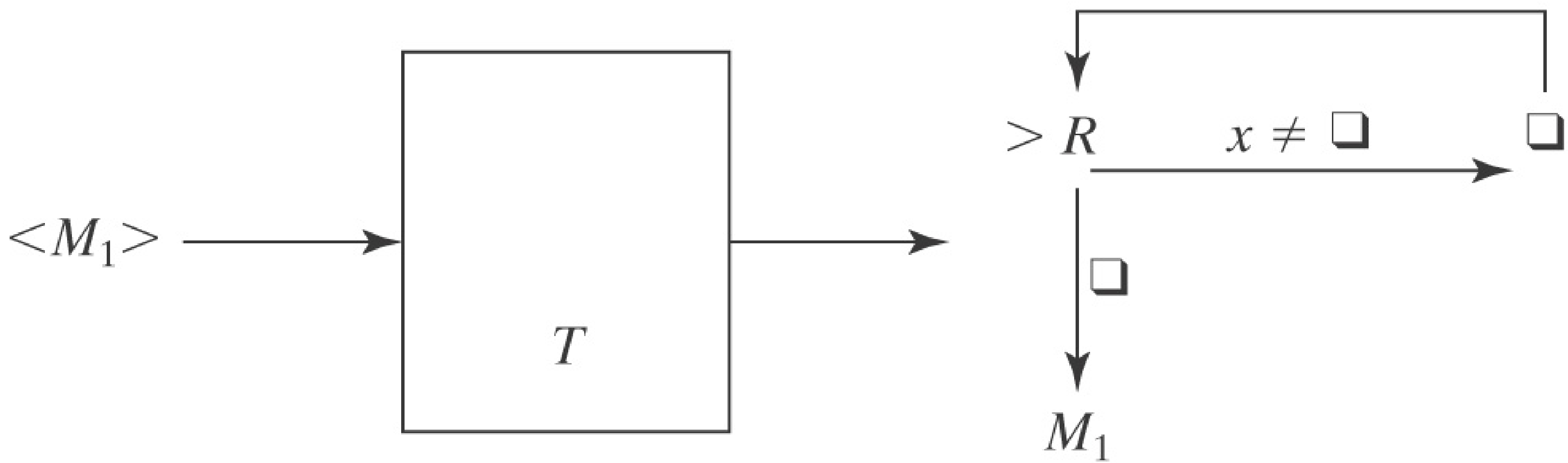
- We can talk about operations on programs (TMs).



Example of a Transforming TM *T*:

Input: a TM M_1 that reads its input tape and performs some operation P on it.

Output: a TM M_2 that performs P on an empty input tape.



Encoding Multiple Inputs

Let:

$<x_1, x_2, \ldots x_n>$

mean a single string that encodes the sequence of individual values:

$x_1, x_2, \ldots x_n.$

The Specification of the Universal TM

On input <*M*, *w*>, *U* must:

- Halt iff *M* halts on *w*.

- If *M* is a deciding or semidedciding machine, then:
    - If *M* accepts, accept.
    - If *M* rejects, reject.

- If *M* computes a function, then *U*(<*M*, *w*>) must equal *M*(*w*).

How *U* Works

U will use 3 tapes:

- Tape 1: *M*'s tape.
- Tape 2: <*M*>, the "program" that *U* is running.
- Tape 3: *M*'s state.

The Universal TM

<table>
<tr><td rowspan="6">❑</td><td colspan="4"><M--------------------------------M,</td><td colspan="3">w-----------------------w></td><td rowspan="6">❑</td><td rowspan="6">❑</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>❑</td><td>❑</td><td>❑</td><td>❑</td><td>❑</td><td>❑</td><td>❑</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>❑</td><td>❑</td><td>❑</td><td>❑</td><td>❑</td><td>❑</td><td>❑</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
</table>

Initialization of *U*:

1. Copy <*M*> onto tape 2.
2. Look at <*M*>, figure out what *i* is, and write the encoding of state *s* on tape 3.

After initialization:

<table>
<tr><td rowspan="6">❑</td><td>❑</td><td>❑</td><td>❑</td><td>❑</td><td colspan="3"><w-----------------------w></td><td rowspan="6">❑</td><td rowspan="6">❑</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td colspan="4"><M--------------------------------M></td><td>❑</td><td>❑</td><td>❑</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>q</td><td>0</td><td>0</td><td>0</td><td>❑</td><td>❑</td><td>❑</td></tr>
<tr><td>1</td><td>❑</td><td>❑</td><td>❑</td><td>❑</td><td>❑</td><td>❑</td></tr>
</table>

The Operation of *U*

	❑	❑	❑	❑	<*w*---	------------	----*w*>		
	0	0	0	0	1	0	0		
❑	<*M*---	------------	------------	----*M*>	❑	❑	❑	❑	❑
	1	0	0	0	0	0	0		
	q	0	0	0	❑	❑	❑		
	1	❑	❑	❑	❑	❑	❑		

Simulate the steps of *M* :

1. Until *M* would halt do:
    1.1 Scan tape 2 for a quintuple that matches the (current state, input) pair.
    1.2 Perform the associated action, by changing tapes 1 and 3. If necessary, extend the tape.
    1.3 If no matching quintuple found, halt. Else loop.
2. Report the same result *M* would report.