

19 April 2021

Universal Turing Machine (... this time I mean it!)

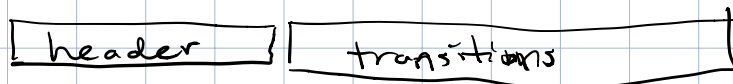
Announcement

No class this coming Friday and Monday.
Updated lecture schedule posted on website.

Turing Machine Descriptions

A description of a TM is a string of 0's and 1's that expresses the TM in some standardized format, that is simple enough to be interpreted by another TM.

For concreteness, standardize on:



where header = $0^n 1 0^m 1 0^k 1 0^s 1 0^t 1 0^r 1 0^u 1 0^v$

n = # states

m = # working tape symbols

k = # input tape symbols (first k out of m)

s, t, r = start, accept, reject states (in range $1 \dots n$)

u, v = blank and left endmarker symbols ($1 \dots m$)

The transition rule δ is represented as a sequence of strings (arbitrary order) in the "transitions" block of code.

Each of them is

$$0^p 1 0^a 1 0^b 1 0^d 1$$

If the machine, when reading symbol a in state p , writes b , transitions to q , moves in direction d where

$$d = \begin{cases} 2 & \text{for direction } -1 \\ 1 & \text{" " " } 0 \\ 3 & \text{" " " } +1 \end{cases}$$

A universal Turing machine is one that takes an input string $x \# y$ (where x, y are both written in 0's + 1's) and...

- if x is not the description of a TM, it rejects input $x \# y$.
- if x is a description of TM M and y is not a valid input description, it rejects $x \# y$.
- if $x \# y$ represents a TM M and input string z then the UTM simulates M processing input z and accepts/rejects iff M accepts/rejects z .

For a Turing machine with input alphabet of size k , an input description is a binary string $0^{a_1}10^{a_2}10^{a_3}1\dots0^{a_\ell}1$ representing the input string a_1a_2,\dots,a_ℓ where $1 \leq a_1, \dots, a_\ell \leq k$.

We can describe the contents of M 's working tape in binary similarly by using run-length encoding with runs of 0's of length ranging $1, \dots, m$. ($m = \text{size of working alphabet}$.)

Def. The configuration of a TM is (p, j, z) where

element of Q $\rightarrow p =$ current state
natural number $\rightarrow j =$ current read/write head position
finite string over Γ . $\rightarrow z =$ working tape contents up to and including last non-blank symbol. (not including left end marker)

Configurations can be run-length encoded as

$$0^p10^j10^{z_1}10^{z_2}1\dots10^{z_\ell}1$$

where $\ell = \text{length of } z$.

Skeleton for UTM: Multi-tape with

Tape 1 = input tape

Tape 2 = simulated working tape

Tape 3 = state tape

UTM:

// copy x from input tape to Tape 2

while (not reading # on Tape 1)

σ = symbol on Tape 1

write σ on Tape 2

move right on Tapes 1, 2

endwhile // now Tape 2 contains x

if IsValidTM(Tape 2) returns false:

Enter reject state, r

// clear Tape 2

while (not reading \perp on Tape 2)

write blank symbol on Tape 2

move left

endwhile

while (not reading \perp on Tape 1)

σ = symbol on Tape 1

write σ on Tape 2

move right on Tapes 1, 2

endwhile // now y is on Tape 2
if **ISVALID INPUT** (Tape 1, Tape 2) returns false:
 Enter reject state, r
// $x \# y$ is a valid TM and its input
// Now we want to actually simulate
// x running on input y

INITIAL CONFIG (Tape 1, Tape 2)

// Takes $x \# y$ on Tape 1.
// Writes initial configuration of
// x processing y on Tape 2.

// $0^s 1 0 1 y$
 start state s location 1 symbols on tape

repeat forever:

SINGLESTEP (Tape 1, Tape 2)

// simulates one transition of M
// description of M resides on Tape 1
// Tape 2 holds configuration of M
// function overwrites Tape 2
// with the config after one
// transition.

if **TEST ACCEPT** (Tape 1, Tape 2):
 Enter **accept** state, t .

IF **TEST REJECT** (Tape 1, Tape 2):

Enter **reject** state, r .

← these break the loop and halt.

How to implement SINGLE STEP?

Copy the state of M
(string $0^j 1$ at start of Tape 2.)
onto Tape 3. This will constitute
the "memory" that stores what state
 M is in.

Write 0^j on Tape 3.

Seek through j 1's on Tape 2,
deleting 0 from Tape 3
each time, until Tape 2
read/write head is in
correct position.

Seek through list of transition rules
on Tape 1 to find one
that applies to current state (Tape 3)
and symbol being read (Tape 2).