Emulating a Universal Turing machine with DSF

Damian Fascio

This universal Turing machine simulates the function of a specific Turing machine by reading its transition table and input data from the tape and performing its task. This task consists of exchanging the input values (0 and 1) and then performing the exchange again leaving the values in their original state.

The initial tape with its symbols is represented by the input elements and the read and write head is represented by the output element at different step of a sequence.

Alphabet (elements used):

ep (numerical value 0, represents a cell with the binary number 0).

epX (numerical value 1, represents a cell with the binary number 1).

e (nunerical value 0, represents a cell with the symbol _).

Working methodology:

The elements representing the symbols written on the tape are entered into the database one by one, starting with the leftmost element, and then following the moves stipulated in the transition table.

The database contains a **epX** element as a preloaded element, to allow copying its specific information to the **ep** element (transforming 0 into 1).

The main functions of this universal Turing machine is carried out through the post-operation functions (**erase** and **write** in this case).

Encoding the transition table on the tape:

All rules that this universal Turing machine reads are input via text embedded within $\mathbf{epX_n}$ elements.

Transition table:

State ***q0*** --> read _ --> write _ --> move to the right --> changes to state ***q1***

State ***q1*** --> read 0 --> write 1 --> move to the right --> continues in the state ***q1***

State ***q1*** --> read 1 --> write 0 --> move to the right --> continues in the state ***q1***

State ***q1*** --> read _ --> write _ --> move to the left --> changes to state ***q2***

State ***q2*** --> read 0 --> write 1 --> move to the left --> continues in the state ***q2***

State ***q2*** --> read 1 --> write 0 --> move to the left --> continues in the state ***q2***

State ***q2*** --> read _ --> write _ --> move to the right --> changes to state ***q3*** and stop execution.

Input data:

_ | 0 | 1 | _ (represented as **e** | **ep** | **epX** | **e**)

Expected first output (head moving from left to right):

_ | 1 | 0 | _ (represented as **e** | **epX** | **ep** | **e**)

Expected second output (head moving from right to left):

_ | 0 | 1 | _ (represented as **e** | **ep** | **epX** | **e**)

Element colors:

this color: represents a preload element.

this color: represents an unprocessed element.

this color: represents a processed element.

Other symbols used:

"|" separate preloaded elements.

"><" indicates the change in direction of the head.

TRANSITIONS TABLE ON TAPE ==========

Sequence example #1

operators used: **EMP**, **REC**

step: 1

input: -

database: []

operator: **EMP**

output: -

value: 0

remain: -

step: 2

input: **epX$_1$**

database: [**epX$_1$**]

operator: **REC**

output: **epX$_1$** > text on paper: " *current state **q0*** "

value: 1

remain: -

step: 3

input: **epX$_2$**

database: [**epX$_1$**, **epX$_2$**]

operator: **REC**

output: **epX$_2$** > text on paper: " *if read a symbol **e*** "

value: 1

remain: **epX$_1$** <---- (alternative remain)

step: 4

input: **epX_3**

database: [**epX_1**, **epX_2**, **epX_3**]

operator: **REC**

output: **epX_3** > text on paper: " *write a symbol* ***e*** "

value: 1

remain: **epX_1**, **epX_2** <---- (alternative remain)

step: 5

input: **epX_4**

database: [**epX_1**, **epX_2**, **epX_3**, **epX_4**]

operator: **REC**

output: **epX_4** > text on paper: " *move to the* ***right*** "

value: 1

remain: **epX_1**, **epX_2**, **epX_3** <---- (alternative remain)

step: 6

input: **epX_5**

database: [**epX_1**, **epX_2**, **epX_3**, **epX_4**, **epX_5**]

operator: **REC**

output: **epX_5** > text on paper: " *changes to state* ***q1*** "

value: 1

remain: **epX_1**, **epX_2**, **epX_3**, **epX_4** <---- (alternative remain)

.....

step: 7

input: **epX$_6$**

database: [**epX$_1$** ... **epX$_6$**]

operator: **REC**

output: **epX$_6$** > text on paper: " *current state **q1*** "

value: 1

remain: **epX$_1$** ... **epX$_5$** <---- (alternative remain)

step: 8

input: **epX$_7$**

database: [**epX$_1$** ... **epX$_7$**]

operator: **REC**

output: **epX$_7$** > text on paper: " *if read a symbol **ep*** "

value: 1

remain: **epX$_1$** ... **epX$_6$** <---- (alternative remain)

step: 9

input: **epX$_8$**

database: [**epX$_1$** ... **epX$_8$**]

operator: **REC**

output: **epX$_8$** > text on paper: " *write a symbol **epX*** "

value: 1

remain: **epX$_1$** ... **epX$_7$** <---- (alternative remain)

step: 10

input: **epX$_9$**

database: [**epX$_1$** ... **epX$_9$**]

operator: **REC**

output: **epX$_9$** > text on paper: " *move to the **right*** "

value: 1

remain: **epX$_1$** ... **epX$_8$** <---- (alternative remain)

step: 11

input: **epX$_{10}$**

database: [**epX$_1$** ... **epX$_{10}$**]

operator: **REC**

output: **epX$_{10}$** > text on paper: " *continues in state **q1*** "

value: 1

remain: **epX$_1$** ... **epX$_9$** <---- (alternative remain)

.....

step: 12

input: **epX$_{11}$**

database: [**epX$_1$** ... **epX$_{11}$**]

operator: **REC**

output: **epX$_{11}$** > text on paper: " *current state **q1*** "

value: 1

remain: **epX$_1$** ... **epX$_{10}$** <---- (alternative remain)

step: 13

input: **epX$_{12}$**

database: [**epX$_1$** ... **epX$_{12}$**]

operator: **REC**

output: **epX$_{12}$** > text on paper: " *if read a symbol **epX*** "

value: 1

remain: **epX$_1$** ... **epX$_{11}$** <---- (alternative remain)

step: 14

input: **epX$_{13}$**

database: [**epX$_1$** ... **epX$_{13}$**]

operator: **REC**

output: **epX$_{13}$** > text on paper: " *write a symbol **ep*** "

value: 1

remain: **epX$_1$** ... **epX$_{12}$** <---- (alternative remain)

step: 15

input: **epX$_{14}$**

database: [**epX$_1$** ... **epX$_{14}$**]

operator: **REC**

output: **epX$_{14}$** > text on paper: " *move to the **right*** "

value: 1

remain: **epX$_1$** ... **epX$_{13}$** <---- (alternative remain)

step: 16

input: **epX$_{15}$**

database: [**epX$_1$** ... **epX$_{15}$**]

operator: **REC**

output: **epX$_{15}$** > text on paper: " *continues in state **q1*** "

value: 1

remain: **epX$_1$** ... **epX$_{14}$** <---- (alternative remain)

.....

step: 17

input: **epX$_{16}$**

database: [**epX$_1$** ... **epX$_{16}$**]

operator: **REC**

output: **epX$_{16}$** > text on paper: " *current state **q1*** "

value: 1

remain: **epX$_1$** ... **epX$_{15}$** <---- (alternative remain)

step: 18

input: **epX$_{17}$**

database: [**epX$_1$** ... **epX$_{17}$**]

operator: **REC**

output: **epX$_{17}$** > text on paper: " *if read a symbol **e*** "

value: 1

remain: **epX$_1$** ... **epX$_{16}$** <---- (alternative remain)

step: 19

input: **epX$_{18}$**

database: [**epX$_1$** ... **epX$_{18}$**]

operator: **REC**

output: **epX$_{18}$** > text on paper: " *write a symbol **e*** "

value: 1

remain: **epX$_1$** ... **epX$_{17}$** <---- (alternative remain)

step: 20

input: **epX$_{19}$**

database: [**epX$_1$** ... **epX$_{19}$**]

operator: **REC**

output: **epX$_{19}$** > text on paper: " *move to the **left*** "

value: 1

remain: **epX$_1$** ... **epX$_{18}$** <---- (alternative remain)

step: 21

input: **epX$_{20}$**

database: [**epX$_1$** ... **epX$_{20}$**]

operator: **REC**

output: **epX$_{20}$** > text on paper: " *changes to state **q2*** "

value: 1

remain: **epX$_1$** ... **epX$_{19}$** <---- (alternative remain)

.....

step: 22

input: **epX$_{21}$**

database: [**epX$_1$** ... **epX$_{21}$**]

operator: **REC**

output: **epX$_{21}$** > text on paper: " *current state **q2*** "

value: 1

remain: **epX$_{1}$** ... **epX$_{20}$** <---- (alternative remain)

step: 23

input: **epX$_{22}$**

database: [**epX$_{1}$** ... **epX$_{22}$**]

operator: **REC**

output: **epX$_{22}$** > text on paper: " *if read a symbol **ep*** "

value: 1

remain: **epX$_{1}$** ... **epX$_{21}$** <---- (alternative remain)

step: 24

input: **epX$_{23}$**

database: [**epX$_{1}$** ... **epX$_{23}$**]

operator: **REC**

output: **epX$_{23}$** > text on paper: " *write a symbol **epX*** "

value: 1

remain: **epX$_{1}$** ... **epX$_{22}$** <---- (alternative remain)

step: 25

input: **epX$_{24}$**

database: [**epX$_{1}$** ... **epX$_{24}$**]

operator: **REC**

output: **epX$_{24}$** > text on paper: " *move to the **left*** "

value: 1

remain: **epX$_1$** ... **epX$_{23}$** <---- (alternative remain)

step: 26

input: **epX$_{25}$**

database: [**epX$_1$** ... **epX$_{25}$**]

operator: **REC**

output: **epX$_{25}$** > text on paper: " *continues in state **q2*** "

value: 1

remain: **epX$_1$** ... **epX$_{24}$** <---- (alternative remain)

.....

step: 27

input: **epX$_{26}$**

database: [**epX$_1$** ... **epX$_{26}$**]

operator: **REC**

output: **epX$_{26}$** > text on paper: " *current state **q2*** "

value: 1

remain: **epX$_1$** ... **epX$_{25}$** <---- (alternative remain)

step: 28

input: **epX$_{27}$**

database: [**epX$_1$** ... **epX$_{27}$**]

operator: **REC**

output: **epX$_{27}$** > text on paper: " *if read a symbol **epX*** "

value: 1

remain: **epX$_1$** ... **epX$_{26}$** <---- (alternative remain)

step: 29

input: **epX$_{28}$**

database: [**epX$_1$** ... **epX$_{28}$**]

operator: **REC**

output: **epX$_{28}$** > text on paper: " *write a symbol **ep*** "

value: 1

remain: **epX$_1$** ... **epX$_{27}$** <---- (alternative remain)

step: 30

input: **epX$_{29}$**

database: [**epX$_1$** ... **epX$_{29}$**]

operator: **REC**

output: **epX$_{29}$** > text on paper: " *continues in state **q2*** "

value: 1

remain: **epX$_1$** ... **epX$_{28}$** <---- (alternative remain)

.....

step: 31

input: **epX$_{30}$**

database: [**epX$_1$** ... **epX$_{30}$**]

operator: **REC**

output: **epX$_{30}$** > text on paper: " *current state **q2*** "

value: 1

remain: **epX$_1$** ... **epX$_{29}$** <---- (alternative remain)

step: 32

input: **epX$_{31}$**

database: [**epX$_1$** ... **epX$_{31}$**]

operator: **REC**

output: **epX$_{31}$** > text on paper: " *if read a symbol **e*** "

value: 1

remain: **epX$_1$** ... **epX$_{30}$** <---- (alternative remain)

step: 33

input: **epX$_{32}$**

database: [**epX$_1$** ... **epX$_{32}$**]

operator: **REC**

output: **epX$_{32}$** > text on paper: " *write a symbol **e*** "

value: 1

remain: **epX$_1$** ... **epX$_{31}$** <---- (alternative remain)

step: 34

input: **epX$_{33}$**

database: [**epX$_1$** ... **epX$_{33}$**]

operator: **REC**

output: **epX$_{33}$** > text on paper: " *move to the **right*** "

value: 1

remain: **epX$_1$** ... **epX$_{32}$** <---- (alternative remain)

step: 35

input: **epX$_{34}$**

database: [**epX$_{1}$** ... **epX$_{34}$**]

operator: **REC**

output: **epX$_{34}$** > text on paper: " *changes to state **q3** and stop execution* "

value: 1

remain: **epX$_{1}$** ... **epX$_{33}$** <---- (alternative remain)

[DELETE DATABASE]

INPUT DATA ON TAPE ==========

step: 36

input: -

database: []

operator: **EMP**

output: -

value: 0

remain: -

step: 37

input: **e** > represents symbol |_|

database: [**e**]

operator: **EMP**

output: **e**

value: 0

remain: -

step: 38

input: **ep** > represents symbol |0|

database: [**e**, **ep**]

operator: **REC**

output: **ep**

value: 0

remain: **e**

step: 39

input: **epX** > represents symbol |1|

database: [**e**, **ep**, **epX**]

operator: **REC**

output: **epX**

value: 1

remain: **e**, **ep**

step: 40

input: **e** > represents symbol |_|

database: [**e**, **ep**, **epX**, **e**]

operator: **EMP** (by decision)

output: **e**

value: 0

remain: **ep**, **epX**

[DELETE DATABASE]

DATA PROCESSING ==========

First head movements: LEFT to RIGHT

step: 41

input: -

database: []

operator: **EMP**

output: -

value: 0

remain: -

step: 42

input: **epX** > Preload data

database: [**epX**]

operator: **REC**

output: **epX**

value: 1

remain: -

step: 43 [STATE **q0**, READ]

input: **e** > represents symbol |_|

database: [epX | e]

operator: **EMP** (by decision)

output: e >>>> process status: _ | | |

value: 0

remain: epX

step: 44 [STATE **q1**, READ]

input: ep > represents symbol |0|

database: [epX | e, ep]

operator: **REC**

output: ep >>>> process status: _ | 0 | |

value: 0

remain: epX, e

Post-operation function: **write**

[write specific information to ep]

step: 45 [STATE **q1**, WRITE]

input: **e** > (only for transformation output)

database: [epX | e, **epX**, **e**]

operator: **REC**

output: **epX** >>>> process status: _ | 1 | |

value: 1

remain: e, **e**

step: 46 [STATE **q1**, READ]

input: **epX** > represents symbol |1|

database: [epX | **e**, **epX**, **e**, **epX**]

operator: **REC**

output: **epX** >>>> process status: _ | 1 | 1 |

value: 1

remain: **e**, **e**

Post-operation function: **erase**

[erase specific information from **epX**]

step: 47 [STATE **q1**, WRITE]

input: **e** > (only for transformation output)

database: [epX | **e**, **epX**, **e**, **ep**, **e**]

operator: **REC**

output: **ep** >>>> process status: _ | 1 | 0 |

value: 0

remain: epX, **e**, **epX**, **e**, **e**

step: 48 [STATE **q1**, READ]

input: **e** > represents symbol |_|

database: [epX | **e**, **epX**, **e**, **ep**, **e**, **e**]

operator: **EMP** (by decision)

output: **e** >>>> process status: _ | 1 | 0 | _ (expected first output)

value: 0

remain: epX, **epX**, **ep**

Second head movements: RIGHT to LEFT

step: 49 [STATE **q2**, READ]

input: **ep** > represents symbol |0|

database: [**epX** | **e**, **epX**, **e**, **ep**, **e**, **e** >< **ep**]

operator: **REC**

output: **ep** >>>> process status: _ | 1 | 0 | _

value: 0

remain: **epX**, **e**, **epX**, **e**, **e**, **e**

Post-operation function: **write**

[write specific information to **ep**]

step: 50 [STATE **q2**, WRITE]

input: **e** > (only for transformation output)

database: [**epX** | **e**, **epX**, **e**, **ep**, **e**, **e** >< **epX**, **e**]

operator: **REC**

output: **epX** >>>> process status: _ | 1 | 1 | _

value: 1

remain: **e**, **e**, **ep**, **e**, **e**, **e**

step: 51 [STATE **q2**, READ]

input: **epX** > represents symbol |1|

database: [**epX** | **e**, **epX**, **e**, **ep**, **e**, **e** >< **epX**, **e**, **epX**]

operator: **REC**

output: **epX** >>>> process status: _ | 1 | 1 | _

value: 1

remain: **e**, **e**, **ep**, **e**, **e**, **e**

Post-operation function: **erase**

[erase specif information from **epX**]

step: 52 [STATE **q2**, WRITE]

input: **e** > (only for transformation output)

database: [**epX** | **e**, **epX**, **e**, **ep**, **e**, **e** >< **epX**, **e**, **ep**, **e**]

operator: **REC**

output: **ep** >>>> process status: _ | 0 | 1 | _

value: 0

remain: **epX**, **e**, **epX**, **e**, **e**, **e**, **epX**, **e**, **e**

step: 53 [STATE **q2**, READ]

input: **e** > represents symbol |_|

database: [**epX** | **e**, **epX**, **e**, **ep**, **e**, **e** >< **epX**, **e**, **ep**, **e**, **e**]

operator: **EMP** (by decision)

output: **e** >>>> process status: _ | 0 | 1 | _ (expected second output)

value: 0

remain: **epX**, **epX**, **ep**, **epX**, **ep**

Third head movement: LEFT to RIGHT

step: 54 [STATE **q3**, ONLY MOVEMENT]

input: **ep** > represents symbol |0|

database: [epX | e, epX, e, ep, e, e >< epX, e, ep, e, e, ep]

operator: **REC**

output: ep >>>> process status: _ | 0 | 1 | _

value: 0

remain: epX, e, epX, e, e, e, epX, e, e, e

[STOP EXECUTION]

Córdoba, Argentina, January, 31, 2025.