Emulating a Turing machine with DSF

Damian Fascio

This simple Turing machine swaps the binary values 0 and 1, leaves the empty places intact, and changes the non-binary value -1 to an empty place.

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

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).

**epY** (numerical value -1, represents a cell with the non-binary number -1)

**e** (nunerical value 0, represents a blank cell).

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

The main operation of this Turing machine is carried out through the post-operation functions (**erase**, **write**, **extract**, **attach**).

Data on tape:

Input: 0 | -1 | 1 | _ | _ (represented as **ep** | **epY** | **epX** | **e** | **e**)

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

Transition table:

State ***Q1*** --> read **ep** --> write **epX** --> move to the right --> continues in the state ***Q1***

State ***Q1*** --> read **epX** --> write **ep** --> move to the right --> continues in the state ***Q1***

State ***Q1*** --> read **epY** --> write **e** --> move to the right --> continues in the state ***Q1***

State ***Q1*** --> read **e** --> write **e** --> move to the right --> changes to the state ***Q2***

State ***Q2*** --> read **e** --> write **e** --> move to the right --> changes to the state ***Q3***

State ***Q3*** --> stop execution

*Sequence example #1*

Sequence A > STATE ***Q1***

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

step: 1

input: -

database: []

operator: **EMP**

output: -

value: 0

remain: -

step: 2 > Preload **epX** element

input: **epX**

database: [**epX**]

operator: **ONL**

output: **epX**

value: 1

remain: -

step: 3 > Read the initial first symbol from left to right (**ep**)

input: **ep**

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

operator: **REC**

output: **ep**

value: 0

remain: **epX**

--- Post-operation function ---

write specific information to: **ep**

step: 4 > Write the new first symbol from left to right (**epX**)

input: **e**

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

operator: **REC**

output: **epX** <---- first cell new symbol

value: 1

remain: **e**

> DELETING DATABASE FROM STEP 3 AND MOVING HEAD TO THE RIGHT (CONTINUES IN STATE ***Q1***)

. . . . . .

step: 3 > Read the initial second symbol from left to right (**epY**)

input: **epY**

database: [**epX** | **epY**]

operator: **REC**

output: **epY**

value: -1

remain: **epX**

--- Post-operation function ---

extract specific information from: **epY**

step: 4 > Write the new second symbol from left to right (**e**)

input: **e**

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

operator: **EMP** (by decision)

output: **e** <---- second cell new symbol

value: 0

remain: **epX**

> DELETING DATABASE FROM STEP 3 AND MOVING HEAD TO THE RIGHT (CONTINUES IN STATE ***Q1***)

. . . . . .

step: 3 > Read the initial third symbol from left to right (**epX**)

input: **epX**

database: [**epX** | **epX**]

operator: **REC**

output: **epX**

value: 1

remain: -

--- Post-operation function ---

erase specific information from: **epX**

step: 4 > Write the new third symbol from left to right (**ep**)

input: **e**

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

operator: **REC**

output: **ep** <---- third cell new symbol

value: 0

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

> DELETING DATABASE FROM STEP 3 AND MOVING HEAD TO THE RIGHT (CONTINUES IN STATE ***Q1***)

. . . . . .

step: 3 > Read the initial fourth symbol from left to right (**e**)

input: **e**

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

operator: **EMP** (by decision)

output: **e**

value: 0

remain: **epX**

step: 4 > Write the new fourth symbol from left to right (**e**)

input: **e**

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

operator: **EMP** (by decision)

output: **e** <---- fourth cell new symbol

value: 0

remain: **epX**

> HEAD MOVEMENT TO THE RIGHT AND CHANGE TO STATE ***Q2***

==========

*Sequence example #1*

Sequence B > STATE ***Q2***

operators used: **EMP**

step: 1

input: -

database: []

operator: **EMP**

output: -

value: 0

remain: -

step: 2 > Read the initial fifth symbol from left to right (**e**)

input: **e**

database: [**e**]

operator: **EMP**

output: **e**

value: 0

remain: -

step: 3 > Write the new fifth symbol from left to right (**e**)

input: **e**

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

operator: **EMP**

output: **e** <---- fifth cell new symbol

value: 0

remain: -

> HEAD MOVEMENT TO THE RIGHT AND CHANGE TO STATE ***Q3***

==========

*Sequence example #1*

Sequence C > STATE ***Q3***

operators used: **EMP**

step: 1

input: -

database: []

operator: **EMP**

output: -

value: 0

remain: -

> STOP EXECUTION

Córdoba, Argentina, December, 31, 2024.