Exercises of chapter 3

February 12, 2020

Mayra Cristina Berrones Reyes

1 Turing machines

Turing machine (TM) was invented by Alan Turing in 1993, and consists of a tape with an infinite length, on which read and write operations can be done, using a certain alphabet, and with the instructions of a transition state table, it solves a certain task.

On this document we demonstrate some examples of the use of a TM on different tasks. First we define the special symbols that show up in every example:

- Σ is our alphabet.
- $\triangleright$ is the start of the tape.
- $-$ means no movement.
- $\sqcup$ is a blank space.
- $\leftarrow$ means move to the left.
- $\rightarrow$ means move to the right.

1.1 Example 1: Multiply a binary number by two

Define a Turing machine that produces a binary number (provided as input) multiplied by two.

In this TM our alphabet is $\Sigma = \{0, 1, \triangleright, \sqcup, \leftarrow, \rightarrow, -\}$ and our transition states are s, q and *stop*.

The transition table is shown in Table 1.

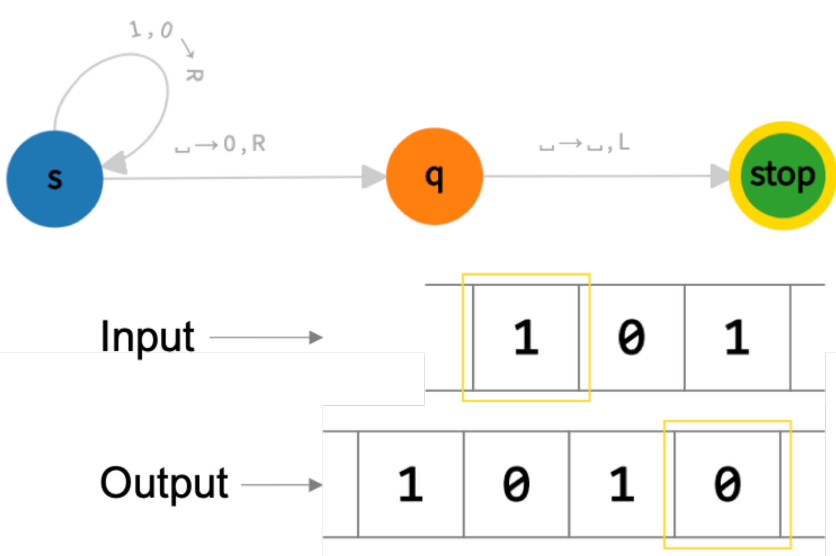
Table 1: Transition table for example 1

State	**Transition**
s, $\triangleright$	(s, $\triangleright$, $\rightarrow$)
s, 0	(s, 0, $\rightarrow$)
s, 1	(s, 1, $\rightarrow$)
s, $\sqcup$	(q, 0, $\rightarrow$)
q, $\sqcup$	($stop$, $\sqcup$, $-$)

```
input: '101'
blank: ' '
start state: s
table:
  s:
    [1,0]: R
    ' '  : {write: 0, R: q}
  q:
    ' ' : {write: ' ', L: stop}
  stop:


```

Listing 1: TM Visualization code

Figure 1: Diagram of all the transitions and the result of an example.



1.2 Example 2: Contains at least one letter

Given the alphabet $\Sigma = \{a, b, c, \triangleright, \sqcup, \leftarrow, \rightarrow, -\}$, the TM must check that at least one of each letter a, b, c appears on a string. This time we have seven states (*start*, and *case*2 to 7), and two acceptance states *accept* and *reject*.

The transition table is shown in Table 2.

Table 2: Transition table for example 2

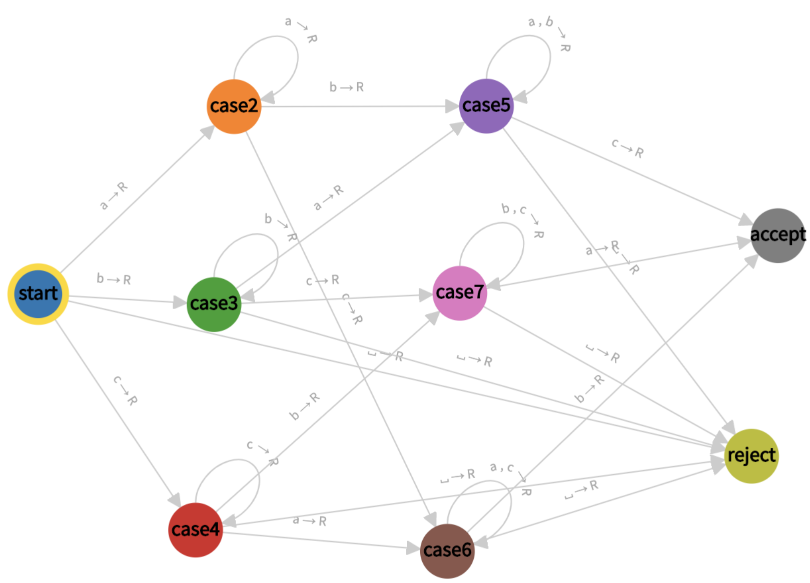
State	Transition	State	Transition	State	Transition
$start$, $\triangleright$	($start$, $\triangleright$, $\rightarrow$)	$case2$, a	($case2$, a, $\rightarrow$)	$case3$, a	($case5$, a, $\rightarrow$)
$start$, a	($case2$, 0, $\rightarrow$)	$case2$, b	($case5$, b, $\rightarrow$)	$case3$, b	($case3$, b, $\rightarrow$)
$start$, b	($case3$, 1, $\rightarrow$)	$case2$, c	($case6$, c, $\rightarrow$)	$case3$, c	($case7$, c, $\rightarrow$)
$start$, c	($case4$, 0, $\rightarrow$)	$case2$, $\sqcup$	($reject$, $\sqcup$, $-$)	$case3$, $\sqcup$	($reject$, $\sqcup$, $-$)
$start$, $\sqcup$	($reject$, $\sqcup$, $-$)				
State	**Transition**	**State**	**Transition**	**State**	**Transition**
$case4$, a	($case6$, a, $\rightarrow$)	$case5$, a	($case5$, a, $\rightarrow$)	$case6$, a	($case6$, a, $\rightarrow$)
$case4$, b	($case7$, b, $\rightarrow$)	$case5$, b	($case5$, b, $\rightarrow$)	$case6$, b	($accept$, b, $-$)
$case4$, c	($case4$, c, $\rightarrow$)	$case5$, c	($accept$, $\sqcup$, $-$)	$case6$, c	($case6$, c, $\rightarrow$)
$case4$, $\sqcup$	($reject$, $\sqcup$, $-$)	$case5$, $\sqcup$	($reject$, $\sqcup$, $-$)	$case6$, $\sqcup$	($reject$, $\sqcup$, $-$)
State	**Transition**				
$case7$, a	($accept$, a, $-$)				
$case7$, b	($case7$, b, $\rightarrow$)				
$case7$, c	($case7$, c, $\rightarrow$)				
$case7$, $\sqcup$	($reject$, $\sqcup$, $-$)				

```
input: 'ccaacb'
blank: ' '
start state: start
synonyms:
  accept: {R: accept}
  reject: {R: reject}

table:
  start:
    a: {R: case2}
    b: {R: case3}
    c: {R: case4}
    ' ': reject
  case2:
    a: R
    b: {R: case5}
    c: {R: case6}

  case3:
    b: R
    a: {R: case5}
    c: {R: case7}
    ' ': reject
```

```
case4:
  c: R
  a: {R: case6}
  b: {R: case7}
  ' ': reject

case5:
  [a,b]: R
  c: accept
  ' ': reject
case6:
  [a,c]: R
  b: accept
  ' ': reject

case7:
  [b,c]: R
  a: accept
  ' ': reject

accept:
reject:


```

Listing 2: TM Visualization code

Figure 2: Diagram of all the transitions and the result of an example.



The input for this TM is the string *ccaacb*, and the output is *accept*. If we change that string to *caacc* the output is *reject*.

1.3 Example 3: Know if a string is a palindrome

Given the alphabet $\Sigma = \{a, b, c, \triangleright, \sqcup, \leftarrow, \rightarrow, -\}$, the TM must take the string and check if is a palindrome or not. This time we have eight states (*start*, and *case*2 to 8), and two acceptance states *accept* and *reject*.

The transition table is shown in Table 3.

Table 3: Transition table for example 3

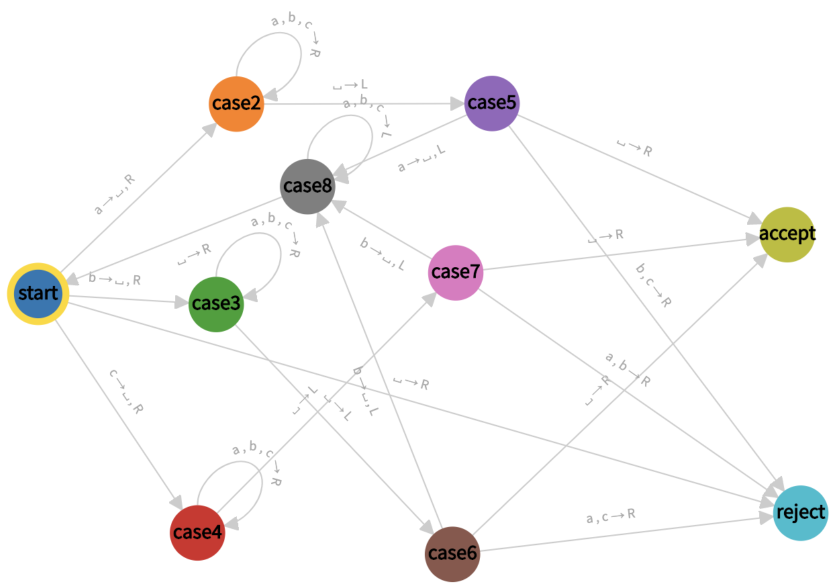
State	Transition	State	Transition	State	Transition
$start$, $\triangleright$	$(start, \triangleright, \rightarrow)$	$case2$, a	$(case2, a, \rightarrow)$	$case3$, a	$(case3, a, \rightarrow)$
$start$, a	$(case2, \sqcup, \rightarrow)$	$case2$, b	$(case2, b, \rightarrow)$	$case3$, b	$(case3, b, \rightarrow)$
$start$, b	$(case3, \sqcup, \rightarrow)$	$case2$, c	$(case2, c, \rightarrow)$	$case3$, c	$(case3, c, \rightarrow)$
$start$, c	$(case4, \sqcup, \rightarrow)$	$case2$, $\sqcup$	$(\sqcup, \sqcup, \leftarrow)$	$case3$, $\sqcup$	$(\sqcup, \sqcup, \leftarrow)$
$start$, $\sqcup$	$(reject, \sqcup, -)$				
State	**Transition**	**State**	**Transition**	**State**	**Transition**
$case4$, a	$(case4, a, \rightarrow)$	$case5$, a	$(case8, \sqcup, \leftarrow)$	$case6$, a	$(reject, a, \rightarrow)$
$case4$, b	$(case4, b, \rightarrow)$	$case5$, b	$(reject, b, \rightarrow)$	$case6$, b	$(case8, \sqcup, \leftarrow)$
$case4$, c	$(case4, c, \rightarrow)$	$case5$, c	$(reject, c, \rightarrow)$	$case6$, c	$(reject, c, \rightarrow)$
$case4$, $\sqcup$	$(\sqcup, \sqcup, \leftarrow)$	$case5$, $\sqcup$	$(accept, \sqcup, -)$	$case6$, $\sqcup$	$(accept, \sqcup, -)$
State	**Transition**	**State**	**Transition**		
$case7$, a	$(reject, a, \rightarrow)$	$case8$, a	$(case8, a, \leftarrow)$		
$case7$, b	$(reject, b, \rightarrow)$	$case8$, b	$(case8, b, \leftarrow)$		
$case7$, c	$(case8, \sqcup, \leftarrow)$	$case8$, c	$(case8, c, \leftarrow)$		
$case7$, $\sqcup$	$(accept, \sqcup, -)$	$case8$, $\sqcup$	$(start, \sqcup, \rightarrow)$		

```
input: 'aacbbcaa'
blank: ' '
start state: start
synonyms:
  accept: {R: accept}
  reject: {R: reject}

table:
  start:
    a: {write: ' ', R: case2}
    b: {write: ' ', R: case3}
    c: {write: ' ', R: case4}
    ' ': reject
  case2:
    [a,b,c]: R
    ' ' : {L: case5}
  case3:
    [a,b,c]: R
    ' ' : {L: case6}
```

```
  case4:
    [a,b,c]: R
    ' ' : {L: case7}
  case5:
    [b,c]: {R: reject}
    a: {write: ' ', L: case8}
    ' ': accept
  case6:
    [a,c]: {R: reject}
    b: {write: ' ', L: case8}
    ' ': accept

  case7:
    [a,b]: {R: reject}
    b: {write: ' ', L: case8}
    ' ': accept
  case8:
    [a,b,c]: L
    ' ' : {R: start}

  accept:
  reject:
```

Listing 3: TM Visualization code

Figure 3: Diagram of all the transitions and the result of an example.



The input for this TM is *aacbbcaa* and the output is *accept*. If we enter as input *abbca* the output is *reject*.

2 Conclusions

As a personal opinion, I think is easier to understand how a TM works if we drafted as a graph first. It also helps to visualize the transition tables and see where it can lead to an error. One of the tricks that I learned at the end of this exercise is to start with a small example, and then ask yourself more difficult questions every time you successfully finish one TM, to see if it still works properly.

3 Reference

To make the graphs we used `https://turingmachine.io/`.