CS 561 Project 2 : Turing machine simulator and non-computable functions, aka Busy Beaver machine

Due on December 14, 2019

1 Project Description

From wikipedia.org

> A **Turing machine** is a hypothetical device that manipulates symbols on a strip of tape according to a table of rules. Despite its simplicity, a Turing machine can be adapted to simulate the logic of any computer algorithm, and is particularly useful in explaining the functions of a CPU inside a computer.
> The "Turing" machine was invented in 1936 by Alan Turing who called it an "a-machine" (automatic machine). The Turing machine is not intended as practical computing technology, but rather as a hypothetical device representing a computing machine. Turing machines help computer scientists understand the limits of mechanical computation.
>
> **Computable functions** are the basic objects of study in computability theory. Computable functions are the formalized analogue of the intuitive notion of algorithm. They are used to discuss computability without referring to any concrete model of computation such as Turing machines or register machines. Any definition, however, must make reference to some specific model of computation but all valid definitions yield the same class of functions. Particular models of computability that give rise to the set of computable functions are the Turing-computable functions and the μ - recursive functions.
>
> In computability theory, a **busy beaver** is a Turing machine that attains the maximum number of steps performed, or maximum number of nonblank symbols produced finally on the tape, among all Turing machines in a certain class. The Turing machines in this class must meet certain design specifications and are required to eventually halt after being started with a blank tape.
> A busy beaver function quantifies these upper limits on a given measure, and is a noncomputable function. In fact, a busy beaver function can be shown to grow faster asymptotically than does any computable function. The concept was first introduced by Tibor Rado as the "busy beaver game" in his 1962 paper, "On Non-Computable Functions".

There are two parts for this programming project:

1. Program a Turing Machine simulator.
2. Solutions to a variant of the Busy Beaver problem, i.e., design four specific Turing Machines.

Finish project in your teams and one of the team member should submit the artifacts on onyx using the following submit command : `submit elenasherman cs561 p2`. The submission must be done by the end of April 22. Please contact your instructor if you have problems with the submission.

2 Turing Machine Simulator

2.1 Design Requirements

The objective is to write an efficient Turing Machine simulator, i.e., the faster the better. The simulator should simulate a variant of Turning Machine with a **bi-infinite tape**, i.e., an infinite in both directions tape. You can assume that all Turing Machines are deterministic and eventually will halt. The machine only has one halting (accepting) state. There is no reject state. Your simulator stops when it reaches the halting state of the machine. At the end of the simulation the content of **visited** tape cells should be printed out to the stdout.

We assume that the usual blank symbol, i.e., $\sqcup$ is now 0 symbol in the tape alphabet. Thus, 0 is not a part of the input string alphabet Σ, but always present in the tape's alphabet Γ, which is in our case $\Gamma = \{0\} \cup \Sigma$. The symbols of the input string alphabet are represented by integer numbers starting at number 1.

States of the TM Q are labeled with integer numbers. We always assume the state with label 0 to be the start state and the halting sate is the state with the largest label, i.e., $|Q| - 1$.

Your simulator should accept a single command line argument which is the name of the input file with the encoding of a Turing machine and the input string.

2.2 Input Specifications

The first line of an input file displays the total number of states in a TM. If there are n states, then state 0 is the start state and state n is the halting state. The second line shows the number of symbols in Σ. If there are m symbols then $\Sigma = \{1, 2, \ldots, m\}$. We set m to be at most 9.

The next $|\Gamma| \times |Q|$ lines define transition function where the current state (from state) and inputs symbol (symbol under the tape head) are defined by the line number of the transition. For example, for a machine with 2 states and 1 symbol in Σ the first transition line describes the tradition from state 0 on symbol 0, i.e., blank. The second transition line describes the transition from state 0 on symbol 1. Third line the transition from state 1 (halting in this case) on symbol 0, and the fourth line the transition from state 1 on symbol 1.

Each transition line will be the following comma delimitated format:

$$next\_state, write\_symbol, move$$

where

- *next_state* – the state that the machine transitions to.
- *write_symbol* – the symbol the machine will write to the cell.

- *move* – the direction, either L or R, to move the tape head to.

The last line of the file will contain the input string for the machine, followed by a carriage return. If this line is blank then the tape should be initialized to blanks (all 0s).

See demo section for example input files.

2.3 Output Specifications

At the end of the computation, your Turing Machine simulator should print out the contents of the visited tape squares followed by a single carriage return. Do NOT print out anything else.

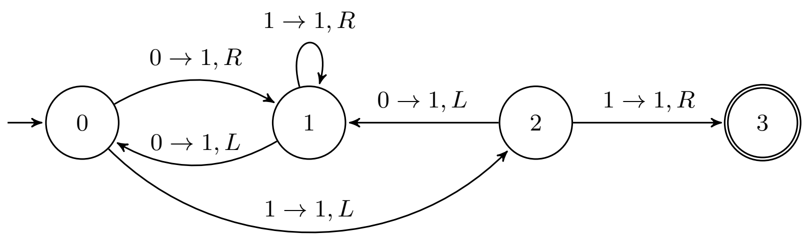
See demo section for output examples.

2.4 Program Specifications

You may use any language that is available on onyx, so that we can compile and run your code on onyx. However, you cannot use any turning machine libraries in any language. The executable must be named **MainProj2**.

3 Busy Beaver

Essentially Busy Beaver is a Turing Machine that is the solution to the Busy Beaver problem, which is defined as the maximum number of (not necessary consecutive) 1s left on the tape after n-state Turing Machine with $\Sigma = \{1\}$ halts. Below is the graphical representation of a 3-state Busy Beaver TM.



If this machine is started with a tape of all blanks, it halts after 13 moves with six consecutive 1s on the tape, i.e., the sum of symbols on the tape is 6. If a Turing Machine writes the maximum possible number of 1s for its number estates the it is called a "busy beaver". See more at `http://www.win.tue.nl/~wijers/shallit.pdf`

You will have to find 4 solutions for a variant of Busy Beaver problem for $n = 1$, $n = 2$, $n = 3$ and $n = 4$ that starts with a blank tape and uses the tape symbols $\Gamma = \{0, 1, 2, 3\}$ (recall 0 is blank), that maximizes the sum of symbols on the tape, i.e., the sum of 1s, 2s and 3s on the tape. Remember that machine must halt. If your solution of Busy Beaver problem does not halt within 3 minutes it will receive a score of 0.

The solutions to 4 Busy Beaver problems should be encoded in the file format specified in Section 2.2.

4 Submission

Your project should be submitted electronically using `submit` command on onyx. You must have onyx account to do so. If you don't have one, please let your instructor know.

You should submit:

1. All source files for the Turing Machine simulator.
2. Four files of the solutions to the variant of Busy Beaver problem in the specified format.
3. 3 to 5 pages single spaced typed report that explains your approach to designing the simulator, and your approach finding four Busy Beaver machines. For example, what optimization you've implemented and what search technique did you use. Share interesting external information as well as your personal insights that helped you with the project. The report should be engaging yet professional.
4. A .txt README file which briefly describes all submitted files and how to compile them.

Note for late submissions: 10% will be deducted per each day.

5 Grading

Grading will be based on the following elements:

- (5%) Have you submitted all required files (including README.txt)
- (5%) Is the source code easy to understand (i.e., good structure and comments)
- (5%) Can the code be compiled on onyx?
- (5%) Does your program read the single argument and print out to the standard out (i.e., screen)?
- (40%) Does your program run correctly?
- (5%) Busy Beaver of $n = 1$ in correct format
- (5%) Busy Beaver of $n = 2$ in correct format
- (5%) Busy Beaver of $n = 3$ in correct format
- (5%) Busy Beaver of $n = 4$ in correct format
- (20%) Report

6 Demo

The above Busy Beaver $n = 3$ and $\Sigma = \{1\}$ is encoded as follows with initially 1101 on the tape.

```
[esherman@onyx cs361]$ cat BB31101.txt
4
1
1,1,R
2,1,L
0,1,L
1,1,R
1,1,L
3,1,R
1101
```

In order to indicate that the machine starts with the empty tape leave that last line blank, i.e.,

```
[esherman@onyx cs361]$ cat BB.txt
4
1
1,1,R
2,1,L
0,1,L
1,1,R
1,1,L
3,1,R

EOF
```

The executable is "MainProj3". Then

```
[esherman@onyx cs361]$java MainProj2 BB31101.txt
1111111
```

Likewise for the second file.

```
[esherman@onyx cs361]$java MainProj2 BB.txt
111111
```