Logarithm Turing Machine (Assessed Individual Coursework, Foundations 2 (F29FB), Spring 2021)

2021-02-28 (Sunday)

Contents

1 general information

Due by: 2021-03-04 T 15:30 (Edinburgh time for Edinburgh students, Dubai time for Dubai students)

This coursework is 18% of the mark for the course.

This is individual work. Failure to reference, collusion, and plagiarism are all FORBIDDEN. Read the university's guidelines to understand what these offenses are.

The standard university late submission guidelines are in force for this coursework.

You must submit via VISION. Submission via email is FORBIDDEN.

2 the cblog function

Some uses of **bold font** indicate references to terminology that is also discussed or defined in the lecture notes.

Let cblog (the composition of **Ceiling** with Binary **LOGarithm**) be the unique **function** such that:

- cblog $\in \mathbb{N} \nrightarrow \mathbb{N}$
- cblog(0) is undefined

- for every $k \in \mathbb{N} \setminus \{0\}$, it holds that $\mathrm{cblog}(k) = j \in \mathbb{N}$ where j is the smallest **natural number** such that $k \leq 2^j$.

This is a simple function:

- cblog(0) is undefined.
- Because $2^0 = 1$, it holds that cblog(1) is 0.
- Because $2^1 = 2$, it holds that cblog(2) is 1.
- Because $2^2 = 2 \times 2 = 4$, it holds that cblog(3) = cblog(4) = 2.
- Because $2^3 = 2 \times 2 \times 2 = 8$, it holds that cblog(5) = cblog(6) = cblog(7) = cblog(8) = 3.
- Because $2^4 = 2 \times 2 \times 2 \times 2 = 16$, it holds that cblog(9) through cblog(16) are all 4.
- The general rule is that $\mathrm{cblog}(2^k) + 1$ through $\mathrm{cblog}(2^{k+1})$ are all $k + 1$.

3 main task: implement cblog with a Turing machine

YOUR MAIN TASK IS: Design a **Turing machine** M_g such that $\mathrm{cblog} = \mathrm{unaryIO}_{1,1}(M_g)$, where the meaning of this equation is as follows:

When $k \in \mathbb{N}$ and your Turing machine M_g is started on a **tape** t that contains only **blank tape symbols** except for k consecutive 1's placed on the tape starting at position 0 (the initial **head position**) and going to the right, the outcome is as follows:

- If $\mathrm{cblog}(k) = j \in \mathbb{N}$, then the **computation sequence** must return the result by **halting** with a **final tape** t' that contains only blank tape symbols except for j consecutive 1's starting at the final head position and going to the right.
- If cblog(k) is not defined, then your Turing machine must signal that the result is undefined by doing one of these two things:
  - The computation sequence may halt with a final tape t' that is an invalid output tape when expecting a single natural number as output. There are three ways to do this:
    - Leave a **symbol** other than 1 or ˆ on the tape.
    - Leave a 1 on the tape to the left of the final head position.
    - Leave more than one number on the tape (i.e., leave a ˆ symbol somewhere between two 1's).
  - The computation sequence may **diverge** (not halt).

So for example, because cblog(4) = 2, if your Turing machine M_g is started with the TM head at the leftmost 1 on the tape

```
1111
```

which is the number 4 in **unary** notation, your TM should finally halt with the TM head at the leftmost 1 on the tape

```
11
```

which is the number 2 in unary notation.

4 slight variations on the definition of Turing machine for this coursework

In this assignment, a TM **state** is any string that matches the regular expression [A-Za-z][-A-Za-z0-9_<>]*, which means that it must begin with a Latin letter and the subsequent characters may also be digits, - (the hyphen), _ (the underscore), and < and > (the less-than and greater-than symbols). If you want to follow the lecture notes, you can write, e.g., q_{12} as q_12.

For the purposes of connecting this coursework to the formalism in the lecture notes, we consider q_k to be the state whose name is of the form q_DIGITS where DIGITS is the standard decimal representation of $k \in \mathbb{N}$. For each state whose name does not match the pattern q_DIGITS, we will allocate an unused k and identify the state with q_k.

You must name your **initial state** q_0. If you submit a TM with no **actions** in state q_0, you will be deemed to have submitted a useless TM that does nothing.

In this assignment, a **symbol** must be one of these 37 characters:

```
^ 0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

We identify s_0 with ^ and s_1 with 1 and assume there is a similar identification for the other 35 symbols.

You must use ^ (the caret, a.k.a. the circumflex accent) as the blank tape symbol and 1 (the digit 1) as the symbol for the digit 1 in the unary input/output.

5 files to submit

You must submit two mandatory files:

- report.pdf
- turing-machine-action-table.txt

5.1 report.pdf

You must submit 1 file named exactly "report.pdf".

This file must be a PDF file in the format defined by Adobe.

Everything must be properly typeset. Handwriting is FORBIDDEN.

The first line of this file must read exactly "Logarithm Turing Machine Individual Coursework, F29FB, Spring 2021, H00123456", where 123456 must be replaced by your student ID number.

Immediately after the first line must follow the following numbered items with these exact numbers.

1. Item 1 must be your action table converted to graph format and presented beautifully. The initial state must be clearly marked.

   Item 1 must use at most 2 pages (preferably 1).

2. Item 2 must present the computation sequence of your TM on the input value 3 using the course's unary input/output conventions. The sequence must be annotated with a brief discussion of how your TM handles this input and the phases it goes through while doing this.

   Item 2 must use at most 2 pages (preferably 1).

3. Item 3 must be exactly (no more, no less) a mathematical expression that represents your action table in correct mathematical notation.

   Item 3 must use at most half a page.

4. Item 4 must be a clear, precise, and brief explanation in English of how your TM accomplishes, or fails to accomplish, the computation of the function cblog. This discussion must include exactly these lettered sub-items with these exact letters:

   A. Subitem A must briefly summarize in at most 5 sentences how your TM accomplishes the computation of cblog.
   B. Subitem B must briefly discuss the symbols you use and what they mean for your TM.
   C. Subitem C must briefly discuss at most 3 (preferably fewer) key loops in your TM that involve more than one state. Discuss what each of these loops accomplishes and how this contributes to computing cblog. It is a good idea to identify 1 or 2 key invariant properties that always hold at some specific loop points.
   D. Subitem D must briefly discuss all failures to compute cblog for particular inputs. You must discuss all failures for any input from 0 through 100 (inclusive). You should have a good idea of the failure pattern of your TM above the input 100 and you should also present this clearly. You must present the failure cases in at most 3 categories and explain why your TM fails for each category (e.g., if your TM fails on 70 numbers through 100, we do not want 70 separate discussions but instead at most 3, preferably fewer).
   E. Subitem E must briefly discuss 1 or 2 of your most significant design choices. You should consider discussing not only choices that helped but also choices that might have made your work harder.
   F. Subitem F must briefly discuss the efficiency of your TM. You should consider, in relation to the value/size of the input, both (1) the number of steps in the computation and (2) the size of the portion of the tape that is actually used at any/all points throughout the computation.

   Item 4 must use at most 1.5 pages.

5. Item 5 must briefly discuss how you would calculate cblog with a TM (1) if you were using binary input/output (as defined by binaryIO in Lecture Notes 14) or (2) if you were using binary input and unary output. Your discussion must cover how similar/different these 2 problems are to the problem your TM was required to solve.

   Item 5 must use at most half a page (and preferably much less).

6. Item 6 must contain anything you think is important to write as part of your submission but which is not covered by the above list items. This part is optional, but if present must be numbered 6 and must contain all material not in items numbered 1 through 5.

There must be nothing after the list of items specified above.

This file must be at most 6 pages (preferably fewer). Any pages after the 6th will be automatically removed before marking. Do not include a title page.

5.2 turing-machine-action-table.txt

You must submit 1 file named exactly "turing-machine-action-table.txt".

5.2.1 required format and content of turing-machine-action-table.txt

Here is an example of what this file might look like for a simple TM that erases the first unary number on the tape and puts a 1 in the space that was after that number:

```
# H00123456
{
  ((q_0, 1), (q_0, ^, R)),
  ((q_0, ^), (q_1, 1, 0)),
}
```

This file must contain the exact same TM that you present in report.pdf. There is no separate mark for this file.

Your TM will be tested for correctness of its input/output behavior for at least the first 100 natural numbers and possibly a selection of larger numbers. Your TM will be allowed up to 10 seconds (using the lecturer's TM simulator) and 10 000 000 steps for each computation sequence.

The file turing-machine-action-table.txt must have the following very precise format. A line can be empty, in which case it is ignored. A line can begin with "#", in which case it is a comment for any human reader and is not part of the TM. Comments in this file are primarily for the student author and the marker is unlikely to read them. The first non-empty non-comment line must contain exactly "{". A line can be of this format:

```
((state,symbol),(state,symbol,move)),
```

In this case, the first state and symbol are the current (old) state and symbol of an action, and the second state and symbol are the new state and symbol of the same action. The last non-empty non-comment line must contain exactly "}". There are no other formats for lines.

The first line must be a comment that contains your student ID number (which looks like H00123456) and nothing else.

All space (32) and tab (9) characters are ignored.

If you get the format of this file wrong, your file will be automatically considered to fail all tests.

If the contents of this file fail to denote a function, your file will be automatically considered to fail all tests.

If the file contains no actions for state q_0, then the automated testing will think your TM always halts immediately. You must name your initial state q_0.

If you fail to submit this file or get the format wrong, your mark will suffer because the automated testing will say that your TM handles every input incorrectly.

5.2.2 program for checking your turing-machine-action-table.txt file (STRONGLY RECOMMENDED)

There is a file you can download at this URL (all on one line with no spaces or line breaks):

```
https://www.macs.hw.ac.uk/~jbw/teaching/hw/F29FB/2020,-21/
  f29fb-2020,-21-cw-checker.py
```

It is also available at and runnable from this file system location on the NFS file systems of the school of MACS in Edinburgh:

```
/home/jbw/f29fb-2020,-21-course-support/f29fb-2020,-21-cw-checker.py
```

This is a Python program which you can use to check your turing-machine-action-table.txt file for syntactic and semantic correctness.

IT IS STRONGLY RECOMMENDED THAT YOU USE THIS PROGRAM MANY TIMES BEFORE YOU SUBMIT!!!

For further instructions, run the program and/or read the documentation near the beginning of the program.

6 marks

Let Rn mean item n in the file report.pdf. Let R4X mean subitem X of item 4 in the file report.pdf. Let TMAT mean the file turing-machine-action-table.txt.

Let TM mean R1, R3, and TMAT taken together. TM is your Turing machine.

Let Understanding mean R2, R4A, R4B, R4C, and R4D, possibly augmented by useful annotations in R1, which present your Understanding of how/why your TM computes or fails to compute cblog.

The marking will be as follows:

9	accuracy of computation of cblog by your TM
8	correctness, clarity, insight, and completeness of your presentation of your Understanding
3	clarity and all other aspects of correctness (e.g., syntax, notation) of your TM
3	R4E (wisdom you may have gained from or further developed in the coursework)
2	R4F (understanding of efficiency)
2	R5 (extrapolation of your learning to related but different problems)

7 questions and answers about the coursework

7.1 what are binary logarithms?

Recall the definition of logarithm: if there exists a number y such that $b^y = x$, then $\log_b x = y$. The result is that if $\log_b x$ is defined, then $b^{(\log_b x)} = x$.

Binary logarithm is the function $\log_2$.

7.2 what is ceiling?

Recall the definition of ceiling from Lecture Notes 2. Formally, $\lceil \cdot \rceil$ is a function such that for all $x \in \mathbb{R}$ it holds that $x \leq \lceil x \rceil$ and $\lceil x \rceil \in \mathbb{Z}$ and for all $k \in \mathbb{Z}$ it holds that if $x \leq k$ then $x \leq \lceil x \rceil \leq k$. Informally, the ceiling of x is the nearest integer to x that is not less than x.

7.3 why is the function named "cblog"?

For this coursework you do not need to understand anything about logarithms other than the definition of cblog given above, but you might appreciate knowing a bit more.

In fact, cblog is the unique function such that

- cblog $\in \mathbb{N} \nrightarrow \mathbb{N}$
- for all $k \in \mathbb{N}$ it holds that cblog(k) = $\lceil \log_2 k \rceil$.

In other words, cblog is the composition of Ceiling with Binary LOGarithm. Informally, cblog(k) calculates the base 2 logarithm of k, and if this number is not an integer, goes up to the nearest integer that is not less than it.

7.4 why is cblog(0) undefined?

cblog(0) is undefined because $\log_2 0$ is undefined because there is no number y such that $2^y = 0$.

7.5 what is unaryIO?

The unaryIO operator is defined in Lecture Notes 9 and builds on the definitions in Lecture Notes 8. The unaryIO operator is merely the precise mathematical statement of the conventions for unary input/output used for TMs from Lecture Notes 6 (which introduces TMs) onwards.

Here is a short summary of what cblog = $\text{unaryIO}_{1,1}(\text{M}_\text{g})$ implies:

- If $k \in \mathbb{N}$ and cblog(k) = $j \in \mathbb{N}$ and your Turing machine M_g is started on a tape t such that t = unaryInput(k) (i.e., k consecutive 1's are placed on the tape starting at position 0 and going to the right), then the computation sequence must halt with a final tape t' and a final head position i such that renumberTape(t',i) = unaryInput(j) (i.e., the final tape must contain only blank tape symbols except for j consecutive 1's starting at the final head position and going to the right).
- If $k \in \mathbb{N}$ and cblog(k) is not defined and your Turing machine M_g is started on a tape t such that t = unaryInput(k), then your Turing machine must signal that the result is undefined by doing one of these two things:
  - The computation may halt with a final tape t' such that t' $\neq$ unaryInput(k) for every $k \in \mathbb{N}$. An easy way to do this is to leave a symbol other than 1 or $\wedge$ on the tape. Other ways are leaving 1's to the left of the final head position and leaving more than one number on the tape (i.e., leaving a $\wedge$ symbol somewhere between two 1's).
  - The computation may diverge (not halt).

7.6 how big is your TM solution?

The lecturer's solution uses 7 active states (and 1 inactive state), 24 actions, and 5 symbols.

ADDED CLARIFICATION: There is no expectation that students will design a Turing machine that small.

7.7 are 10 000 000 steps enough to compute cblog(100)?

The lecturer's solution calculates cblog(100) in less than 10 000 steps, so limiting your TM to 10 000 000 steps is reasonable for computations of cblog(k) where $k \leq 100$. Your TM is allowed to be 1000 times less efficient than the lecturer's.

7.8 can I use symbols other than ˆ and 1?

You may use any of the 37 symbols allowed in this coursework during the middle of your computation but if your TM halts with a final tape that contains anything other than a single natural number starting at the final position of the head, the output will not count as a natural number and will instead count as the function being undefined for that input.

7.9 what should my TM do on an initial input tape that has symbols other than ˆ or 1 on it, or more than one number on it, or 1's to the left of the starting position?

By the rules of unaryIO, it does not matter what your TM does on input tapes that do not encode a single natural number in unary notation. For example, in the past, some students have had their TM's print a message on the tape when this happens. The assignment does not care.

7.10 can I include my name or ID or username or the course code or the date in my file names?

Variation in capitalization, spelling, or punctuation of any file name is FORBIDDEN. You must not include your name, computer account, email address, or student ID number in the name of any file you submit. The only allowed file names are:

- report.pdf
- turing-machine-action-table.txt

7.11 can I submit a .docx or .odt or .gif or .png or .svg file?

Files in the .docx or .odt or .gif or .png or .svg formats are FORBIDDEN.

The only allowed formats are plain text (.txt) and PDF (.pdf).

7.12 what file formats are allowed?

report.pdf The file report.pdf must be a PDF file.

turing-machine-action-table.txt The file turing-machine-action-table.txt must be a plain text file in the UTF-8 character encoding.

UTF-16 and UTF-32 (all variants) are FORBIDDEN and if your turing-machine-action-table.txt uses one of these encodings, it will be automatically considered to fail all tests.

7.13 can I use different item numbers or subitem letters or bullets in report.pdf?

No.

Item 1 must be numbered 1. Item 2 must be numbered 2. Item 1 must come before item 2. In other words, report.pdf must be nothing but a big numbered list from 1 to 5 (or possibly 6) with item 4 containing a lettered sublist from A to F.

7.14 how can I draw the graph in report.pdf?

The graph of your TM must be produced with some kind of graph-drawing software and must have properly typeset node and edge labels. Handwriting is FORBIDDEN.

Your lecturer sometimes draws TM graphs using dot (easier) and sometimes uses LaTeX with PSTricks (harder, but can make it look nicer). There are a lot of other options also.

ADDED CLARIFICATION: Drawing by hand is forbidden, except that it is okay if you draw by hand provided you use software that turns your drawing into perfect circles/ovals/etc. and straight lines and smooth curves.

ADDED CLARIFICATION: At a minimum, there must be node (state) and edge labels.

ADDED CLARIFICATION: It is a good idea if you pick descriptive state names. Your start state must be named q_0.

ADDED CLARIFICATION: You may (and it is a good idea) place explanatory labels identifying groups of nodes that act together in a certain way.

ADDED CLARIFICATION: It is okay to include an image file generated by your graph-drawing software.

7.15 what is a mathematical expression?

Here are examples of expressions:

- 1
- 7 + 9
- $\{2, 3\} \cup \{3, 4\}$
- f(3)

Expressions stand for mathematical objects/values.

Here are examples of statements (which are formally called “formulas”) which are NOT expressions:

- x = 1
- f(x) = x + 1 if $x \in \mathbb{N}$
- x = y or f(x) = 3

Statements assert/claim mathematical truths.

report.pdf item 3 is asking for an *expression*, not a statement or statements.

7.16 what goes in item 6 in report.pdf?

We’re hoping no one needs to use item 6. It’s there to prevent things that shouldn’t go in items 1 through 5 from going in those items. For example, if you feel helplessly compelled to include a poem in every coursework submission, item 6 is where it belongs.

7.17 what line ending characters can my turing-machine-action-table.txt file use?

Lines may be ended by LF (10) or by CR (13). A CR LF sequence counts as ending a line and then an empty line. Because empty lines are ignored in turing-machine-action-table.txt, it does not cause any problems to interpret CR LF this way.

7.18 what are the precise syntax rules for the turing-machine-action-table.txt file?

All space (32) and tab (9) characters are ignored.

A BNF-like grammar for the file format (assuming all spaces and tabs are already removed) is:

```
tm ::= ignored* begin line* end ignored*
ignored ::= newline | comment newline
newline ::= "[\n\r]"
comment ::= "#[^\n\r]*"
begin ::= "\{" newline
line ::= ignored | action
action ::= "\(\(" state "," symbol "\),"
            "\(" state "," symbol "," move "\)\)," newline
state ::= "[A-Za-z][-A-Za-z0-9_<>]"
symbol ::= "[A-Z0-9^]"
move ::= "[LR0]"
end ::= "\}" newline
```

7.19 ADDED QUESTION: what tape position does the Turing machine head start at?

In every Turing machine computation sequence, the head always starts at position 0 on the tape.

7.20 ADDED QUESTION: if the number 5 has been written on the tape, what tape positions does it occupy?

When using the unary input/output conventions of this course (which are specified as the unaryIO operator), if a single natural number 5 is written on the tape, then there are five digit one symbols on the tape at positions 0, 1, 2, 3, and 4, and all other tape positions contain a blank symbol.

7.21 ADDED QUESTION: if my TM passes the tests you provide do I need to write anything for item 4 subitem D?

You do not need to describe failures if there are none.

If there is even one number k such that your TM computes cblog(k) wrong, you must report this, regardless of whether the checker program I supplied detects this.

7.22 ADDED QUESTION: is it good enough if my TM passes the tests you provide?

The assignment is to implement cblog correctly, and to do this with reasonable efficiency for the numbers up to 100. The assignment is NOT just to pass a specific testing regimen.

The checker program that you have been supplied is only an aid. You MUST NOT rely on your TM only being checked in the way that program checks it.

7.23 ADDED QUESTION: "f29fb-2020,-21-cw-checker.py –comprehensive-test" takes 2 minutes. is this too long?

The 10 second time limit is per computation sequence. The command

```
f29fb-2020,-21-cw-checker.py --comprehensive-test
```

runs 135 computation sequences (as of the version of 2021-02-24). So 2 minutes is an average of 1.125 seconds per computation sequence. That is fine. The purpose of the time limit is to allow us not to have to wait a week to get the testing results for all 174 students before we start marking.

7.24 ADDED QUESTION: how can I get the tape positions to line up vertically when running f29fb-2020,-21-cw-checker.py?

I suggest you run the checker program like this:

```
f29fb-2020,-21-cw-checker.py --run-on-tape '^^^^^^@111^^^^^^'
```

The extra blanks on either side in the initial tape prevent the checker from reallocating memory for the tape in the middle of the computation, which will make your display look nicer because each tape position will always have the same horizontal position. Of course you should choose the number of blanks on each side of the initial input based on how much space your TM actually uses. The @ specifies the location that will become position 0 on the tape.

7.25 ADDED QUESTION: how should I write the configurations in the computation sequence in report item 2?

If you explain your notation for TM configurations properly, you may use any reasonable notation.

You may indicate the sequencing of the configurations (from first to last or from first to the point where you indicate it goes on forever) using any reasonable notation, provided you explain your notation.

Please avoid relying solely on color for indicating some part of the information, because your work might be marked by someone with some kind of color-blindness.