Middle East Technical University | Department of Computer Engineering

CENG 280

Formal Languages and Abstract Machines

Spring 2015-2016

Take Home Exam 3

Due date: May 27, 2015, Fri, 23:55

1 Objectives

To familiarize with computation utilizing Turing Machines and inspect aspects of language hierarchy

2 Specifications

You are expected to describe the operations taken by the indicated version of the 2D Turing Machine by roughly formalizing the seven-tuples that mathematically define the machine. Operation specifications are described in great detail in the text of the first question. Then, you should simulate that machine on a simpler version of Turing Machine having two-tapes with the traditional movement capabilities of the standard Turing Machine. Finally, you should utilize known proofs to deduce whether the 2D Turing Machine computationally does the same deed as the standard one.

Give low-level descriptions of Turing Machines to compute the given functions recursively in the second question. State clearly how recursion works on the final machine. You can combine the already-defined machines to finally obtain the last one. Then you are expected to name the overall mathematical operation conducted using these machines, i.e. identify the language of the machine.

Give higher-level procedural descriptions of a Turing Machine that decides the language in third question. Please pay particular attention to relate the execution of Turing Machine to the algorithm you devise. As a warm-up, first write-down the low-level details of some standard Turing Machine that checks whether the input strings to your algorithm obeys some specified general form or not.

State the nature of languages given in fourth question providing a formal proof that exploits given facts. If you are going to utilize well-known theorems, give clear references to them. However, when you are asked to proof something by constructing abstract machines, you should do it as if you had no access to well-known theorems governing the properties of general classes of languages. In other words, start from scratch in case of constructive proofs.

Please explicitly state any assumptions you make and pay particular attention to the notation you use. Your proofs should be sound and valid.

3 Questions

1 Consider a deterministic Turing Machine **M** which enables two-dimensional variable-length grid-like computation on data and its operations are described in the following specifications. (30 points)

- **M** has finite control over basically two distinct types of tapes having two dedicated heads. One of them is an auxiliary tape that holds positional information etc about the main 2D tape and its entities, and, as stated, its read/write head can move independently from the other. From now on, any reference to the word "head" will imply the head pointing to the 2D grid structure as the main tape.
- Initial (and then current) array size as number of rows and number of columns is stored in the auxiliary information tape which is semi-infinite to the right. This tape is not considered while computing the size of the array. Then, row-wise or column-wise number of valid entities and blanks together with their relative (or absolute) positions are appended on the right to this tape which may also be used for auxiliary computations if necessary.
- Initial dimensions of the array are to be provided in the information tape.
- Each conceptual row of the 2D tape constituting the grid structure should be considered to be infinite to both ends.
- In accordance with the initial number of rows and columns provided in the information tape, required number of entities are filled with a certain dummy initialization symbol other than the blank symbol included in the tape alphabet. Rows and columns are initially aligned as if the tape were bounded on the left hand side. At this instance, constructed array has the same number of rows and columns everywhere, which may be changed afterwards in accordance with the operator functionality to be described.
- The head initially points to the uppermost-left data entry of the 2D array just like where the constant pointer with the name of the array points to following the array initialization in C programming language.
- Following the initial "allocation" phase, left and right hand-sided untouched portions of the 2D tape should be conceptually thought as if they were filled with blanks. Therefore, when accessing a conceptually valid yet physically non-existing square on the tape, shifting and padding with blanks may have to take place.
- The head can be moved to point to any part of the 2D tape and take actions by means of the operations to be defined in the following.
  - *LA* (Left Allocate) : Move the head one square left of its current position in the 2D space. If an unallocated space is to be touched, then make room for it by writing the dummy initializer symbol on the proper portion of the tape. Contents of the information tape should be updated accordingly as well. Otherwise, the tape head just moves one square to its left.
  - *LB* (Left Blank): Obey the same specifications as *LA*, yet this time write a blank symbol instead of the dummy initializer if the accessed location is currently invalid.
  - *RA* (Right Allocate) : Move the head one square to its right. If this movement results in exceeding the boundary of row in question, then a new room is made for the corresponding square on which the dummy initialization character is written and information tape is updated. Otherwise, just an ordinary move to right is executed and action specified by the transition function is taken.

- *RB* (Right Blank) : Obey the specification in *RA*, writing a blank symbol instead of the dummy initializer in the edge case.
- *UA* (Up Allocate) : Move the head one square up of its current position. If the space is not already allocated either with the dummy symbol or with blank i.e. no read is possible, a new conceptual tape should be inserted on top the current data structure and the head is positioned to the square to be which is to be initialized with the dummy symbol. Namely, at the end of this operation the head can only read the dummy symbol with success in its subsequent transitions. Contents of the information tape should be updated accordingly.
- *UB* (Up Blank) : Perform exactly the same task as in *UA* except for enabling the head to read only a blank symbol in transitions to follow.
- *DA* (Down Allocate) : Obey the same specifications outlined in *UA* except that the head is made to point one square down of its current position.
- *DB* (Down Blank): Perform the same task in *DA* initializing the new room with a blank in the edge case.
- *T* (Transpose access) : Move the head to the array location specified topographically by exchanging the current row and column indices of the square the head currently points to. If the location is invalid, then make the head stay at its current position.

a) Formally represent **M** as a seven-tuple using C-like row-major array representation of the 2D-tape including all operations *LA*, *LB*, *RA*, *RB*, *UA*, *UB*, *DA*, *DB*, and *T*. In other words, you should unroll the conceptual 2D grid on "physical" 1D tape in your representation.

You should design your own formal **encoding** of the contents of the information tape and the main type by adopting dedicated markers to separate adjacent rows or to distinguish sections of information, the dummy allocation symbol and possibly other symbols included in input and tape alphabets as you please.

Formally define the **type** of the state transition function considering the two heads.

Verbally define and mathematically exemplify how information tape content is updated together with the changes in encoding of the main tape in response to each of the operations considering at least **one edge case** and **one occasional case** governing the operators.

b) Assume that you are given a deterministic semi-infinite two-tape TM **N** with two heads capable of moving only in the conventional L and R directions. Sketch a proof to simulate **M** using **N** by covering all cases in the corresponding transition functions with the freedom of avoiding very low-level tedious formalism. Do not use any auxiliary TM's.

c) Is **M** equivalent to the standard Turing Machine in terms of computational power? State your thoughts in the format of a formal proof with reference to well-known extensions to standard TM's, avoiding too much low-level detail.

2 Formally **construct** deterministic Turing Machines with possibly multi-tape extensions to compute the following functions. Assume that input parameters $x \in \mathbb{N}$, and $y \in \mathbb{N}^+$ are initially present in the tape(s) having unary representation of consecutive 1's (together with other tape symbols you would like). You may also represent 0 with a dedicated tape symbol. At the end, output(s) **should** reside on tape(s). You may re-use already defined TM's in boxes. However, be explicit about connecting transitions.
(30 points)

a) Given $x \leq y$ always holds, following function performs a round-off like computation

$$f(x,y) = \begin{cases} 0 & \text{if } x \leq y/2 \\ 1 & \text{otherwise} \end{cases}$$

b) The second function tries to divide the given integer by 5 and returns the result as an integer.

$$g(x) = \frac{x}{5} + f(x \mod 5, 5)$$

c) This final function performs the following recursive computation.

$$h(x) = \begin{cases} 0 & \text{if } g(x) = 0 \\ 1 & \text{if } g(x) = 1 \\ 1 + h(g(x)) & \text{if } g(x) > 1 \end{cases}$$

Verbally **identify** the language recognized by the TM that computes h.

3 Concerning the language included below, answer the questions. (20 points)

$$L = \{\, a^n b^i c^j \mid i = \log_3 n,\ j = n \times i,\ n \in \mathbb{N} \,\}$$

a) Formally **define** a deterministic one-tape Turing Machine $\mathbf{M}_f$ that accepts whenever an input string $w \in \Sigma^*$ is of the form $a^*b^*c^*$, otherwise it rejects. Restrict input alphabet to be $\Sigma = \{a, b, c\}$.

b) Give high-level (pseudo-code-alike) description of a deterministic Turing Machine $\mathbf{M}$ that constitutes an **algorithm** for L such that $L(\mathbf{M}) = L$. The TM $\mathbf{M}$ could be a multi-tape machine but you are **not** allowed to make use of other TM's that compute recursive functions, however you may freely refer to $\mathbf{M}_f$.
(*Hint*: $\log_3 27 = \log_3[3 \times 9] = \log_3 3 + \log_3 9 = 1 + \log_3[3 \times 3] = 1 + \log_3 3 + \log_3 3 = 1 + 1 + 1 = 3$)

4 Assume that following statements are true. (20 points)

- L_1 is recognized by a 2-stack PDA.
- L_2 is a finite language.
- There exits a DPDA that accepts all the strings in $L_3 = \overline{L_1} \setminus L_2$.
- Some branch of computation tree of a NTM accepts the strings in L_4.
- There exists a five-tape DTM that enumerates all strings in $L_5 = \Sigma^* \cap \overline{(L_3 L_4)}$.

a) Recall the language hierarchy and separately **determine** the **most specific** class of languages each of L_1, L_2, L_3, L_4, and L_5 belongs to. Write down your reasoning and explain your answer. Otherwise you will receive **no** credit.

b) Given that $L_6 = L_5^R \cup \overline{L_3}$, **determine** whether L_6 is **most specifically** a recursive or a recursively-enumerable language by constructing high-level schematic Turing Machines and forming connections in-between. If you think L_6 is an undecidable language, use proof-by-contradiction technique.

4 Submission

- **Late Submission:** You have 3 days in total for late submission of all homeworks. All homeworks will be graded as normal during this period. No further late submissions are accepted.
- You should submit your THE3 as a .tex file. You can use templates provided in previous exams with appropriate modifications.
- Soft-copies should be uploaded strictly by the deadline.

5 Regulations

1. **Cheating: We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations.
2. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.