York University
CSE 2001 – Unit 2 Models
Instructor: Jeff Edmonds

Read Jeff's notes. Read the book. Go to class. Ask lots of question. Study the slides. Work hard on solving these questions on your own. Talk to your friends about it. Talk to Jeff about it. Only after this should you read the posted solutions. Study the solutions. Understand the solutions. Memorize the solutions. The questions on the tests will be different. But the answers will be surprisingly close.

1. Loops: Do some computational problems require two loops or can all of them be accomplished by a single loop?

2. Suppose we want to design a TM to know whether a graph G is connected. What is the difference between G and $< G >$? When do we use $< G >$ and why is this done?

3. What is the difference between a computational problem that is a function vs one that is a language? If L is a language, what (if anything) do the following mean: $L(x)$, $x \in L$, $L = \emptyset$, $L = \{0,1\}^*$, L is decidable.

4. (Answer in Slides) TM Transitions: In the slides we proved that a TM that is only able to write the characters $\{0, 1, b\}$ on the tape is as powerful as one that has a larger tape alphabet Σ. The simpler machine simulated the more complex one by grouping its cells into blocks of some fixed length so that each block contained a 0/1 code for the required character from Σ. Suppose this block size is four, input character 0 is encoded with 0000 and 1 with 0001. The simulation must start by inserting three zeros in front of every bit of the input. For example, on input $b1011bb\ldots$, the output of this first phase should be $\ldots bb0001000000010001bb\ldots$. Note that we don't actually care if the the output is at the beginning of the tape as long as it is surrounded by blanks.

   (a) Writing down TM descriptions is hard. Instead, start by writing pseudo code using variables and loops. This is Jeff's level 4 abstraction of a TM. The only allowed actions are to read and write to the tape where the head is and to move the head to the left and right. I am also a big believer in *loop invariants*. This is a clear picture of what the tape looks like at the beginning of each iteration.
   Hint: Use the following Loop Invariant. For some $i \in [0, n]$, the first i bits of the input have been blanked out. These bits have been copied into a block past the input separated by a single blank with three zeros inserted before each. The head is on the b before

   (b) (Sorry: see the answer to the previous question before continuing.)
   Your next task is to prove that you know how to translate this code into a TM. You do not need to give the full translation. That would be far too tedious. Suppose a program had 13 lines of code and two variables $x, y \in \{1..5\}$, then what Pooh bear should write on the wall is the current line number that is being executed and the value of x and of y. Hence for each $k \in \{1..13\}, x', y' \in \{1..5\}$, let $q_{\langle line=k, x=x', y=y' \rangle}$ denote the state in which the TM is on line k, x has the value x', and y has the value y'. What are the states for the above program and how many of them are there? Give the general idea for how the transition function $\delta(q, c) = \langle q', c', direction \rangle$ is defined.

   (c) Consider line 2 of the code,
   i.e. "a = bit at head (i.e. the $i{+}1^{st}$ input bit), Write a blank, Move head right".

      i. This line of code is a level 3 abstraction of a TM, i.e. we interpret these blocks of transitions as lines of Java code. Explain what this code gets the TM to do.

      ii. Give a level 1 abstraction of this line, namely "Meaningful State Names". This involves giving each relevant entry of the table defining $\delta(q, c) = \langle q', c', direction \rangle$. The change from the level 0 abstraction is that we have given states meaningful names $q_{\langle line=39, x=3, y=0 \rangle}$.

iii. Give level 2 abstraction, namely "Meta Names and Transition Function". We talk about a block of states $q_{\langle line=39,x=x,y=y\rangle}$. for $x \in \{0,1,2,3,4\}$, and $y \in \{0,1,2,3,4\}$. We fill in blocks of the transition function table $\delta(q_{\langle line=39,x=x,y=y\rangle}, c) = \left\langle q_{\langle line=40,x=x,y=c\rangle}, c, right\right\rangle$.

(d) Do the same for lines 3 and 4, i.e. "Move head right until it is at a blank" and "Move head right one (i.e. into the output)".

(e) Do the same for line 9, i.e. "Write a".

(f) Bonus 10 marks if correct: The problem is to redo the previous problem, except the TM can only write 0 or 1. The tape starts with a two blanks before the input and an infinite number after it, but once a blank gets a 0/1 written in it, it will never be blank again. Hint: At first I was thinking that this could not be done, because the TM could not differentiate between whether what is written on the tape is the initial input or some markings to know that progress had been made. Then I saw how to do it. See if you can do it too.

5. (Answer in Slides) TM computing $J(I)$

   (a) The input consists two blocks each consisting of exactly a million 0/1 characters. The first is the binary description of a JAVA program J, the second of an input I. Describe, if possible, a simple TM for computing whether $J(I) = 1$, meaning that program J on this input halts and accepts or not, $J(I) = 0$. Focus on giving a meaningful name to each of the states of your machine. Describe the complete $\delta(q, c) = \langle q', c', direction\rangle$ function in all of its details.

   (b) Redo the question when J is arbitrarily long, but I is still of this fixed length. Remember that lots of problems can be solved that have arbitrarily long inputs so if you want to say that this one is uncomputable, don't give this as your reason.

6. (Answer in Slides) Recall that a Universal TM is a TM $M_{universal}$, which when given input $\langle$"M", $I\rangle$, simulates TM M on input I. You are to give a sketch of how such a $M_{universal}$ would be built.
   To make it easier, lets assume that M only has one tape and one head.
   Hint: Because $M_{universal}$ is one fixed TM, it must have one fixed set of states $Q_{universal}$ and one fixed tape alphabet $\Sigma_{universal}$ (i.e. the set of characters c that it is allowed to write in each tape call.)
   Hint: Fix $\Sigma_{universal} = \{\ '0',\ '1',\ '\langle\ ',\ ',',\ '\rangle'\ \}$.
   Hint: Let $M_{universal}$ have three tapes each with its own head.
   When $M_{universal}$'s computation begins, the description of M will appear on $M_{universal}$'s first tape and I on its second.

   (a) We must assume that the TM M described in $M_{universal}$'s input has an arbitrarily large set of states and an arbitrarily large tape alphabet Σ_M.
   Explain how you would describe M on $M_{universal}$'s first tape. Remember this description must be encoded using characters from $\Sigma_{universal} = \{\ '0',\ '1',\ '\langle\ ',\ ',',\ '\rangle'\ \}$.

   (b) At the beginning of $M_{universal}$'s simulation of the ${t_M}^{th}$ time step of M's computation, $M_{universal}$'s first tape will still contain the description of M. What would be best to put on $M_{universal}$'s second and third tape? Remember what you have written there must be encoded using characters from $\Sigma_{universal} = \{\ '0',\ '1',\ '\langle\ ',\ ',',\ '\rangle'\ \}$.
   Also where will $M_{universal}$ three tape heads be.

   (c) We say that M can compute in one time step any function of "what it knows", where "what it knows" is its current state and the character that its head currently on. More formally, if M is in state q and its head is seeing the character c, then $\delta_M(q, c) = \langle q', c', direction\rangle$ specifies M's next state and next actions.
   Is it possible that $M_{universal}$ "knows" q, or c, or for that matter δ_M?

   (d) Quickly sketch the actions taken by $M_{universal}$'s simulation of the ${t_M}^{th}$ time step of M's computation.

   (e) Let $T(|I|)$ denote the running time of M on I and let $|$"M"$|$ denote length of the description of M stored on $M_{universal}$'s first tape.
   What is the running time needed by $M_{universal}$ to simulate $M(I)$?

(f) It is not fair that $M_{universal}$ is allowed three tapes but M only one. Suppose instead, $M_{universal}$ was only allowed one tape.

Remember that a one tape TM $M_{universal}$ is able to simulate our three tape $M_{universal}$ by interweaving the cells of the three tapes on one tape or by having a larger tape alphabet $\Sigma_{universal}$ to include characters like $'0\langle 1'$ to signify that $'0'$ is on its first tape $'\langle '$ on its second and $'1'$ on its third.

The challenge is that the one tape version of $M_{universal}$ has one not three heads.

Show how the straight forward implementation requires $[\mathcal{O}(|\text{“}M\text{”}|) + \mathcal{O}(T(|I|))] \times T(|I|)$ time.

Show how this can be improved to only $\mathcal{O}(|\text{“}M\text{”}|) \times T(|I|)$ time with the smart trick of moving the “M” and the “q”.

7. (Solution in Slides)

Think carefully about the definitions of Turing Machines. For every state q that the machine might currently be in and character c that might be written in the cell of the tape pointed to by the head, we define $\delta(q, c) = \langle q', c', direction\rangle$ where q' is the next state, c' is the character to write, and *direction* is the direction in which to move the head one cell.

For each of these three specified changes to the TM model, explain the effect it would have on the reasonableness and on the computational power of the model. Do one of:
- A: Either argue that the set of problems computable does not change because the new and the old versions of TM can simulate each other
- B: or show that the new model is weaker by giving a computable problem that no longer can be computed
- C: or show that the new model is stronger by giving an easy algorithm for an arbitrary uncomputable problem (for example the halting problem).

(a) Instead of requiring the characters c that are written in each cell of the tape to be from a set of a fixed size (eg binary or ascii), allow the c and the c' in the above definition $\delta(q, c) = \langle q', c', direction\rangle$ to be arbitrarily large integers.

(b) Change the model so that the TM “knows” the location of the head. More formally, $\delta(q, i, c) = \langle q', c', direction\rangle$ also depends on the index i of the cell that head is currently at.

(c) Change the model so that the TM no longer has a head but can specify the index of a cell of the tape to read and to write to. The TM still has its input stored in the first n cells of the input and is only allowed a fixed/constant/finite number of states and alphabet size. Formally, $\delta(q, c) = \langle q', c', i, i'\rangle$ where c is the last value read, c' will be written to cell indexed by i and the next c will be read from that indexed by i'.

8. Eric Ruppert had a contest to make as a TM with only 6 states that starting with string 11111100000 runs for as long as possible. I gave him a solution with only 5 states that ran for 3,983,884 time steps.

```
My idea is think of the contents of the tape in binary and to count
down and to extend the length of the used tape by one each decrement.
However, we flip the role of zeros and ones.  And we flip the order of
the bits from left to right.

Our initial tape is to contain #11111100000.  (Here # indicates the
start of the tape)
Changing zeros to ones and ones to zeros
and flipping the order of the bits from left to right gives
 11111000000#, which is 1984 in binary.
Each meta step will decrement this number by one and add a leading
zero.
```

```
The loop invariant is that after the i^th meta step, the number in
binary will be 1984-i with i leading zeros.
For example,
i=0: _________11111000000#  = 1984  with 0 leading zeroes
i=1: ________011110111111#  = 1983  with 1 leading zeroes
i=2: _______0011110111110#  = 1982  with 2 leading zeroes
i=3: ______00011110111101#  = 1981  with 3 leading zeroes
i=4: _____000111110111100#  = 1980  with 4 leading zeroes
i=5: ____0000111110111011#  = 1979  with 5 leading zeroes

Changing ones back to zeros and zeros back to ones
and flipping the order of the bits back again from right to left then
gives
the actual tape content to be
i=0: #11111100000_________
i=1: #000000100001________
i=2: #1000001000011_______
i=3: #10000010000111______
i=4: #110000100001111_____
i=5: #0010001000011111____

Let n = 1984 be our initial value.
Let m = 11 be the initial number of bits.
The i^th meta step   (going from i-1 to i)
will change the integer from n-i+1 to n-i
and change the number of bits from m+i-1 to m+i.
To do this, the algorithm must move the from left to the right and
back to the left.
This takes 2(m+i) time.
The last pass of the TM will just go once till it finds the blank.
Hence, the total time is
  [sum_{i = 1..n}  2(m+i)] +  m+n
  = 2mn + [sum_{i = 1..n}  2i] + m+n
  = 2mn + n(n+1) + m+n
  = 2(11)(1984) + (1984)(1985) + 11 + 1984
  = 3,983,883

To make it easier to understand,
I am first going to write the code with zeros and ones switched
and left and right flipped.

Suppose the current state is
  i=4: _____000111110111100#  = 1980  with 4 leading zeroes
  with the head on the right most bit, i.e., the low order bit  (here a 0)
  in state s0
and we must change it to
  i=5: ____0000111110111011#  = 1979  with 5 leading zeroes
  with the head on the right most bit, i.e., the low order bit  (here a 1)
  in state s0

Psuedo Code:
Move left changing zeros to ones until you reach the rightmost 1.
Change this one to zero and go one more step left.
```

The algorithm then moves the head all the way to the left to the first blank and changes it to a 0.
The algorithm then moves the head back all the way from left to right. This reestablishes the loop invariant.
The code terminates when the integer on the tape is all zero.

Translate this into TM transitions.