COT 4210: Discrete Structures II
Exam #2 Solutions
March 21, 2013

Lecturer: Arup Guha

(Directions: Please justify your answer to each question. No answer, even if it is correct, will be given full credit without the proper justification.)

1) (15 pts) Consider the following formal definition of a Turing Machine M:

$$\Sigma = \{0,1,\#\}, \Gamma = \{0,1,\#,\$,B\}, start\ state = q_0$$

$\delta(q_0, \#) = (q_1, \#, R)$
$\delta(q_1, 0) = (q_2, \$, R)$
$\delta(q_1, 1) = (q_{12}, \$, R)$
$\delta(q_1, \#) = (q_{accept}, \#, R)$
$\delta(q_2, 0) = (q_2, 0, R)$
$\delta(q_2, 1) = (q_2, 1, R)$
$\delta(q_2, \#) = (q_3, \#, R)$
$\delta(q_3, \$) = (q_3, \$, R)$
$\delta(q_3, 1) = (q_4, \$, L)$
$\delta(q_3, 0) = (q_{reject}, \$, L)$
$\delta(q_4, \$) = (q_4, \$, L)$
$\delta(q_4, \#) = (q_5, \#, L)$
$\delta(q_5, 0) = (q_5, 0, L)$
$\delta(q_5, 1) = (q_5, 1, L)$
$\delta(q_5, \$) = (q_1, \$, R)$
$\delta(q_5, \#) = (q_{accept}, \#, R)$
$\delta(q_{12}, 0) = (q_{12}, 0, R)$
$\delta(q_{12}, 1) = (q_{12}, 1, R)$
$\delta(q_{12}, \#) = (q_{13}, \#, R)$
$\delta(q_{13}, \$) = (q_{13}, \$, R)$
$\delta(q_{13}, 0) = (q_4, \$, L)$
$\delta(q_{13}, 1) = (q_{reject}, \$, L)$

(a) Show the steps this Turing Machine executes with the input #011#100. Use the notation from the textbook.

1. q_0#011#100
2. #$q_1$011#100
3. #$\$q_2$11#100
4. #1q_2$1#100
5. #11q_2$#100
6. #$11#$q_3$100
7. #11q_4$#$00
8. #1q_5$1#$00
9. #$\$q_5$11#$00
10. #$q_5$$11#$00
11. #$\$q_1$11#$00
12. #$\$\$q_{12}$1#$00
13. #$\$1q_{12}#$00
14. #$\$1#$q_{13}$$00
15. #$\$1#$\$q_{13}$00
16. #$\$1#q_4\$\$0
17. #$\$1q_4#$\$0
18. #$\$\$q_5$1#$\$0
19. #$\$q_5$\$1#$\$0
20. #$\$q_1$1#$\$0
21. #$\$\$q_{12}$#$\$0
22. #$\$\$#$q_{13}$$\$0
23. #$\$\$#$\$q_{13}$\$0
24. #$\$\$#$\$\$q_{13}$0
25. #$\$\$#$\$q_4$\$\$
26. #$\$\$#$q_4$$\$\$
27. #$\$\$\$q_4$#$\$\$
28. #$\$\$q_5$\$#$\$\$
29. #$\$\$\$q_1$#$\$\$
30. #$\$\$#$q_{accept}$$\$\$
(6 pts just eyeball)

(b) Give a brief description in English of the set of strings this machine accepts.

It accepts any string that starts with a # (1 pt), followed by some 0s and 1s (1 pt), followed by another # (1 pt), followed by a string whose prefix is the complement (swap 0s for 1s and 1s for 0s) of the first string of 0s and 1s (2 pts). The machine never looks past this portion of the input, so what follows is irrelevant. Thus, strings like #0110#1001 and #0110#100110101###10 are accepted. (Note: I have placed a no extra characters in the first example and several in the second.)

(c) What is similar about the states q_2 and q_{12}? What is the key difference between the two states.

Both states signify that we are currently moving from left to right and that we've marked off a character on the left side of the string. (2 pts)

The difference is that q_2 indicates that the marked character was a 0 while q_{12} indicates that the marked character was a 1. (2 pts)

2) (10 pts) Consider a Turing Machine model that is the same as a standard machine, but is never allowed to move in the same direction twice. How many different languages over the input alphabet {0,1} can be specified using this model? Justify your answer.

This machine can only ever read the first two input characters. Thus, it can only act differently between 7 categories of strings: ε, 0, 1, 00$\sum^*$, 01$\sum^*$, 10$\sum^*$, and 11$\sum^*$. (5 pts)

Essentially, whatever the machine would do on the string 00, it will behave in the same exact way for 001, 000101 or any other string that starts with 0. For these 7 categories of strings, the machine may either accept or not accept.

Thus, every language that can be described by such a machine must be a subset of the seven items listed above. (2 pts)

There are exactly $2^7 = 128$ such subsets. (3 pts)

(Note: Based on further reasoning, there might be a small chance that one could reduce this bound. I will accept a lower answer with a further justification that's valid.)

3) (12 pts) Let the language L be as follows: L = { <G, k> | G is an undirected, connected graph that can be colored with k or fewer colors. } Note: Remember that a valid coloring of a graph is one where each vertex is assigned a color such that no two vertices connected by an edge are assigned the same color. Prove that L is decidable.

We describe an algorithm to solve the given problem:

Let n = the number of vertices in the graph.

For each of the n vertices, try assigning each possible combination of k colors. There are k^n such combinations. (6 pts)

For each color assignment, check to see if it's valid. If any of the assignments are valid, accept. If not, reject. (4 pts)

This terminates because the number of combinations to check is not infinite and a finite amount of work is done for each combination. It produces the correct answer because each possible color combination is attempted. (2 pts)

4) (15 pts) Consider the following enumerator of the positive composite integers greater than 1:

1) List all multiples of 2.
2) List all multiples of 3.
3) etc.

What is the flaw?

Use this same principle (listing multiples of different primes) to describe how to create a valid enumerator for the positive composite integers greater than 1. Remember that an enumerator may print the same value more than once, but it must guarantee to eventually print out any particular item in the language.

The flaw is that we'd never finish listing all possible multiples of 2, thus we would never get ot print 3 (and any other odd values). (5 pts)

In order to fix this flaw, we must not print all the multiples of 2 before proceeding to some multiples of 3, 5, etc. One technique would be to have an outer loop that provides a fixed value at which to stop printing multiples. In the inner loop, we go through each unique prime's multiples. (4 pts) In pseudocode, we have the following:

```
limit = 2

while (true) {
    int div = 2
    list used = new list()

    while (div < limit) {

        if (!used.contains(div)) {

            for (int i=div; i<=limit; i+=div) {
                print(i)
                used.add(i)
            }
        }
        div = div + 1
    }
    limit = limit + 1
} (6 pts)
```

The key here is that we never try all multiples of 2. We only try them up to some limit before trying multiples of other primes. In this manner, we can see that limit can become arbitrarily high and any chosen positive integer (greater than 1), will eventually be printed.

5) (15 pts) Let L = { $<M_1, M_2>$ | M_1 and M_2 are TMs such that $L(M_1) \cap L(M_2) \neq \emptyset$ }. Prove that L is undecidable.

Assume to the contrary, that L is decidable. (3 pts) Let R be the Turing Machine that decides membership in L. (2 pts) We will w rite a decider, S, for A_{TM} as follows (2 pts):

Turing Machine S(Marchine M, String w) {

Create M' such that M' accepts w and no other string.
return R(M, M')
} (5 pts)

If M and M' accept a string in common, it MUST BE w, since this is the only string accepted by M'. Thus, when we find out that the intersection of L(M) and L(M') is non-empty, we can confidently conclude that M accepts w, as desired. On the other hand, if M doesn't accept w, then this intersection will indeed be empty. (3 pts)

6) (15 pts) Let SS_{TM} = { $<M_1, M_2>$ | M_1 and M_2 are Turing Machines with $L(M_1) \subseteq L(M_2)$. }. Show that SS_{TM} is mapping reducible to EQ_{TM}. Namely, Given an input $<M_1, M_2>$ describe an algorithm to compute an ordered pair $<M_1', M_2'>$ such that if and only if $<M_1, M_2> \in SS_{TM}$, $<M_1', M_2'> \in EQ_{TM}$.

Let $M_1' = M_1$. (4 pts)

Create multitape TM M_2' as follows:

On its first work tape, it copies the input and runs the directions of M_1 on it. If the q_{accept} state in M_1 is reached, the machine M_2' will transition to a state that will begin copying the original input onto the second work tape. From there, the machine will start running the steps of M_2 on this second work tape. (8 pts)

We have constructed M_2' such that it only accepts strings accepted by both M_1 and M_2. Thus, $L(M_2') = L(M_1) \cap L(M_2)$. (2 pts)

If $L(M_1)$ equals $L(M_2')$, then we know that $L(M_1) \subseteq L(M_2)$. Alternatively, if these two sets are NOT equal, there must be some string w such that $w \in L(M_1)$ and $w \notin L(M_2)$. Equivalently, this means that $L(M_1)$ is NOT a subset of $L(M_2)$, as desired. This completes our mapping and proves that

if and only if $<M_1, M_2> \in L(M_1)$, then $<M_1', M_2'> \in L(M_2)$. (1 pt)

7) (15 pts) Let L = { <M, s> | M is a TM with state s such that, M never enters state s on any possible input.} Is L decidable? Prove your answer.

L is not decidable. (3 pts)

Assume to the contrary, that L is decidable. In this case, we must have some Turing Machine R that decides membership in L. (3 pts)

We will now use R to build a decider, S, for E_{TM}: (2 pts)

Turing Machine S(Machine M) {
return R(M, q_{accept})
} (5 pts)

Any TM that doesn't accept any strings will never reach the accept state. But, this is precisely what R tells us. So, if R tells us that M never reaches the accept state, we have proof that M can't accept any strings and L(M) = {}. Once we know this, we can determine whether or not M is an element of E_{TM}. But, E_{TM} is undecidable, which provides our contradiction. It follows that our original assumption, that L is decidable is false. Thus, L is undecidable. (2 pts)

8) (3 pts) Which company runs the website maps.google.com? **Google**