Name:
Email: *@princeton.edu*

Midterm

COS 116 Spring 2010: The Computational Universe

This is an in-class exam. You may refer to the pseudo-code handout and your own page of notes. No collaboration is allowed. Write and sign the honor pledge:

"I pledge my honor that I have not violated the Honor Code during this examination."

---

1) Even a decade ago we could make cheap robots like Scribbler, and make a chess-playing program that beat the world's best human chess player. Briefly, why is it that today we do not all own personal robotic butlers like C3PO in *Star Wars*?

(8pts) Many possible answers, for example:

We have not yet solved the problem of natural language recognition and understanding (automatic voice menu systems don't even work!)
-- or --
Gross motor motions are relatively easy but the kind of tactile and control feedback even to get fingers on a hand to grasp simple objects is still challenging.

2) Which of these is possible? Provide a short explanation with each answer:

   (a) An iphone app that will tell you "yes" if a particular starting configuration in Conway's Game of Life will lead to a critter living in cell (100,100) after one million generations/steps, and "no" otherwise.

(5pts each = 20 total)

Yes, simply run a simulation on the iphone (which is a general purpose computer) for 100 million steps and check the cell.

   (b) A Turing Machine (using the notation in the Davis article) that answers the same question as in part (a).

Yes, Turing Machines can simulate iphones.

(c) Princeton University decides to store the text of every publicly linked Web page active today and receives a $1 million grant to do so. Can it do so for that amount of money?

Yes, in class we estimated the storage cost as $100,000.

(d) Can we get a snapshot of the Web, as described in (c), as is at exactly noon EST Jan 1, 2011?

No, because it takes time to crawl the web, which is always evolving.

3) The professor's MacBook Pro with 2.53GHz Intel Core 2 Duo Processor, 2GB Memory and 160GB hard drive can simulate a Turing Machine. (We saw this demonstrated in lecture.)

(a) Can a Turing Machine simulate the MacBook Pro?

(3 pts) Yes, it can simulate any specific computer implementation.

(b) Which of these two machines, if either, is more powerful, and why?

(5 pts) The TM has infinite memory and is therefore more powerful.
It can solve problems the MacBook cannot solve due to finite memory.

4) How can the DNA of a mouse describe everything about the mouse? A mouse has many cells; and every cell in contains its DNA; and the DNA needs to describe every cell. Explain briefly how this can be.

(8pts) The DNA can create a program ("instructions") for creating the mouse, and therefore is more concise than the mouse itself.

5) The OR gate described in class will answer TRUE if either *or both* of its inputs is is TRUE. A different gate that we did not discuss is the XOR (for "exclusive or") gate, which will answer TRUE only if exactly one of its two inputs is TRUE.

(a) Write the truth tables for an OR gate and for an XOR gate. Let's name the two inputs A and B, the output of OR is O and the output of XOR is X.

(8pts)

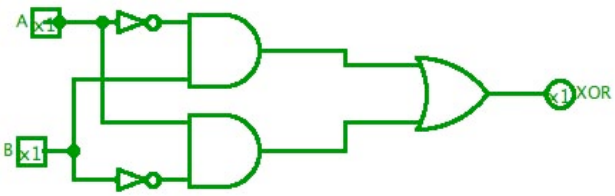
A	B	OR	XOR
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

(b) Write a formula that expresses (with AND, OR and NOT) the logic of XOR.

(8 pts) X = ((NOT A) AND B) OR ((NOT B) AND A)

(c) Draw a circuit that implements XOR (using AND, OR and NOT gates).

(8pts)



6) In the Stewart article, ants follow trajectories determined by black and white squares. Are there trajectories wherein an ant repeatedly and infinitely visits the same square? What if any are the implications?

(8 pts) No. Cohen-Kong Theorem implies that no bounded trajectories exist. If the ant were to visit the same square infinitely often, it would imply that it does not always build a 'highway' (thus resolving the question posed by Nathan in Stewart's article).

7) In the Gale-Shapley algorithm for finding stable matchings, one of the lines is:
F = first woman on M's list to whom he has not yet proposed

Let's be more explicit by writing pseudo-code to implement that idea. Assume you already have an array called W (of length L) that contains all women in order of decreasing preference by M. That is, M's first choice is W[1], second choice is W[2], and so on. Moreover, assume you also have another array P (of length N where N<L) that contains the list of all women that M has already proposed to.

(a) Write the psuedocode here:

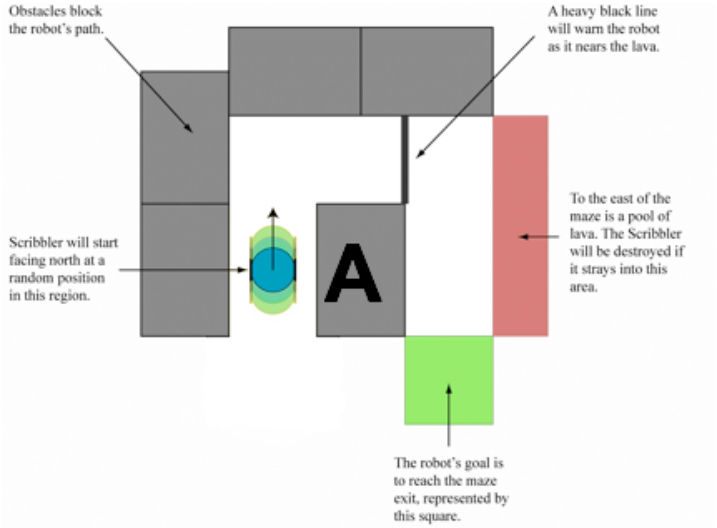
(10 pts)

```
i = 0
foundWoman = 0
while ( foundWoman = 0 and i < L)
{
        i = i + 1
        inList = 0
        for (j = 1 to N)
        {
                if  (P[j] = W[i]) then inList = 1
        }
        if (inList = 0)   // he has not proposed to W[j]
        {
                F = W[i]
                foundWoman = 1
        }
}
```

Note:
Output is F, the first woman on M's preference list to whom he has not proposed.

(b) What is the worst-case situation in terms of running time for this operation? Express the running time in terms of number of iterations, as a function of N and L?

(6 pts) Every time he considers a woman on his preference list P (length N) he has to examine the list W (length L) of women to whom he has already proposed.
So the worst case running time is proportional to NxL.

8) 

In Lab, the Maze was constructed so that when Scribbler detected the obstacle, it was too early to turn right because the passage was still a few inches ahead. Many people proposed this solution:

Move forward until obstacle is detected.
Move forward a few inches.
Turn right.

How effective is this solution and why? Suppose Scribbler ends up in another Maze (with the same geometric structure, but with different distances between walls). Would this solution work there? You may assume that only Wall A moves up or down and that all walls have the same reflectance properties.

(8pts) The best strategy is to move as close as possible to the wall, turn right and then go. This solution is effective as long as the wall is always detected at the same distance from the wall (so that a robot could travel that distance to get near the wall). As long as wall reflectance properties do not change this distance would not change. Thus the suggested solution works as long as 'a few inches' would get robot right next to the front wall.

9) Feedback:
(a) What is your favorite part(s) of this class, so far?

(0 pts) No right or wrong answer.

(b) What aspects(s) would you change?

(0 pts) No right or wrong answer.