FINAL STUDYGUIDE – Winter 08

WHAT YOU CAN BRING
Anything. Notes, book, past homework. However, no computers, sorry.

TOPICS COVERED
The entire quarter. However, will concentrate on material since the midterm. Make sure you can answer questions about talks by Matt Sotille, Victor, and Heiner Linke. See Q1 below for a guide. For chapter 8, concentrate on sections *Preparing the Defenses*, *Development of Defensive Technologies and the Impact of Regulation*, and *A Program for GNR Defense*. For chapter 9, I'll have to say that RK sets up critiques that he can easily shoot down (called by some "setting up strawmen"). Critique that I see missing: "RK, you take reported work and hype it up beyond recognition." Professor Linke gave a good example. And RK responds how? Would like to see this question asked in a debate! Anyway, for chapter 9 I would like you to concentrate on this section: *criticism from ontology*.
Please view the remainder of the RK and DG debate. I'll likely ask a question about the end of the debate that is similar in form to debate-questions I asked on homework earlier in quarter.

WHAT YOU SHOULD STUDY
There will be a 60/40 split on the "theory" side and the "application" side. Remember that you will not have a computer to check your answers so make sure you can solve problems using paper and pencil. I have attached some sample questions below. I did not include many given most will have the same flavor as midterm. I did include a question related to outside speakers and a couple related to Ruby.

WHAT YOU SHOULD BRING
You will write your answers on your own notebook paper. So bring blank paper. If I were you, I would write my name at the top of every page *before* starting. We will supply a stapler.

FINAL ADVICE
Same as for midterm: don't count on studying during the final!

Practice Questions

Q1. Give a synopsis of Professor Farley's guest lecture as it relates to our class. Use structure as below. (Note I will give you outline answer of what I would expect for this question.)

Topic: AI-based problem solving. He discussed search algorithms that looked for solutions. He went over both hill-climbing search and genetic algorithms. Both take a current state, produce a proposed new state, and then call a fitness function to see if the proposed new state is better, i.e., closer to a solution.
Example: used the design of a new NASA antenna for genetic algorithms. Reported that a genetic algorithm designed a better antenna than human experts in an open competition. He noted that the genetic algorithm had no deep antenna-design knowledge, just a means to search the design space. It did have a well defined fitness function that could be automatically applied. Also mentioned 8-queens problem.
Did PF support our textbook? Yes in that there are existence proofs of computers designing better hardware through search. One can imagine better software also being built by computers through search-based approaches. This, then, could be start of a fast evolution of intelligent machines.
Did PF disagree with our textbook? Yes in terms of practical issues. In particular, the search algorithms he discussed require a fitness function to determine if new generations are better than old. For the antenna example, the fitness function was automated. However, in more complex domains, it appears that a human judge must enact the fitness function. In this case, you lose all the power of fast search. How long will it take a human to go through millions of proposed designs and grade them? In particular, if we are trying to

evolve an intelligent robot through genetic algorithms, who will judge if one version is more intelligent than another? Would seem we would need an intelligent judge. Kind of a circular problem.
Other discussion in class: are humans at the end of the evolutionary line? Are our brains still evolving? Would we predict that we will have super-intelligent descendants sometime in future, through evolution alone (no computer enhancements)? Or are machines ready to take over and move ahead?

Q2. Assume the assignment of a string to cw as below:

```
cw = '<person>
        <address>[…]</address>
        <age>22</age>
        <name>John Doe</name>
    </person>'
```

The […] represents lots of characters that I am not going to list. You can just assume that it elides from 0 to 100 characters.

2.a Give Ruby code that will return what is in the age tag. I know you can see it is 22, but assume that it can be any integer from 1-100. Ruby can't "see it" by staring at a page, so you need to pull it out from cw. Store it in the variable age.

2.b Give Ruby code that will print "ok" if the age is greater than 20 and "not ok" if it is 20 or less. Again, assume you cannot know ahead of time what is in the age variable; you will have to write code that checks it.

Q3. Here is a new algorithm I found while rummaging around on the web. It is a means of "simplifying" a word. It's not as drastic as the hashWord algorithm, but in the same spirit.

Step 0. Make the word lowercase.
Step 1. If the word has any double l's (i.e., "ll"), double o's (i.e., "oo") or double p's (i.e., "pp"), squeeze each one down to just a single l, o, or p, respectively. I'll assume that there are no words that contain triple letters or above.
Step 2. If the word ends in "es", remove the es. (Remove plural ending.)
Step 3. If the word ends in "s", remove the s. (Remove plural ending.)
Step 4. Print the word that remains after above steps.

Some Ruby code that might help: "hello"[k,2] will return the 2-character substring starting at location k, i.e., will take a slice out of the word. "hello".slice(k,2) will do the same.

Here are some examples:

Before: Hellos	Printed: helo	
Before: goes	Printed: go	
Before: pooppers	Printed: poper	
Before: possesses	Printed: posses	(Not so good: changes possesses into plural of posse!)

Implement the algorithm in Ruby. Assume that the variable w contains the initial word.