CSCI-561 University of Southern California
Homework 4 - Due Date 12/02/2003

Question 1 (20 points)

Consider the following six well-formed formulas (wff):

wff	clause form
1. $\forall$ x, CS-class(x) => likes(John, x)	~CS-class(x) $\vee$ likes(John, x)
2. CS-Class(AI)	CS-Class(AI)
3. takes(Susan, OS) $\wedge$ ~fail(Susan, OS)	~takes(Susan, OS) , ~fails(Susan, OS)
4. $\forall$ z takes(Susan, z) => takes(Larry, z)	~takes(Susan, z) $\vee$ takes(Larry, z)
5. $\forall$ x,y takes(x,y) $\wedge$ ~fail(x,y) => CS-Class(y)	~takes(x, y) $\vee$ fails(x,y) $\vee$ CS-Class(y)

(a) Convert these 6 wff to clause form

(b) Use the resolution refutation algorithm to prove the following:

likes(John, OS) $\wedge$ likes(John, AI)

With each resolution step, also write down the substitution made during unification, if any.

sol)

add ~likes(John, OS) $\vee$ ~likes(John, AI) to the knowledgebase -- 6
Using 1, 6 , infer ~CS-class(OS) $\vee$ ~likes(John, AI) ({OS|x}) -- 7
Using 1, 7, infer ~CS-class(OS) $\vee$ ~CS-class(AI) {(AI|x}) -- 8
Using 2, 8, infer ~CS-class(OS) -- 9
Using 5, 9, infer ~takes(x, OS) $\vee$ fails(x, OS) ({OS | y}) -- 10
Using 3, 10, infer ~takes(Susan, OS) ({Susan|x}) --11
Using 3, 11, infer NIL

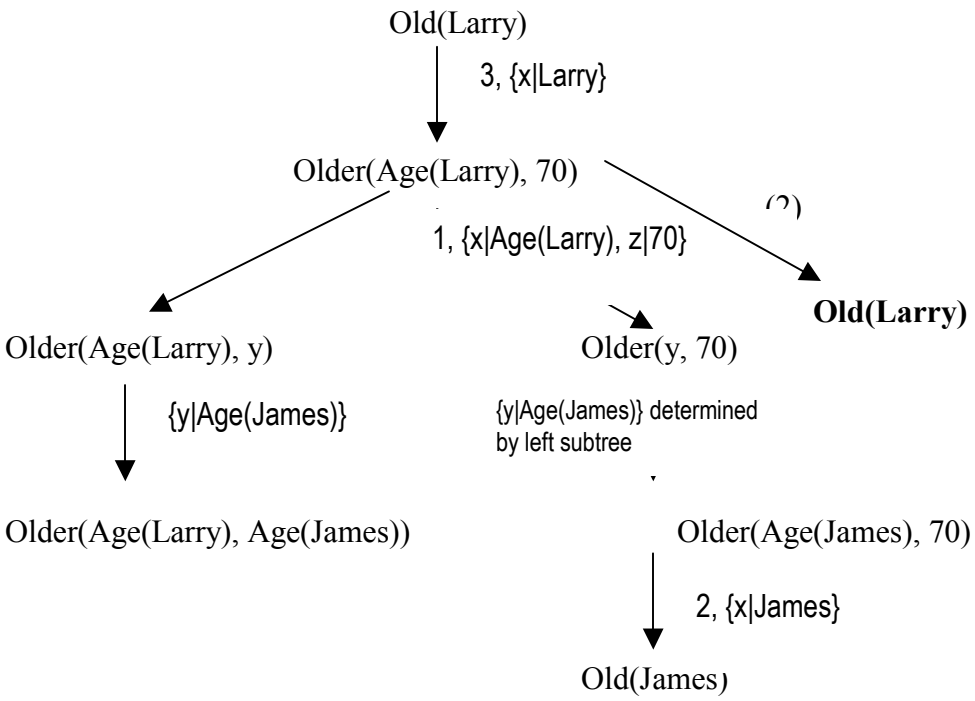
Question 2 (10pt)

Consider following five sentences.

1. $\forall$ x,y,z Older(x,y) $\wedge$ Older (y,z) => Older(x,z)
2. $\forall$ x, Old(x) => Older(Age(x), 70)
3. $\forall$ x, Older(Age(x), 70) => Old(x)
4. Old(James)
5. Older(Age(Larry), Age(James))

Prove Old(Larry) by drawing a **Backward-chaining** inference tree. Precisely specify the rule number from above for each rule you use, or which of the other nodes in your tree you use. Also specify the substitutions used for unification at each step.

Sol)



Question 3 (10pt)

Consider the problem of planning a route from one city to another using situation calculus. The basic action of the agent is Go(x, y), which takes it from city x to city y provided there is a direct route between the cities. DirectRoute(x, y) is true if and only if there is a direct route from x to y; you can assume that all such facts are already in the knowledge base. The agent begins in LA and must reach Sandiego.

(a) Write a suitable logical description of the initial situation of the agent, given the situation calculus framework.

At(agent, LA, S0)
At(LA,S0)

(b) Write a suitable logical query whose answers will provide possible paths to the goal.

∃s At(agent, Sandiego, s)
Ep At(SanDiego, PlanResult(p, S0))

(c) Write a sentence describing the Go action using a successor-state axiom.

$$\forall a,x,y,s \;\; At(Agent, y, Result(a, s)) \Leftrightarrow [(a = Go(x,y) \wedge DirectRoute(x,y) \wedge At(Agent,x,s)) \vee (At(Agent,y,s) \wedge \neg(\exists z \;\; a=Go(y,z) \wedge z \neq y))]$$

A a,x,y,s Go(x,y,Result(a,s)) <=> {(a=At(x,s)^DirectRoute(x,y)}

Question 4 (20pt)

(a) Tired of having tons of rotten food in your refrigerator at home, you have decided to design a new breed of refrigerator controller that can take into account the freshness of the food inside the refrigerator, and adapt the behavior of the refrigerator depending on how fresh the food inside is. Looking at your CSCI561 slides, you decide that this is a perfect challenge for the fuzzy logic formalism. You want to automate the operation of the refrigerator based on the following 3 rules. Activate (the possibility of turning on the compressor of your refrigerator, such as to cool down the contents of the refrigerator) is evaluated based on two criteria: **temperature** inside the refrigerator, and **freshness** of the food inside the refrigerator. The temperature ranges from 0 to 70 degrees F and freshness ranges from 0 (rotten) to 10 (extra-fresh). Activate ranges from 0 to 10, indicating how strongly you should activate the refrigerator's compressor in an attempt to cool down the contents of the refrigerator. The following are the decision rules:

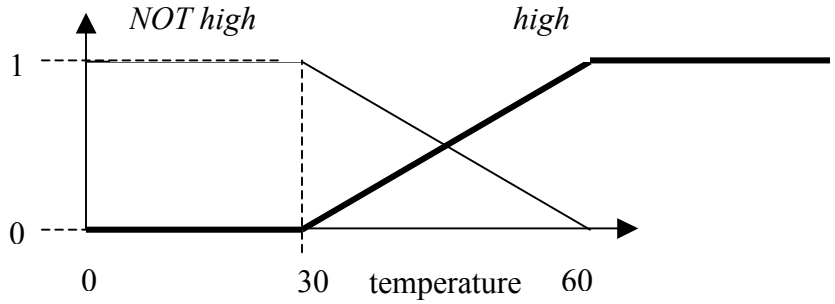
1. If the temperature is low and the freshness is high, then Activate is low
2. If temperature is high then Activate is medium
3. If the freshness is not high and temperature is very high then Activate is high

(i) Write the corresponding fuzzy IF-THEN rules. Use BLOCK (Capital) letters for fuzzy operators and underline the fuzzy terms.

1. IF temperature == low AND freshness = high THEN Activate = low
2. IF temperature == high THEN Activate = medium
3. IF freshness == NOT high AND temperature == very high THEN Activate = high

(ii) For the variable *temperature* define the fuzzy sets *high* and *not high.* Draw fuzzy membership functions that you think will work well for your problem in a diagram. Define the hedge *very* and draw the fuzzy term *very high.* Clearly label each set on the diagram. Label and indicate the range of values for each of the horizontal and vertical axes on the diagram.

These fuzzy sets can have many various shapes BUT must make sense, i.e., consistently define the terms high and *not high*. *NOT high* should be the complement of *high*, e.g., *1 –high*.



Very High: $F=F^2$

(iii) Using "clipping" fuzzy inference as studied in class, draw the full inference diagram that would be used to compute the system's output for **temperature=40** and **freshness=2**. Hint: show how the temperature and freshness values are used by each of our 3 fuzzy rules, drawing diagrams similar to that of question (ii). Defuzzify the output (do not worry if your drawings are not perfectly precise, as long as you correctly label their key points, e.g., label the axes at the location of the maximum of a given curve, etc) and write down the amount of activation predicted by your fuzzy reasoning system.

Question 5 (40 pt)

This question is about RDF and has three parts, with the goal of familiarizing you with RDF and with ontology design and use in the RDF framework:

- Design an RDF ontology.
- Add to it instances based on your ontology.
- Construct query files to do Jena command-line queries.

(a) Design an RDF ontology.

Build an ontology of your choice. It should have at least the complexity of the university ontology shown in the sample university-pure-rdf.rdf available in the homeworks section of our class web site.

(b) Add to it instances based on your ontology.

Make the instances use all parts of your ontology. Don't make all the instances trivial copies of one another. Remember that you are going to be querying this file in the next question, so you want to have interesting results. MAKE sure you use all the RDF constructs that we talked about

in the RDF lecture in class. That includes: BAG, ALT, SEQ, Blank nodes, Typed Literals, "catalog" and its entries. You can ignore Reification.

Use any legitimate RDF/XML you want. It would be best to use abbreviations to make your file more readable and compact.

You can look at usc-simpler-pure.rdf for my example based on the university ontology. [I have the ontology and instances in two separate files. You will have them combined into one.]

(c) Construct query files to do Jena command-line queries.

You should construct at least 5 interesting queries. Put each in a file so that you can use the command-line querying that you saw in the RQDL part of the tutorial. Here are a couple of queries that work for my files:

```
SELECT *
WHERE (<http://www.usc.edu/>, <u:#univDept> , ?dept)
      (?dept,  ?y , ?value)
USING u for <http://www.usc.edu/2001/univ-rdf/1.0>

SELECT ?x, ?fname
    WHERE (?x, <u:#univName>, ?fname)
    USING u for <http://www.usc.edu/2001/univ-rdf/1.0>
```