CS345 Final Examination

Monday June 7, 1999; 8:30–11:30AM

Directions

The exam is *open book*; any written materials may be used. Answer all 10 questions on the exam paper itself. The total number of points is 135. You may make additional comments at the bottoms of pages or at the end if you feel the need to clarify or justify your answers. Do not forget to **sign the pledge** below.

I acknowledge and accept the honor code.

____

Print your name here: ____

1	2	3	4
5	6	7	8
9	10		

Problem 1: (10 points) Consider the following two conjunctive queries with negation:

Q_1: `panic :- a(X,Y) & a(Y,Z) & NOT a(Z,X)`
Q_2: `panic :- a(X,Y) & NOT a(Y,X)`

a) Q_2 is not contained in Q_1. Give a counterexample database.

b) Now, consider whether $Q_1 \subseteq Q_2$. Of the partitions of $\{X, Y, Z\}$, for which does the basic canonical database yield the head of Q_1?

c) Is $Q_1 \subseteq Q_2$? Either complete the proof or give a counterexample.

Problem 2: (10 points) Consider the rules:

```
p :- q & NOT r
r :- NOT p
q :- NOT r
```

There are eight possible models — the eight subsets of $\{p, q, r\}$. Draw a graph below, with these eight subsets as nodes, and an arc $M \rightarrow N$ if $GL(M) = N$; i.e., the Gelfond-Lifschitz transform applied to model M yields model N.

Problem 3: (15 points) This question deals with the following Datalog program:

r_1: `p(X,Y) :- a(X,Y)`
r_2: `p(X,Y) :- a(X,Z) & q(Z,Y)`
r_3: `q(X,Y) :- b(Z,Y) & p(X,Z)`

a) Without reordering any subgoals in the rule bodies, draw in the space below the rule/goal graph for this program starting with the goal node p^{bf}.

b) Following your rule/goal graph from (a), rewrite the rules above by splitting predicates, so IDB predicates have unique binding patterns when the rule bodies are evaluated from left to right, in the order above.

c) If we reorder subgoals of the original Datalog program, we can develop an equivalent program whose rule/goal graph has the minimum possible number of goal nodes with IDB predicates. Tell how to reorder subgoals:

and show the rule/goal graph starting with goal node p^{bf} in the space below:

Problem 4: (15 points) Suppose we have a global $par(X, Y)$ "parent" predicate and the following two views that are limited in the binding patterns they allow:

v_1^{bf}`(X,Y) :- par(X,Z) & par(Z,Y)`
v_2^{bf}`(X,Y) :- par(X,Y) & par(Y,Z)`

We are also given the query:

Q: ans^{bff}(X,Y,Z) :- par(X,Z) & par(Z,W) & par(W,Y)

a) It is proposed that the following is a solution whose expansion is equivalent to the query:

S: ans^{bff}(X,Y,Z) :- v_2^{bf}(X,Z) & v_2^{bf}(Z,W) & v_2^{bf}(W,Y)

What is the expansion of S?

b) Does S meet the requirements for binding patterns in its subgoals? Explain briefly.

c) Is your expansion $E(S)$ from part (a) contained in Q? Give a containment mapping or a counterexample.

d) Is Q contained in your $E(S)$? Give a containment mapping or a counterexample.

e) Give a solution whose expansion is equivalent to Q. Show both the solution and its expansion, and show that the expansion is equivalent to Q.

Problem 5: (15 points) Here are rules that define the sublist relationship; that is, list L is a *sublist* of list M if L can be formed by striking out any zero or more elements from M. The deleted elements need not be consecutive. For example, the sublists of $[0,1,0]$ are nil, $[0]$, $[1]$, $[0,1]$, $[1,0]$, $[0,0]$, and $[0,1,0]$. The EDB predicate elt represents a finite set of elements that may appear on the lists.

r_1: `subl(nil,nil) :-`
r_2: `subl(cons(X,Xs), cons(X,Ys)) :- elt(X) & subl(Xs,Ys)`
r_3: `subl(Xs, cons(Y,Ys)) :- elt(Y) & subl(Xs,Ys)`

a) If $elt(0)$ and $elt(1)$ are true, then $subl(cons(0,nil),\ cons(0,cons(1,nil)))$ can be deduced. Show the sequence of needed inference steps that would be made during a bottom-up evaluation such as seminaive. You can just show the instantiated heads that are inferred and give the number of the rule used at each step.

b) If the query is $subl(X,\ cons(0,cons(1,nil)))$, i.e., find the sublists of the list $[0,1]$, then the only binding pattern for $subl$ is fb. We may therefore apply the magic-sets transformation directly. Using the division into five groups as outlined in class and the notes, write the rules:

Group I:

Group II:

Group III:

Group IV:

Group V:

c) Simplify the rules by eliminating 0th supplementary predicates and also eliminating any supplementary predicate that is used only once as a subgoal.

Problem 6: (15 points) Suppose we have a REFER-style database with the following entity sets and relationships (which we shall refer to as *views*):

1. $doctor(D)$ = D is a doctor.
2. $patient(P)$ = P is a patient.
3. $illness(I)$ = I is an illness.
4. $treats(D, P)$ = doctor D treats patient P.
5. $has(P, I)$ = patient P has illness I.

The following query asks which doctors are treating patients with the mumps.

```
ans(D) :- doctor(D) & patient(P) & illness('mumps') &
          treats(D,P) & has(P,'mumps')
```

a) In Hospital 1, the data about patients, their assigned doctors, and their illnesses is kept in a relation $Admits(Patient, Illness, Physician)$. Write the rules that define the five views (entities and relationships) listed above in terms of the relation $Admits$.

b) Express the query in terms of the relation $Admits$ by direct substitution for the views in the query.

c) Find a minimal equivalent query to the one in your answer to (b).

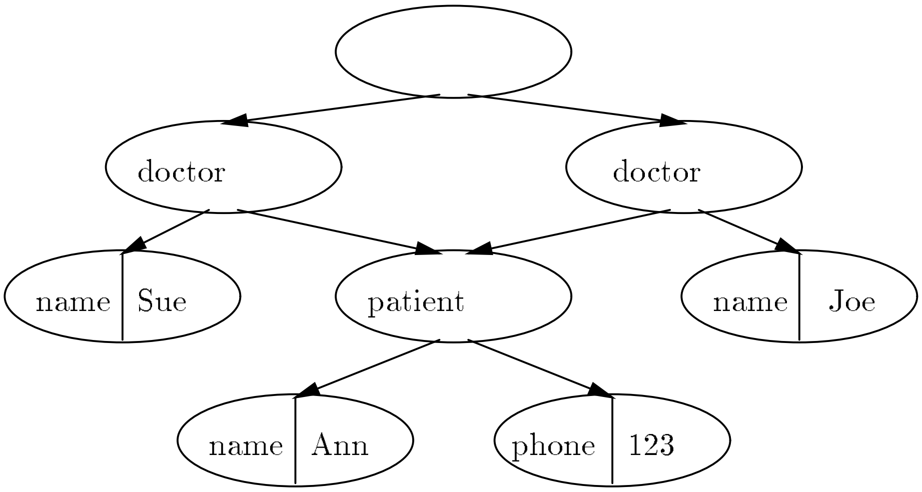
d) At Hospital 2, the policies are somewhat different. First, patients are assigned to a room, by a relation $Beds(Patient, Room)$. Then, doctors are assigned to rooms, and treat all the patients in that room; the relation $Assigned(Doctor, Room)$ makes the assignment. Last, patients are assigned to rooms so all patients in a room have the same disease. You can tell the illness of the patients in a room by the relation $Wards(Room, Illness)$. Write the rules defining the five views in terms of the relations of Hospital 2.

e) Write the query at Hospital 2 by expanding the views in the query in terms of the Hospital-2 relations.

---

f) Minimize your solution from (e).

Problem 7: (15 points) Below is an example object at a source `docSource` that involves doctor subobjects. Doctor subobjects in turn have name subobjects and patient subobjects. Patient subobjects have name and phone subobjects. However, since this data is semistructured, the figure below is only "typical." in fact, doctors and patients may have subobjects with labels other than those shown, they may have several subobjects with the same label, and they may have no subobject with the labels shown. Note further that we have adopted a simplified convention, where atomic objects have a vertical bar separating the label on the left from the value on the right, while other objects have a label and a set of arcs leading to their subobjects.



a) Suppose we wanted to keep information about patient ages. Suggest a reasonable way to incorporate such information into the `docSource` object.

b) Suggest a reasonable way to incorporate information about a patient's medical history, which consists of information about each time the patient was admitted to a hospital, the attending physician, and the diagnosis. Sketch *on the diagram of the previous page* a history subobject for patient Ann, who was treated by doctor Sue for mumps in 1996 and by doctor Joe for measles in 1998. Describe the general idea here:

c) At our "medical mediator" `medMed`, we want to have patients as immediate subobjects of the root object. Write an MSL rule that will define the mediator to have subobjects labeled `patient`, with name and phone subobjects. Your mediator should have one patient subobject for every patient that appears at the source `docSource`. Note that, because MSL, like Datalog, uses a set model, you do not have to worry about the possibility that the mediator would be given two copies of an identical object.

d) Now write MSL rules that do the same as (c), but also includes the name(s) of the doctor(s) that have treated each patient, as subobjects of the patient object.

Problem 8: (15 points) Below is a Datalog program with negation:

```
p(X) :- a(X) & NOT q(X)
q(X) :- b(X,Y) & p(Y)
q(X) :- c(X)
```

Note that in the second rule, the second subgoal is p, not q, and there is no negation in that rule. The EDB consists of the following five facts: $\{a(1), a(3), b(1,2), b(2,3), c(3)\}$.

a) Give the instantiated rules that do not have a false EDB subgoal. You should omit the true EDB subgoals from the instantiated rules.

b) Is this program and EDB stratified? Why or why not?

c) Is this program and EDB locally stratified? Why or why not?

d) If we compute the well-founded model by inference (not the alternating fixpoint), what positive inferences do we make in the first round about the atoms of interest: $q(i)$ and $p(i)$, for $1 \leq i \leq 3$?

e) At that point, what is the largest unfounded set?

f) In the space below, show the computation of the alternating fixpoint for as many rounds as are needed for convergence. *Note*: Part (f) is not intended as a follow-on to parts (d) and (e), which were about another method for computing the well-founded model.

Problem 9: (15 points) Suppose *par* is a global predicate, and there are two views defined in terms of *par*:

```
v1(X,Y) :- par(X,Z) & par(Z,Y)
v2(X,Y) :- par(X,Z) & par(Z,W) & par(W,Y)
```

a) Using Skolem functions, write inverse rules for *par* facts in terms of the views v_1 and v_2.

b) Suppose that the only fact in view v_1 is $v_1(0,1)$, and the only fact in v_2 is $v_2(1,2)$. Using your rules from (a), what *par* facts (including those with function symbols) can you infer?

c) Suppose now that we ask a query about the predicate *par*, and our query is the usual "ancestor" predicate defined by the Datalog program:

```
anc(A,B) :- par(A,B)
anc(A,B) :- par(A,C) & anc(C,B)
```

We now evaluate these rules and your rules from (1) using seminaive evaluation. Give an example of an $anc(X, Y)$ fact that we can infer with the following two properties:

i. There is at least one function symbol in X or Y.

ii. The analogous *par* fact, $par(X, Y)$, is *not* one of those that can be inferred from the inverse rules.

---

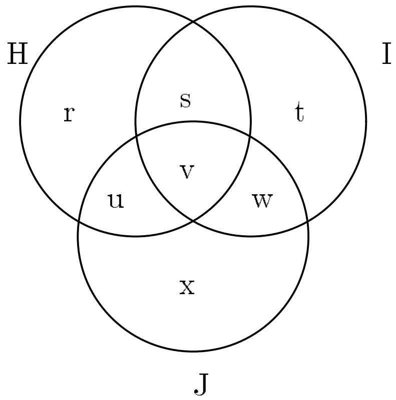
d) What are all the *anc* facts *without* function symbols that you can infer?

Problem 10: (10 points) [*This problem is rather tricky.*] The following questions involve a hypergraph with five nodes, A, B, C, D, and E.

a) Suppose there are two hyperedges, H and I. Every node is in one or both of the hyperedges. How many different selections of the hyperedges H and I make the hypergraph acyclic? Note that the hyperedges are named, so for example, the assignment ($H = \{A, B, C, D\}$; $I = \{C, D, E\}$) is different from the assignment ($H = \{C, D, E\}$; $I = \{A, B, C, D\}$). Explain your reasoning briefly.

---

b) Now, consider a hypergraph with three hyperedges, H, I, and J. We can divide the nodes into seven categories, depending on which subset of the hyperedges it is in; as before, we assume each node is in at least one hyperedge. The following "Venn diagram" gives us names for the seven regions:



Let the letter r stand for the statement "region r is nonempty," and similarly for the letters s through x. Write an expression (using these seven letters and the logical connectives `AND`, `OR`, and `NOT`) that is true exactly when the hypergraph is *cyclic*. Explain your reasoning briefly.

c) If the hypergraph from (b) has only the five nodes A, B, C, D, and E, how many assignments of hyperedges H, I, and J will be cyclic? As in (a), consider two assignments different even if they are the same sets of nodes assigned to different hyperedges H, I, and J. Explain your reasoning briefly.