Logic and Prolog Midterm, CSC173With Answers and FFQOct. 26, 20

Please write your name on the bluebook. This is a closed-book exam. There are 75 points, one per minute. Stay cool and write neatly.

1. Propositional Calculus, Karnaugh Maps, Circuits (15 min)

A. (5 min) Make a truth table for this PC formula:

$[(\neg X \wedge Z) \vee (\neg X \wedge \neg Z)] \wedge [(Y \vee \neg X) \Rightarrow (\neg Y \wedge \neg X)]$

(Hint: get this right!)

B. (5 min) Make a Karnaugh map for the resulting Boolean function (the column under the $\wedge$ between the [ ] expressions.)

C. (5 min) Convert the Karnaugh map into a circuit.

Answer:
given XYZ values as below, my column for the expression looks as below, from whence the K. map dictates $\neg X \wedge \neg Y$ which by DeMorgan is $\neg(X \vee Y)$, a two-gate circuit that doesn't use Z input.

```
X Y Z  EXP
0 0 0  1
0 0 1  1
0 1 0  0
       0
       0
...    0
       0
1 1 1  0
```

2. Models (10 min)

A. (5 min). Consider a vocabulary of three propositions, P, Q, and R. How many models are there of the sentence $P \Rightarrow Q$?

B. (5 min). Below are the models of three knowledge bases (collections of sentences, or one big CNF sentence, etc.) R, S, T. The vocabulary has three propositions A, B, C. Here 1 means True and 0 means False.
1. Does $S \models T$?
2. Does $T \models S$?
3. Does $R \models S$?
4. Does $R \models T$?

```
A B C  R S T
0 0 0  1 1 0
0 0 1  0 0 0
...    0 1 1
```

```
        1 1 1
        0 0 0
        0 0 0
        0 0 1
1 1 1   0 1 1
```

Answer:

A. There are 3*2 models of $P \Rightarrow Q$, 3 from TT for $P \Rightarrow Q$, then either value for R. There are $s^3 = 8$ models of the variables P, Q, R but that's not the question (though it's nice to point that out, too). Also I know my exposition of this was confusing this year (the new overheads should fix that though).

B. $A \models B \Leftrightarrow M(A) \subseteq M(B)$, so only entailment here is $R \models S$. People did very well on this.

3. FOPC Inference (15 min)

Given these premises:

1. Anyone who procrastinates is sluggish.
2. Everybody has a friend who procrastinates.

Express them in FOPC, put them into clause form, and use resolution to answer the question: Is anyone sluggish and if so, who?

Answer:

$$\forall x(P(x) \Rightarrow S(x))$$

$$\forall x \exists y(F(x, y) \wedge P(y))$$

So we get clauses

$$\neg P(x) \vee S(x) \quad (1)$$

Skolemizing with a function f(x),

$$F(x, f(x)) \quad (2)$$

$$P(f(x)) \quad (3)$$

Assert that no one is sluggish:

$$\neg S(x) \quad (4)$$

Unify (3) and (1) , and the result with (4) (or the other order) and we find that if you give me any x, I'll give you good old $f(x)$, the individual generated by the Skolem function, in short it's x's laid-back friend who procrastinates and who is sluggish. This is the lackadaisical individual who gives us the substitution that allows the derivation of the null clause. Another correct way to

axiomatize this situation would be to use another assertion to the effect that someone really does exist. This existence will skolemize to some individual A, and it will then be A's friend who puts things off until the last minute.

The FOPC question continued to give the most trouble this year. Some made the classic mistake of translating 2nd statement to $\forall x \exists y(F(x,y) \Rightarrow P(y))$. Some got confused by forgetting to remove the $\wedge$ that connects clauses (2) and (3) initially. Lots of people blanked on resolution proof, and almost nobody seems to understand Skolemization, nor how to use the results of unification to answer the question. Maybe a vague idea, but not crisp.

4. Unification (10 min)

A. What is the most general unifier of these clauses as given by Prolog (below)? (List the substitutions and show the unified result clause). Here the predicate, function, and constants (l; g; and a, c,d respectively) are lower case, the variables (U, V, W, X, Y, Z) are upper case.

?- l(X,Y,g(a,Y),d) = l(Z,c,g(W,U),V).

B. Show a less general unifying substitution and the result of that unification.

Answer:

A.

```
X = Z,
Y = c,
W = a,
U = c,
V = d.
l(X,c,g(a,c),d)
```

B. Use $X \mid e$ and $Z \mid e$, instead of $X \mid Z$, say. The idea is to have less generality by having fewer variables and more individuals result from the unification. So that substitution looks like $l(e,c,g(a,c),d)$, which is clearly less general since it now has no universally-quantified variable.

3-SAT (10 min)

Gloomy Gus: "Everyone knows 3-SAT is NP-complete...it's hopeless, we're doomed!"
Perky Polly: "That's only worst case, silly...chin up, look for the bluebird!".
Cheekie Chrissie: "Satisfy THIS!".
$(C \vee E \vee \neg A),\ (\neg B),\ (B \vee \neg C \vee \neg A),\ (\neg A \vee C \vee \neg D),$
$(E \vee \neg A),\ (D \vee \neg A \vee \neg E),\ (B \vee \neg A),\ (C \vee \neg A \vee \neg E)$
Tactful Tammy: "I think Christopher means we should all just sit down and find truth values for A, B, C, D, E so all those clauses (and thus the AND of them all) are true."

Answer: The unit clause heuristic linearizes this problem, starting with (-B) and deriving (-A) and that's it. A learning experience...most people pulled a Polly and invented the unit-clause trick and also the idea that it can propagate to yield more unit clauses. And also the idea that one true literal means whole disjunct is true is another powerful speedup. These are some of the tricks used by modern SAT-solvers to satisfy sentences with millions of clauses: very useful in many applications!

Those Gloomy Guses who slogged thru pages of model-checking now at least can be motivated by the power of these ideas when they see it's a one-line SAT problem.

Prolog Structures (15 min.)

```
wordsof(X,[X]) :- word(X).
wordsof(X,P) :- grammat(X,Subparts), wordsoflist(Subparts,P).

wordsoflist([],[]).
    wordsoflist([P | Tail], Alist) :-
    wordsof(P,Headwords), wordsoflist(Tail,Tailwords),
    append(Headwords, Tailwords, Alist).

grammat(sentence,[noun_phrase,verb_phrase]).
grammat(noun_phrase,[ det, noun]).
grammat(noun_phrase,[ johnny]).
grammat(verb_phrase,[transv, object]).
grammat(verb_phrase,[intransv]).
grammat(det,[a]).
grammat(noun,[boy]).
grammat(noun,[snowball]).
grammat(transv,[throws]).
grammat(intransv,[throws_up]).
grammat(object, [noun_phrase]).
word(johnny).
word(a).
word(boy).
word(snowball).
word(throws).
word(throws_up).
```

A. Above is a Prolog database that defines a structure whose name is `grammat`. Draw a picture of that stucture (nodes and arcs) below, labeling the nodes appropriately (by their functor names).

B. Describe in one English sentence what the prolog query
`?- wordsof(sentence, X).`
(along with liberal use of the semicolon) can accomplish.

C. Give some example prolog output from the query in part B. above.

Answer:

A. Imagine a nice tree that looks like a good old tree version of the given grammar.

B. generate all legal sentences from the grammar.

C. as below

```
?- wordsof(sentence,X).
```

```
X = [a, boy, throws, a, boy] ;
X = [a, boy, throws, a, snowball] ;
X = [a, boy, throws, johnny] ;
X = [a, boy, throws_up] ;
X = [a, snowball, throws, a, boy] ;
X = [a, snowball, throws, a, snowball] ;
X = [a, snowball, throws, johnny] ;
X = [a, snowball, throws_up] ;
X = [johnny, throws, a, boy] ;
X = [johnny, throws, a, snowball] ;
X = [johnny, throws, johnny] ;
X = [johnny, throws_up] ;
false.
```

People generally did well on this. Good on ya!