CSC173
LOGIC, PROLOG 2014
ANSWERS AND FFQ
6th Nov 2014

Please write your name on the bluebook. You may use two sides of handwritten notes. 85 points possible, but perfect score is 75 points (one per minute). Stay cool and please write neatly.

Points and question structure a bit different from actual exam but all questions and answers exactly the same. – CB

1. Propositional Calculus, Karnaugh Maps, Circuit: 15 min

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

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

(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 function looks as below, from whence the Karnaugh 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  FN
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 four propositions, P, Q, R, S. How many models are there of the sentence $(P \Rightarrow Q) \vee R$?

B. (5 min). Below, the three left columns enumerate the models of the eight-member *domain* of variables A, B, C. The three right columns are the models of three *knowledge bases* (collections of sentences, or one big CNF sentence, etc.) R, S, T whose variables are A, B, C. Here 1 means True and 0 means False.

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

1. Does $S \models T$?
2. Does $T \models S$?
3. Does $R \models S$?
4. Does $R \models T$?

Answer:

A. There are 14 models of $(P \Rightarrow Q) \vee R$ from its truthtable, plus either value for S. There are $2^4 = 16$ models of the variables P, Q, R, S but that's not the question (though it's nice to point that out, too).

B. $X \models Y \Leftrightarrow M(X) \subseteq M(Y)$, so only entailment here is $R \models S$.

3. Inference: 15 min

Lord Peter: "Somebody had a key".
Mr. Holmes: "Anyone with a key is a suspect, or there was a break-in".
Father Brown: "A break-in leaves evidence and there wasn't any."
All together: "Do we have any suspects?"

1. Translate the first three assertions into three non-clausal, reader-friendly FOPC sentences with arrows and quantifiers as needed. Use the domain of people. Use K(x) for "x has (or had) a key", S(x) for "x is a suspect", B for "there was a break-in", E for "there was evidence".

2. With a view to helping the police with their enquiries using question-answering methods with resolution theorem proving, put the last question into a FOL sentence.

3. Translate all four FOL sentences into clause form.

4. Use resolution theorem proving to find the suspect(s) in the case, if any. Remember that here the last question is "what you're trying to prove" so make sure you follow the usual procedure to obtain the form of a proof by contradiction that will answer the detectives' question. Identify the clauses involved in each resolution operation and their resolvent.

Ans. for Question 3:

1. $\exists(x)K(x)$
   $\forall(x)[K(x) \Rightarrow S(x)] \vee B$
   $(B \Rightarrow E) \wedge \neg E$

2. $\exists(x)S(x)$. Assert there's a suspect out there, which is how we frame question-answering proofs.

3. $K(a)$. The existential quantifiation gives a Skolem constant, some individual a.
   $\neg K(x) \vee S(x) \vee B$
   $\neg B \vee E$
   $\neg E$ (Father Brown's third sentence gives two clauses)
   either leave last sentence alone with view to negating it in the next step, or you can negate it now in prep for RTP and QA: ultimately you get the answer:
   $\neg S(x)$.
   The point is not to Skolemize it first. Then one may add $\vee Ans(x)$ to it to extract the answer (as below): $\neg S(x) \vee Ans(x)$.

4. Below $->$ shows resolution, inputs above, resolvents at right.:

```
   ~B v E
      ~E
-> ~B
    B v S(x) v ~K(x)
->  S(x) v ~K(x)
           K(a)
->  S(a)     (unification and resolution)
   ~S(x) v Ans(x)  (to extract answer)
->  Ans(a)  (unification and resolution)
```

   So our suspect is a, that nefarious, lurking Skolem constant who has or had a key. Of course a could be anyone, so plenty of detective action ahead!

4. 3-SAT: 10 mins

A. (5 min): Find truth values for A,B,C,D,E that satisfy the following conjunctive normal form (CNF) expression. This means all five clauses must be satisfied. Some variables may be either T or F.

$$(D \vee A \vee \neg E) \wedge (\neg B \vee C) \wedge (\neg A \vee \neg C) \wedge (B \vee C) \wedge (A \vee D \vee E)$$

B. (5 min): Part A is an example of 3-SAT, the problem of deciding if a CNF expression, with each clause having ≤ 3 variables, can be satisfied. How hard is 3-SAT in general? If your solution to A did not "feel that hard", why not?

Ans.

4A. With 0 for false, 1 for true: B and E can be either 0 or 1.

```
A B C D E
0 0 1 1 0
  1     1
```

4B. 3-SAT is the original and most famous (and very useful) worst-case NP-Complete problem, meaning we don't today have a general solution in better than exponential time. If we had a polynomial-time solution then several other important, exponential-seeming problems are also polynomial.

This small problem instance is easily done by constraint propagation. First, establish C is true and B can be anything (2) and (4), thus A must be false from (3) and knowing $\neg A$ allows similar reasoning on D and E through (1) and (5). We got a linear-time (O(N)) solution for our N variables. The hope is that the testee either invents it (maybe remembers it if we touched on it in class) or appreciates it if he did a brute force search of 32 5-tuples.

5. Unification: 10 mins

A. (5 min): Find the most general unifier of these two clauses, written here as in Prolog, with Variables Capitalized and constants, predicates, functions all in lower case.

```
p(f(g(X,Y),h(Y)), r(h(g(Z,a)),g(Y,Z)))
p(f(g(W,V), h(V)), r(h(V),W)).
```

B. (5 min.) Show a less-general unifier for these two clauses.

Answer:

5A:

```
X = g(g(Z, a), Z),
Y = g(Z, a),
W = g(g(Z, a), Z),
V = g(Z, a).
```

or

```
X = W
Y = V
W = g(g(Z, a), Z),
V = g(Z, a).
```

5B: Substitute `b`, say, everywhere for `Z` in the MGU. match still works, but less general (no universally quantified variable.) This answer is more obvious in the first form of the answer (only one variable on the right-hand side) – having W and V running around on RHS may lead to some confusion.

6. Prolog: 10 mins

A. Consider the predicate p/4 defined as the four-argument function:

```
p(_,[],[],[]).
p(P,[X|Xs],[X|L1],L2) :- X < P, p(P,Xs,L1,L2).
p(P,[X|Xs],L1,[X|L2]) :- X >= P, p(P,Xs,L1,L2).
```

A.1 (3 min): What does p/4 do? Show an example query and Prolog's response.

A.2 (3 min): What types of terms cannot be used as the first p/4 argument?

A.3 (2 min): Describe where and why cuts can be used in the p/4 predicate.

ANSWER:
6A.1: p/4 is true if its first argument is a value P, second is a list L of items, third is a list of items from L that are less than P, fourth a list of items from L that are $\geq$ P. Most obvious type of example is

```
?- p(3, [1,3,5,4,2], L, M).
L = [1, 2],
M = [3, 5, 4];
false.
```

6A.2: The first argument must be comparable to items in the lists. So for lists of integers, integer 4 is OK first argument but list [4,3] is not.

6A.3: Putting ! in between the comparison and recursive call in the last two rules is natural. Turns out that accomplishes nothing in our example above, but read on –

```
% A different query using p/4, without cuts

?- p(3,X,[1,2],[4,5]).
X = [1, 2, 4, 5] ;
X = [1, 4, 2, 5] ;
X = [1, 4, 5, 2] ;
X = [4, 1, 2, 5] ;
X = [4, 1, 5, 2] ;
X = [4, 5, 1, 2] ; % all true but maybe TMI and
                   % too much work...doesn't scale
false.

% With cuts
?- q(3,X,[1,2],[4,5]).
X = [1, 2, 4, 5] ;  % *boom!*
false.
```

7. The Cut!: 10 mins

The Cut! Suppose we consult the following database of facts and rules:

```
a(1).
a(a).
b(2).
b(3).
c(X,X) :- a(X).
c(X,Y) :- a(X),!,b(Y).
c(X,X) :- b(X).
```

List all of the possible solutions to the query
`?-c(A,B).`
in order.

ANSWER:

7. The right answer is first below, and the second result shows what would happen without the cut.

```
?- c(A,B).
A = B, B = 1 ;
A = B, B = a ;
A = 1,
B = 2 ;
A = 1,
B = 3.

% Now, just for fun, what was the cut's effect?
% without it...

?- c(A,B).
A = B, B = 1 ;
A = B, B = a ;
A = 1,
B = 2 ;
A = 1,
B = 3 ;   % the cut eliminated the next three ans's
A = a,
B = 2 ;
A = a,
B = 3 ;
A = B, B = 2 ;
```