Midterm Answers, with FFQ(TM) feature

CSC 242

March 2007

*Write your **NAME** legibly on the bluebook. Work all problems. You may use two double-sided pages of notes. Please hand your notes in with your bluebook. The best strategy is not to spend more than the indicated time on any question (minutes = points).*

Thanks to Paul Ardis for help in test debugging. All remaining problems are my own – CB.

1. FOPC: 15 Min.

Put the following assertions into FOPC formulae, convert to clause form, and use resolution to answer the question "What courses would Bob like?". Make sure you show what converts to what, what resolves to what, etc. so the process is clear, not just the answer.

Let's keep this simple by assuming "typed" variables: in particular let's assume that x varies over the domain of courses and y varies over the domain of departments. Thus you don't have to assert and carry around facts like $Course(x)$ or $Department(y)$. Use the predicate notation $L(x), E(x), F(x)$ for "Bob likes x, x is Easy, and x is Fun".

A. *Bob likes any course that is easy or fun.*
B. *Every department has at least one easy course.*
C. *No courses are fun.*

Answer:

$$A : \forall(x)(F(x) \vee E(x)) \Rightarrow L(x)$$

FFQ: Bob liking it doesn't force it to be easy or fun: he could like hard courses too, this doesn't say. So don't use $\Leftrightarrow$ for this.

$$B : \forall(y)(\exists(x)E(x))$$

$$C : \forall(x)(\neg F(x))$$

Thus

$$A \rightarrow (\neg F(x) \wedge \neg E(x)) \vee L(x)$$

And we get clauses (numbered for convenience)

$$A \rightarrow 1.\ \neg E(x) \vee L(x)$$

$$A \rightarrow 2.\ \neg F(x) \vee L(x)$$

FFQ: lots of people did not use the distributive law to get real CNF clauses.

$$B \rightarrow 3.\ E(O(y)),$$

where $O(.)$ is the Skolem function returning the easy course offered by department y.

FFQ: need a Skolem something, and it's a function: the course depends on the department.

$$C \ \rightarrow 4.\ \neg F(x)$$

One way to extract the answer is just to assert that Bob doesn't like any courses, look for a contradiction (deriving null clause), and keep track of the substitutions for x. Alternatively, you can assert an $Ans(x)$ predicate along with the negated assertion... in the latter case you need to watch when that clause has the negated question resolved away...

$$5.\ \neg L(x) \vee Ans(x)$$

Now we can resolve 5 and 1 to get

$$6.\ \neg E(x) \vee Ans(x)$$

and 6 and 3 to get the null clause with x bound to $O(y)$, the easy course offered by a department, or (as shown here) with $Ans(O(y))$ appearing in our "question clause".

FFQ: DON'T FORGET to assert the negative of what you're trying to prove! The algorithm doesn't terminate until it derives the null clause.

2. More FOPC: 10 Min.

FFQ: this question is a little loose since it starts out questioning semantics of these horrible English sentences! In fact a student came up with a reading I hadn't figured out: "Only Bob" modifies likes, so he's he only person who likes easy courses.

FFQ: Generally I think the sharpest insight here is you need E(x) on the right hand side of your clauses (LHS of implictions) or you can't resolve (modus ponens) anything, so none of the English that allows it to be on the left (right) is what the authors want.. OK here comes the question—

Question 1 above was inspired by an exercise in a well-known AI text (not ours, but we did use it for several years). In part, it asks you to translate two sentences like the following into FOPC and then infer something from them.

D. *Bob only likes easy courses.*
E. *CSC282 is an easy course.*

I claim the authors screwed the pooch a couple of ways here.

2.1 (3 Min): First, the example doesn't work: we have no inference rules that apply. Rephrase D into unambiguous English, translate into FOPC, and observe you're stuck with nothing to deduce. (Don't use typed variables.) Say why it fails to function as expected.

2.2 (2 Min): Second, the bad example is also ambiguous. Rephrase D into *different* unambiguous English and proceed as in 2.1.

2.3 (3 Min): Now state in umambiguous English a sentence that actually accomplishes what the authors intended by D, translate that into FOPC, and show you can make the obvious deduction,

Ans.

2.1. If Bob likes a course, it's easy.

$$\forall x((C(x) \land L(x)) \Rightarrow E(x))$$

Notice that this, taken with $(E(A) \land C(A))$, which is derived from E. using Skolem constant A to represent CSC282, will not let you conclude anything, say by trying to use *modus ponens* or resolution.

2.2. If Bob likes anything, it's a course and it's easy.

$$\forall x((L(x) \Rightarrow E(x) \land C(x))$$

Same problem.

What they meant to say was:

2.3. If a course is easy, then Bob likes it.

$$E(x) \land C(x) \Rightarrow L(x).$$

Along with $(E(A) \land C(A))$, this allows conslusion that Bob likes CSC282, as he should.

3: Minimax and $\alpha - \beta$ 10 Min.

Consider two-person **non zero-sum games**, in which the players may have different utility functions. Suppose each player knows the other's utility function.

1. What has to change in the minimax algorithm, if anything, and why?
2. Does alpha-beta pruning still work? Why or why not?

Ans. 1. Minimax will work as usual if it's set up right. We'll be backing up a vector of evaluations and at each level the player will choose what is best for him, even if it is also good for the other player. Thus if we assign increasingly positive values for states increasingly better for Max and increasingly negative values for states increasingly better for Min, then minimax will work unmodified. If both players have increasingly positive values, each player just picks the maximum value, so it's a "maximax" algorithm.

2. However, alpha-beta pruning will not work because built into it is the idea that what's good for max is bad for min – for example min won't let max go down a path since min can force something worse, so max knowing this doesn't have to explore that path. But without zero-sum assumption, the same state could be good for both min and max; you can't assume that just because max likes it that min won't, and *vice versa.*

4. Three-Person Nondeterministic Game: 10 Min.

A, B, and C all love D and decide on a duel with pistols to settle the matter once for all. A, B, and C are rational and each desires to survive and win. A is a lousy shot, averaging 1/3 kill, 2/3 miss per shot. B is better, with 2/3 kill, 1/3 miss per shot. C is a crack shot, 1/1 kill, 0/1 miss per shot. They therefore agree that A will shoot first, then (if alive) B, then (if alive) C, then (if alive) A, and so on around the order until there is only one survivor. You are A's second: what's your advice about whom A should shoot at first, and why?

Ans.

Tell A to shoot in the air (had to be, right?). If A first kills B, C kills A certainly and immediately. If A first kills C, then A and B have a probabilistic duel with B having first shot. This actually works out, via geometric series sum, to a probability of 1/7 that A will live but you could approximate these probabilistic duels by 2:1 odds and get the same strategy. If A first kills neither, then B has 2/3 chance of killing C, after which it's almost like A had killed C. That is, a probabilistic duel between A and B with A having first shot – the probability of A living in this scenario works out to (2/3)(3/7) = (6/21), by geometric series. However, there is also a 1/3 chance that B will miss C, in which case C kills B immediately, and thus with total probability (1/3)(1/3) A will shoot C his 2nd (and, one way or the other, last) time round. Thus by my calculations A's chances of living by firing in the air are (6/21) + (1/9) = approx .41.

FFQ: most of the answers were better than the one provided on the *Car Talk Puzzler* from which I stole this. I'd heard it before, tho. People mostly got it, except that I probably should have said there should be no collusion between A, B, and C... still, I'm pretty sure it doesn't change anything anyway. One person had a glimmering that you need to solve some series to get the exact probabilities, but nobody made the full leap.

5. CSP: 15 Min.

CryptoBoy's got a crossword puzzle grid (that is, an array of blank squares and blacked out squares) and a fairly small vocabulary of specialized words; he'd like to maximize the number of words from the special vocabulary that are placed in the grid. A complete solution would be good but very unlikely: he plans to try to fill remaining blanks himself. He decides to use backtracking (depth-first tree search) to place words.

A. Describe the situation formally as a Constraint Satisfaction Problem (variables, values, constraints, success criteria).

FFQ: most people figured out that word spaces, not letter spaces, are the right variables to be filled up, but not everyone...

B. How can he implement the fact that the best solution could have spaces with no words from the vocabulary assigned? What are the consequences for his search? Can he speed things up with constraint propagation or forward checking?

C. Actually, CryptoBoy always forces certain spaces to contain a particular word (or small subset of vocabulary words). Does this allow constraint propagation of any sort and if so exactly what?

Ans. A. Variabless are the horizontal and vertical spaces to be filled up by words from the vocabulary. Each one has a length.

Values are the words.

Unary constraint: the length of a word assigned to a space is the length of the space.

Binary constraint: where two nonblank spaces intersect, the associated words must share the same letter at that point of intersection.

Success: This isn't a succeed-or-fail problem,, it's a maximization problem – maximize the number of letters placed, the number of words placed, the number of words weighted by their length placed, somesuch metric.

B. Allow a special blank word of each length that does not violate the shared-letter constraint. Assigning it to a space means that the algorithm is giving up on using a vocabulary word there. Unfortunately this seems to mean that it is hard to guide the search – one could try to assign longer spaces first, but the pesky blank word means that there is no real concept of failure, and so search must go down to the maximum depth (number of spaces) before you can give up. Worst, since every space has a potential value (blankword) that does not conflict with any of its neighbors, there is no way to take advantage of constraint propagation, forward checking, etc.

FFQ: suggestion: just search down to a given depth, increasing it incrementally. Actually CryptoBoy does that, but you have to have some way to order the spaces and going down to a particular depth has to leave some out. CB tries to place long words first.

C. Forcing a space to be non-blank does allow for constraint propagation, but only to the spaces that intersect it. Since the blank word is a possible value for each of these neighbors the constraints cannot propagate beyond this one level. CryptoBoy was able to knock down the possibilities from 10^{20} to 10^9 for a 13×13 puzzle with a few forced words, and you usually have them available since presumably they form the hook or theme of your puzzle[1].

6. Pronoun Reference: 15 Min.

You've been hired by Google as a consultant in Natural Language Understanding. Your first job is to find antecedents (references) for pronouns. In fact, your first assignment is:
A. Give the antecedents for all the pronouns (*italicized*) in the following little story. That is, complete the list:
him – John.
he – ...
...
it – ...

I think there's one obvious reading — if you want to be cute feel free, but explain yourself.
B. Say what sort of knowledge (syntactic, lexicographic, semantic) you needed to do A.

FFQ: I should have said: "each indidual case of A". Most of you did the right thing here, which is to convince me you understand the issues.
C. Recommend technical approaches to implementing antecedent-finding in a natural language understanding system.

A Little Story

John went to the store to buy a shirt. The salesclerk asked *him* if *he* could help *him*. *He* said *he* wanted a blue shirt. The salesclerk found *one* and *he* tried *it* on. *He* paid for *it* and left.

Ans.

him, John. he, clerk. him, John. he, John. he, John. one, blue shirt. he, John. it, blue shirt. he, John. it, blue shirt.

The first him would need lexical information to identify him as a person, and one could find John as the most recent person mentioned who wasn't the subject of the current sentence, which is a lexico-syntactic sort of way to go about this pronoun reference question. Likewise with the "one", the most recent noun before current subject is "blue shirt". Also, there is only one impersonal noun in the story, so we can safely bind "it" to shirt with purely lexical information.

Aside from that, I see almost entirely script-driven or role-driven (i.e. some formalization of "common-sense knowledge") reference-finding used here. One could possibly get some use out of lexical entries for salesclerk, but it seems weak, and there is no explicit linking of John to a "customer" concept in the story. So somewhere we'll have to encode the roles and actors in a buying-and-selling script or scenario, and match those to the unfilled slots in the story (the pronouns).

[1]For a good time, google CryptoBoy and find puzzles, code, etc.