Comprehensive Examination and Answers

Artificial Intelligence

May 11, 2006

Work each question in a separate bluebook. Write the problem number and your PIN on the bluebooks. There are 7 problems. Answer any 6 problems. No reference materials.

Problem 1: Resolution and answer extraction

In a theatrical domain, where comedy and drama are types of play:

Every drama is a play.
In every drama there is a villain.
Every villain in a drama gets killed.
No comedy has a villain in it who gets killed.
Othello is a drama.

A. Formalize these propositions as FOL formulas using (only) the predicates $P(x)$ for "x is a play", $C(x)$ for "x is a comedy", $D(x)$ for "x is a drama", $V(x, y)$ for "x is villain in y", $K(x)$ for "x gets killed", and O for "Othello".

B. Convert the formulas to clause form and use resolution with answer extraction to answer the question,

What play is not a comedy?

Be sure to indicate at each step which literals of which clauses you are resolving.

C. Is your proof a *unit proof*? a *set of support proof*? an *ordered proof*? Why or why not?

D. Factor $\mathrm{P(A)} \vee \mathrm{P(x)} \vee \mathrm{P(f(x))}$ in all possible ways. (N.B.: Wrong factors lose marks!)

E. Paramodulate $\mathrm{f(x)} = \mathrm{A} \vee \mathrm{P(x)}$ into $\mathrm{Q(f(y))}$ in all possible ways.

Problem 1 Answers

A: 1. $\forall \mathrm{x}.\, \mathrm{D(x)} \Rightarrow \mathrm{P(x)}$
2. $\forall \mathrm{x}.\, \mathrm{D(x)} \Rightarrow \exists \mathrm{y}.\, \mathrm{V(y,x)}$
3. $\forall \mathrm{x}.\forall \mathrm{y}.\, (\mathrm{D(x)} \wedge \mathrm{V(y,x)}) \Rightarrow \mathrm{K(y)}$
4. $\forall \mathrm{x}.\forall \mathrm{y}.\, (\mathrm{C(x)} \wedge \mathrm{V(y,x)}) \Rightarrow \neg \mathrm{K(y)}$
5. $\mathrm{D(O)}$

B: Q: $\exists \mathrm{x}.\, \mathrm{P(x)} \wedge \mathrm{Nt}\ \mathrm{C(x)}$
Denial: $\forall \mathrm{x}.\, \neg \mathrm{P(x)} \vee \mathrm{C(x)}$

1. $\neg \mathrm{D(x)} \vee \mathrm{P(x)}$
2. $\neg \mathrm{D(x)} \vee \mathrm{V(f(x),x)}$ f(x) is a Skolem function
3. $\neg \mathrm{D(x)} \vee \neg \mathrm{V(y,x)} \vee \mathrm{K(y)}$
4. $\neg \mathrm{C(x)} \vee \neg \mathrm{V(y,x)} \vee \neg \mathrm{K(y)}$
5. $\mathrm{D(O)}$
6. $\neg \mathrm{P(x)} \vee \mathrm{C(x)} \vee \underline{\mathrm{Ans(x)}}$ denial, with answer predicate

7. P(O) r[1a,5]
8. V(f(O),O) r[2a,5]
9. $\neg$V(y,O) $\vee$ K(y) r[3a,5]
10. K(f(O)) r[8,9a]
11. $\neg$C(x) $\vee \neg$V(f(O),x) r[4c,10]
12. $\neg$C(O) r[8,11b]
13. $\neg$P(O) $\vee$ Ans(O) r[12,6b]
14. $\square \vee$ Ans(O) r[7,13]

Thus, Othello is a play that is not a comedy.

C: The above proof is
– a unit proof, as each step uses a unit clause;
– *not* a set-of-support proof, as steps 7-12 don't use the S of S ({6}); – *not* an ordered proof, as steps 11-13 use non-initial literals

D: f[a,b]: P(A) $\vee$ P(f(A));
that's all – no other factors are possible

E: 1. f(x) = A $\vee$ P(x)
2. Q(f(y))

p[1a$_L$,2]: Q(A) $\vee$ P(x)
p[1a$_L$,2(subterm)]: Q(f(A)) $\vee$ P(x)
p[1a$_R$,2]: Q(f(f(x)) $\vee$ P(x)

Problem 2: Action and Planning

A: If the server is nearby and I wave, the server will attend to me.
If the server is attending to me and I request item x, I will (soon) have x.

In the initial state the server is nearby: we want to achieve the goal of having the ketchup, K.

Use the situation calculus and answer extraction to find a plan to achieve this goal. *Perform resolutions to simulate a regression search, i.e., from the goal backward.* Use a simple representation as follows:
$N(s)$ means that the server is nearby in state s;
$A(s)$ means that the server is attending to you in state s;
$H(x, s)$ means that you have item x in state s;
$do(a, s)$ is the state resulting from doing action a in state s;
W is the action of waving (to the server);
$r(x)$ is the action of requesting item x; and
K is the ketchup.

For each resolution step, clearly indicate which literals of which clauses you are resolving.

B: Imagine that you also want to have mustard M, which you could obtain with a separate request (after the request for ketchup). With the axioms you have used so far, would you be able to derive a plan leading to a state in which you have both the ketchup and the mustard? If so, indicate in words (without getting into the actual proof) how this would go. If not, say why not (again, without getting into formal details).

C: In a STRIPS encoding of this problem (which you *don't* need to give) would there be any difficulties about the "ketchup and mustard" problem mentioned in (b)? Why or why not?

Finally, answer just one of parts D or E:

D: What are the advantages of methods based on planning graphs (Blum & Furst '95) or on SAT-solving (Kautz & Selman's SATPLAN '96) (where you should just consider *one* of these two approaches), compared with earlier planners that performed regression searches, seeking a "flawless" plan (e.g., SNLP- or POP-style planners)?

E: What is advantageous about *hierarchical* planning, often called "hierarchical task network (HTN) planning"? (You may remember this idea from Sacerdoti's early planner NOAH, as well as various later systems like PRS and RAP.)

Problem 2 Answers

A: 1. $\neg N(s) \vee A(do(W,s))$
2. $\neg A(s) \vee H(x,do(r(x),s))$
3. $N(S_0)$

Goal: $\exists s\, H(K,s)$
Denial: $\forall s\, \neg H(K,s)$

4. $\neg H(K,s) \vee \underline{Ans(s)}$
5. $\neg A(s) \vee \underline{Ans(do(r(K),s))}$ r[4,2b]
6. $\neg N(s) \vee \underline{Ans(do(r(K),do(W,s)))}$ r[5,1b]
7. $\square \vee \underline{Ans(do(r(K),do(W,S_0)))}$ r[3,6]

Thus the plan is to wave to the server and to request the ketchup.

B: With the axioms so far, we couldn't prove the existence of a state in which we have both K and M, because there is no assurance that when we wave to the server again, or request the mustard, the ketchup will "stay put"; this is the *frame problem*. So we would need frame axioms saying that during waving and requesting I do not cease to "have" whatever I had before; or explanation closure axioms saying that the only actions that falsify a "have"-relation are, say, a taking-away action or a using-up action.

C: In a STRIPS encoding the frame problem is dealt with by the *STRIPS assumption*, i.e., that the only things that change are those explitly changed by the operators that are applied. So once I have K, I will continue to have it, as long as I apply no operator whose result is that I cease to have it.

D: These methods are faster because in effect they combine forward and backward search. GRAPHPLAN makes a forward sweep that finds all the facts that *might* be true at a given time step in a plan, and all the operators that *might* be applicable. This is followed by a backward search (from a state in which the goal *might* be true, and this search is greatly constrained by the initial forward sweep (and by incompatibilities of certain properties and certain actions with one another). SATPLAN performs a greedy search for a sequence of actions (operator instances) leading to the goal, by iteratively trying to maximize the number of true formulas among the set of formulas that describe the effects of the possible actions at each time point, and what conditions persist. The boolean variable whose value is "flipped" at any point could be concerned with an action or state at any time point in the emerging plan.

E: The advantage of hierarchical planning is that the "combinatorics" of a high-level plan are much more manageable than those of a detailed plan (as a few high-level steps will achieve the goal). Hence, once a high-level plan has been constructed, then – assuming it is workable – the lower-level planning can also be broken into manageable chunks, each chunk corresponding to a step of the high-level plan.

Problem 3: Games

For A and B, consider a three-person zero-sum game of perfect information.

A: Can the minimax algorithm be modified to apply to this situation? If so how, and if not why not?

B: Will $\alpha - \beta$ pruning work in this situation? If so how, and if not why not?

C: Now consider two-person games. It seems reasonable in a zero-sum game to represent MIN's and MAX's evaluation of a particular position as negatives of one another. How does this simplification affect the implementation of the $\alpha - \beta$ algorithm?

D: If we believe the fortunes of a game (say chess) are not going to swing wildly in a single move, why do we initialize α and β to $-\infty$ and $+\infty$? Would it not make more sense to initialize them to a smaller window of possible evaluation scores and thus not consider any "silly" moves? Or is this itself a silly idea? One worry might be that we have a search failure, when no moves evaluating into this window are found...what should we do then?

Answer Problem 3: Games

A: Yes. let X,Y.Z be the players. Worst case, X assumes Y and Z are colluding against him. The game tree's levels are for moves by XYZXYZ... X lumps the YZ move-pairs into one logical move and minimaxes against that "super-opponent".

B: Definitely. There is nothing different between the super-opponent YZ and a single opponent. $\alpha - \beta$ pruning works exactly the same way in this zero-sum situation as it always did, using the worst that Y and Z colluding together can do to X to generate the β cutoffs.

C: This simplification to minimax (Negamax) means the $\alpha - \beta$ can change from a co-routine organization (corresponding to looking at things from MIN's and MAX's point of view alternately) to a simple recursive call with α and β swapped and their signs reversed on each call (corresponding to 'flipping the board').

D: Good idea: In fact, if α and β are set so close together that the search is guaranteed to fail (but return a bound on the possible solution) it's called the Scout algorithm, due to Judea Pearl. Combined with NegaMax it yields NegaScout, which is used by Deep Blue. The trick is that the $\alpha - \beta$ algorithm returns a "best value so far" that gives information about bounds. Starting out with tighter bounds means that there will be more pruning, but still information will be returned. If everything is pruned away, then the interval (called an "aspiration window") must be widened and the search performed again. Scout is really only effective if the nodes are explored in order of their goodness, as when one gets the maximum bang out of $\alpha - \beta$.

Problem 4: Bayes Nets

Fig. 1 shows the canonical textbook Bayes Net.

A: What is the probability that D is true given E is true?

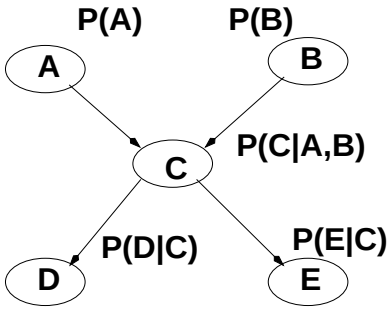


Figure 1: Standard Bayes Net example.

B: A program to produce such a Bayes net would be given nodes in some order. Considering only the predecessor nodes (those earlier in the node order, which it has seen already), it must choose parents for the current node N such that N is conditionally independent of its predecessors in the node ordering, given those parents. The tree in Fig. 1 came from submitting A, B, C, D, E in that order to such an algorithm. In Topsy-Turvy Land, inhabitants tend to think in terms of diagnostic rather than causal evidence, and so they submit nodes in this order: E, D, C, B, A. What Bayes net structure results? To get started: E has no predecessors, hence no parents. From Fig. 1 we see that if E is true that has something to do with C being true, which in turn affects the probability of D being true, so D needs E as parent. (Of course the algorithm does not have Fig. 1 to work from, but it must know the dependencies reflected in Fig. 1 to do its job, so you can safely *pretend* Fig. 1 is available.) You take it from here.

C: A "Direct Sampling" process computes the joint probability density function of $P(A, B, C, D, E)$ in such a network when there is no associated evidence. What is the direct sampling algorithm?

D: Suppose we want to use sampling but also have some evidence event e with which our samples must be consistent. This leads to "rejection sampling" to compute the estimated $P(X = x \mid e)$: describe the rejection sampling algorithm.

Answer Problem 4: Bayes Nets

A:

$$P(d \mid e) = \alpha P(d, e) = \alpha \sum_{A,B,C} P(A, B, C, d, e)$$

$$= \alpha \sum_{A,B,C} \prod P(x_i \mid parent(x_i) = \alpha \sum_a \sum_b \sum_c P(a)P(b)P(c \mid a, b)P(d \mid c)P(e \mid c)$$

B: Fig. 2. See Russell and Norvig pp 496-498.

C: generate samples from the joint distribution as follows, until you have enough to have confidence you know the joint distribution as accurately as you want. Randomly pick a value for root nodes from their prior distributions, then compute the values of their children from those values and the connecting CPTs. Repeat recursively until all random variables have values. Add sample in to joint PDF totals and iterate.

D: Just do direct sampling but toss out any samples with random variable values inconsistent with known evidence. This exacerbates the problem of needing lots of samples to get at unlikely events.

Problem 5 and Embedded Answers: Neural Nets

Consider a multilayer perceptron with a single hidden layer: each output node z_j reports an activation

$$z_j = g(a_j)$$

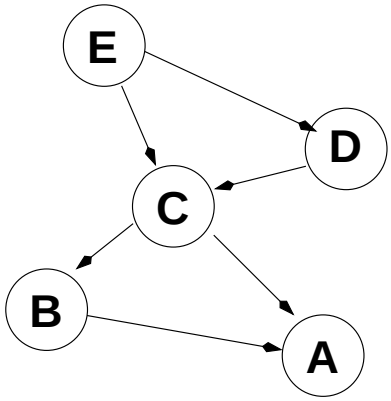


Figure 2: With this order we need more links, hence more CPT numbers.

where g is some function of the weighted sum of inputs from the hidden layer:

$$a_j = \sum_k w_{jk} z_k$$

Each hidden node depends similarly on the input nodes z_i:

$$z_k = g(a_k)$$

$$a_k = \sum_i w_{ki} z_i$$

We wish to compute the update to a weight w_{ki} from input node i to hidden node k given the error at output node j. Beginning with the chain rule

$$\frac{\partial E_j^n}{\partial w_{ki}} = \frac{\partial E_j^n}{\partial a_j} \frac{\partial a_j}{\partial w_{ki}}$$

let us assume we have already computed a value:

$$\delta_j = \frac{\partial E_j^n}{\partial a_j}$$

A: Show that the update can be expressed as a product of the input node value and a function of *only* the hidden node:

$$\frac{\partial E_j^n}{\partial w_{ki}} = \delta_k z_i$$

by deriving a formula for δ_k.

Answer:

$$\begin{aligned}
\frac{\partial a_j}{\partial w_{ki}} &= \frac{\partial}{\partial w_{ki}} \sum_{k'} z_{k'} w_{jk'} \\
&= w_{jk} \frac{\partial}{\partial w_{ki}} z_k \\
&= w_{jk} \frac{\partial}{\partial w_{ki}} g\left(\sum_{i'} w_{ki'} z_i \right) \\
&= w_{jk}\, g'(a_k)\, \frac{\partial}{\partial w_{ki}} \sum_{i'} w_{ki'} z_i \\
&= w_{jk}\, g'(a_k)\, z_i \\
\delta_k &= \delta_j w_{jk} g'(a_k)
\end{aligned}$$

B: The derivation above only uses one node from the output layaer. Supposing the network output and the desired labels from the training data are vectors, how do you train the net?

Answer: Assuming an additive error function $E^n = \sum_j E_j^n$,

$$\begin{aligned}
\frac{\partial E^n}{\partial w_{ki}} &= \sum_j \frac{\partial E_j^n}{\partial w_{ki}} \\
\delta_k &= \sum_j \delta_j w_{jk} g'(a_k)
\end{aligned}$$

C: Describe in words the difference between online and batch training for neural networks.

Answer: Online: adjust weights after each data item. Batch: accu mulate updates over entire dataset. Both: iterate over data until sati sfied.

D: What factors are relevant in choosing a good activation function g?

Answer: Monotonic, easily differentiable, probably should be boun ded from above and below.

Problem 6 and Embedded Answer: Probability

A: Show that $E[(X - E[X])^2] = E[X^2] - (E[X])^2$, with $E(.)$ the expectation (expected-value) function.

Answer:

$$\begin{aligned}
E[(X - E[X])^2] &= E\left[X^2 - 2XE[X] + (E[X])^2\right] \\
&= E[X^2] - 2E[X]E[X] + (E[X])^2 \\
&= E[X^2] - (E[X])^2
\end{aligned}$$

B: The above quantity is called $\mathrm{Var}(X)$. Give an example of two random variables X and Y such that $\mathrm{Var}(X + Y) \neq \mathrm{Var}(X) + \mathrm{Var}(Y)$.

Answer: $X = Y$, X not constant

C: Give an example of two random variables X and Y such that $\mathrm{Var}(X+Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$.

Answer: $P(X,Y) = P(X)P(Y)$

D: Prove

$$P(X, Y \mid e) = P(X \mid Y, e)P(Y \mid e).$$

Answer:

Following is equal to above term by term, and LHS = RHS, QED.

$$\frac{P(X,Y,e)}{P(e)} = \frac{P(X,Y,e)}{P(Y,e)}\frac{P(Y,e)}{P(e)}.$$

E: Prove

$$P(A \mid B, C) = \frac{P(B \mid A, C)P(A \mid C)}{P(B \mid C)}$$

Answer:

$$LHS = \frac{P(A,B,C)}{P(B,C)}.$$

$$RHS = \frac{P(B \mid A, C)P(A \mid C)P(C)}{P(B \mid C)P(C)}$$

$$= \frac{P(B \mid A, C)P(A,C)}{P(B,C)} = LHS,$$

QED.

Problem 7. Default Reasoning

A: One of the earliest proposals for implementing non-monotonic reasoning was within the framework of inheritance networks. In addition to the "subset" links and "element-of" links customary in inheritance networks, these contained default links and negated default links. Illustrate the representations of the classical "Tweety" and "Nixon diamond" examples as default inheritance networks (i.e., birds generally fly, penguins generally don't, and Tweety is a bird, specifically a penguin; and Quakers are generally pacifists, Republicans are generally not pacifists, and Nixon is a Quaker and a Republican). Explain your notation, especially what your link types or labels mean. Then explain what difficulties are encountered in attempting to use such networks in a consistent way, and say what sorts of approaches have been proposed for addressing the difficulties. (Mention of "credulous" and "skeptical" inheritance might be appropriate.)

B: Default Logic (DL) puts the content into inference rules: a new class of rules is introduced:

$$\frac{A : MB}{C},$$

or "if A is provable and it is consistent to assume B then conclude C". Applying these rules leads to "extensions" of our knowledge – though they can also lead to inconsistency.

Give an example of defaults that will create an inconsistency.

Then consider the following defaults and fact:

$$\frac{BaseballPlayer(x) : MHeight(x, 6-1)}{Height(x, 6-1)}$$

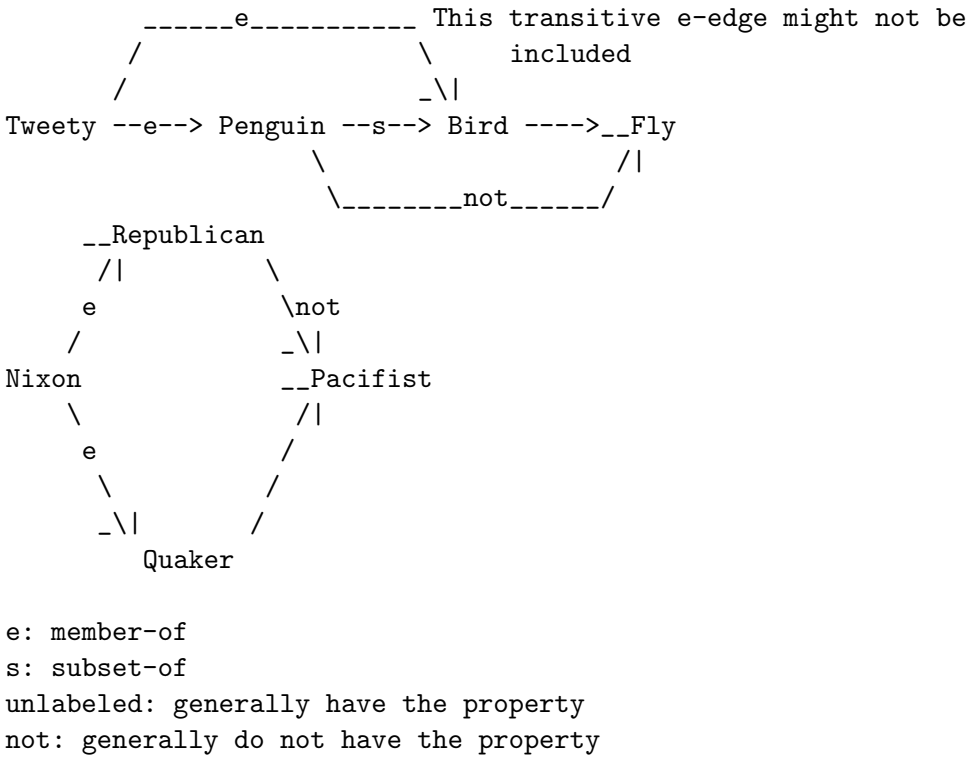
$$Pitcher(Dizzy)$$

and assume that some reasonable ontological hierarchy is given as well. Explain (on general principles) why the given defaults will not lead to inconsistency.

However, we do get multiple extensions; explain. How could they be avoided in the example?

Other problems with default logic concern (i) computation with the M operator, and (ii) the status of defaults as "assertable knowledge"; explain.

Answer Problem 7: Default Reasoning



If we sanction transitive inference we can conclude defeasibly that Tweety doesn't fly (via the lower path) or that he does (via the middle or upper path). Similarly in the Nixon diamond the upper path sanctions the defeasible conclusion that Nixon is a pacifist, while the lower path sanctions the opposite defeasible conclusion. The "credulous" attitude always sanctions as many conclusions as possible, and would thus arbitrarily pick one or the other possible conclusion in the two examples. The "skeptical" attitude refuses to commit to a default conclusion if it conflicts with another default conclusion. Thus no conclusion about flying or pacifism would be reached in the examples.

More sophisticated methods try to find a balance between the two attitudes. In particular, one principle is that more specific information wins over less specific information. Thus the lower pathway in the Tweety example "wins" – Tweety doesn't fly (as intuitions demands). Another principle is that a pathway providing a default conclusion may be neutralized by another pathway that defeats one of the nodes in

the pathway, so that the conclusion of the defeated pathway can no longer conflict with the conclusion of another (otherwise undefeated) one. These methods still leave the question of Nixon's pacifism open, i.e., they remain skeptical of both conclusions because they conflict.

B. An example of defaults that create an inconsistency is

$$\frac{True : MB}{\neg B}, \quad \frac{True : M\neg B}{B},$$

where *True* is the always-true formula (or we could simply not have a premise at all).

The given "baseball defaults" don't lead to inconsistency because they are *normal* defaults, i.e., the formula in the consistency test is the same as the conclusion, and such defaults are known not to introduce inconsistency into a consistent KB.

We get multiple extensions (much as in the A-examples) because, assuming the ontology implies Dizzy is an adult male and a baseball player, we can conclude from the first default that he is of height 6-1, after which we can no longer apply the second default (assuming that the KB also implies that being of height 6-1 is incompatible with being of height 5-10). Conversely, if we apply the second rule first, we conclude Dizzy is of height 5-10. One way to avoid the latter extension would be to require more specific rules to be applied first (again much as in A); another would be to refine the second rule to say

$$\frac{Adult - Male(x) : M[Height(x, 5-10) \wedge \neg Basebell - Player(x)]}{Height(x, 5-10}$$

which would make it inapplicable to Dizzy.

A computational problem with the M operator is that consistency is undecidable in FOL, so the consistency check is potentially nonterminating. A problem with the status of defaults is that claims like "*Baseball players are normally 6-1 in height*" are treated as sentences that can be true of false in ordinary language, but defaults are rules, and therefore cannot be treated as true or false, and cannot be inferred. For instance, we cannot model the inference "*Baseball players are normally 6-1 in height, and soccer players are normally of the same height as baseball players; therefore, soccer players are normally 6-1 in height.*"