b2nd Lt David Crow

ENG/20M

Assignment #3 – Knowledge Representation and Planning

Artificial Intelligence - CSCE 523

Turnin: E-mail me a zip file containing your typed solutions to and associated files for all questions.

1. (5 points) What logic rule did the Cheshire Cat use in this argument, and is it sound?

"To begin with," said the Cat, "a dog's not mad. You grant that?"  
"I suppose so," said Alice.  
"Well, then," the Cat went on, "you see a dog growls when it's angry, and wags its tail when it's pleased. Now I growl when I'm pleased and wag my tail when I'm angry. Therefore I'm mad."

Effectively, the Cheshire Cat said the following:

* If an animal growls when it’s angry () and wags its tail when it’s pleased (), then the animal is not mad ().
* I growl when I’m pleased () and wag my tail when I’m angry ().
* Therefore, I am mad ().

Logically, this looks like this:

I suppose if I had to pick a logic rule, I’d claim that the Cheshire tried to apply a weird blend of Modus Ponens and Modus Tollens. His logic isn’t sound, though, because the following is not a valid logical argument.

There are two problems here. First, note that the negation of is not equivalent to . Second – even if we assume the negation is correct – we are *denying the antecedent*. In other words, does not provide sufficient information to conclude . For both of the stated reasons, this argument is not sound.

1. (5 points) Translate the following Lewis Carroll sentences into a Propositional Logic Knowledge Base and derive two statements from the Knowledge Base:

All hummingbirds are richly colored.  
No large birds live on honey.  
Birds that do not live on honey are dull in color.

We can write the above statements as the following propositional logic statements (respectively):

* .

We can now derive the following two statements (among others) from the knowledge base:

* , which we derive from and via transitivity
* , which we derive from and via transitivity

1. (5 points) Translate the following into First Order Logic and then convert to Conjunctive Normal Form (CNF):

According to the Pidgeon: If little girls eat eggs, then they are a kind of serpent. Alice (who is a little girl) eats eggs. Therefore, she is a kind of serpent.

1. given
2. given
3. given
4. given

Note that (2), (3), and (4) are already in CNF. We can now convert (1) into CNF:

1. [1] UI
2. [5] Implication
3. [6] De Morgan’s
4. [7] Associativity
5. (10 points) Determine, for the following pairs of sentences, if they can be unified and, if they can, give the most general unifier; if not, discuss why.

We can unify these. Here’s our unifier:

This yields the following sentence:

We can unify these. Here’s our unifier:

This yields the following sentence:

To unify and , we must use to find . Although this is nonsensical, it’s logically sound.

To unify and , we must use to find .

However, we can’t unify and . We’ve already shown that must be replaced with and that must be replaced with . If we follow those replacements, though, we find because . Thus, these sentences can’t be unified.

1. (15 points) Use resolution with refutation to show that the following three queries can be inferred from the given knowledge base. At each resolution step also indicate the corresponding identifier and binding list.

KB: 1:

2:

3:

4:

5:

6:

implication

7:

implication

8:

implication

De Morgan’s

associativity

9:

implication

De Morgan’s

associativity

10:

implication

11:

implication

De Morgan’s

associativity

12:

implication

De Morgan’s

associativity

13:

implication

14: [negated goal]

15: [4, 7]

16: [10, 15]

17: [1, 9, 16]

18: [2, 11, 17]

19: [14, 18]

20: [negated goal]

21: [1, 12, 18]

22: [20, 21]

Here, the negated goal is . We convert to CNF in the following way:

existential instantation

De Morgan’s

23: [negated goal]

24: [5, 7]

25: [10, 24]

26: [1, 9, 24]

27: [23, 25, 26]

28: [15, 17, 23]

1. (10 points) Translate the knowledge base from problem 5 into a formula list for Otter and use it to perform a proof by refutation for the queries from problem 4 and for the two below. A copy of the Otter executable and documentation can be found in the course directory. If you want to run Otter on a non-Windows computer, you can access the information you will need at <http://www-unix.mcs.anl.gov/AR/otter/>. If a sentence cannot be proved, determine what needs to be added to the knowledge base to make it provable, and determine whether this invalidates the KB. Turn-in the Otter files and a copy of the screen output for each query.

Otter cannot prove this as-is. However, we can add the following statements to prove that Bill is the brother of Amy:

These statements do not invalidate the knowledge base.

Otter proved parts b, c, d, and e.

1. (25 points) For the following logic problem, a) encode the problem and have Otter solve it, and b) represent the problem as a constraint satisfaction problem and solve using the backtracking algorithm with forward checking.

Link, Zelda, and Ganondorf fought three different evil creatures: the Octorock, an Iron Knuckle, and a Poe. They fought them with three different weapons: a bow and arrows, magic, and a sword. Determine who fought what creature and with what weapon.

1. Ganondorf did not fight the Octorock.
2. The Iron Knuckle was not fought against with magic.
3. Zelda fought the Octorock.
4. Link has a sword and did not fight the Poe.
5. Zelda fought with the bow and arrows.

Otter proved this constraint satisfaction problem. Results can be found in the PROOF sections of “otter-7-out-link.txt,” “otter-7-out-zelda.txt,” and “otter-7-out-ganondorf.txt.”

For the backtracking algorithm with forward-checking, I label Link, Zelda, and Ganondorf with their names. I label the enemies and weapons with the first letter of their names. I label the constraint violations with the number corresponding to the

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Link | | | | | | Zelda | | | | | | Ganondorf | | | | | | Backtracking |
| O | I | P | B | M | S | O | I | P | B | M | S | O | I | P | B | M | S |
| Link: O | O |  |  |  |  |  | X |  |  |  |  |  | X |  |  |  |  |  | Constraint 3 |
| Link: I | X | O |  |  |  |  |  | X |  |  |  |  |  | X |  |  |  |  |  |
| Link: B | X | O |  | O |  |  |  | X |  | X |  |  |  | X |  | X |  |  | Constraint 4, 5 |
| Link: M | X | O |  | X | O |  |  | X |  |  | X |  |  | X |  |  | X |  | Constraint 2, 4 |
| Link: S | X | O |  | X | X | O |  | X |  |  |  | X |  | X |  |  |  | X |  |
| Zelda: O | X | O |  | X | X | O | O | X |  |  |  | X | X | X |  |  |  | X |  |
| Zelda: B | X | O |  | X | X | O | O | X |  | O |  | X | X | X |  | X |  | X |  |
| Ganondorf: P | X | O | X | X | X | O | O | X | X | O |  | X | X | X | O | X |  | X |  |
| Ganondorf: M | X | O | X | X | X | O | O | X | X | O | X | X | X | X | O | X | O | X |  |

1. (25 points) Use the Graphplan planning algorithm to solve the blocks world planning problem shown in Figure 1 and the Rush Hour problems in Figures 2 and 3. The executable, fact, and operator files for the blocks domain are in the course directory. You must modify the fact file to solve blocks world planning problem shown in Figure 1. And write your own fact file and operator file for the two Rush Hour problems shown in Figures 2 and 3. Assume there are no trucks – only cars of length two. Define actions as a movement of a car one square north, south, east, or west depending on orientation. During execution of Graphplan, respond to the prompts to perform automatic time steps, and information, and for other hit ‘B’ for build until goals. Note your operator file (for Rush Hour) should be the same for both problems. The only part that should be different is the initial condition in the fact file.

Turn in a copy of your domain files and a printout of the solutions the planner found. For the Rush Hour problems, also include a report on the search time required and a comparison on the ease of use between Graphplan and the search you implemented in Assignment 1. Which makes the most sense for this domain? And why?

Problem coding recommendations: The ops file will be fairly short; the work is in getting the fact file correct. For the facts create a car object for each car, and location for each grid square. Your preconditions should label whether vehicles can go horizontally or vertically, where the nose and tail of the car are located on the board, which squares are free, and what locations are horizontally and vertically adjacent to each other.



Figure 1: Block World planning problem



Figure 2: Initial state of Rush Hour Problem 1



Figure 3: Initial state of Rush Hour Problem 2