Assignment 3 – Parth Satra (200062999/pasatra)

1. **[30 points] Consider the following English sentences.**

- **John is a entrepreneur.**
- **Entrepreneurs are wealthy.**
- **Wealthy people have big cars.**
- **Big cars are a lot of work to maintain.**

a. [12 points] Convert the sentences into first order predicate logic. Use the following lexicon:

Answer

- $ent(john)$
- $\forall x\ ent(x) \rightarrow wealthy(x)$
- $\forall x\ wealthy(x) \rightarrow car(car-of(x), x) \wedge big(car-of(x))$
- $\forall x\ (\exists y\ car(x, y)) \wedge big(x) \rightarrow work(x)$

b. **[9 points] Convert the logic statements into CNF.**

Answer

Conjunctive normal forms

- $ent(john)$
- $\neg ent(x) \vee wealthy(x)$
- $\neg wealthy(x) \vee car(car-of(x), x)$
- $\neg wealthy(x) \vee big(car-of(x))$
- $\neg car(x, y) \vee \neg big(x) \vee work(x)$

c. **[9 points] Using resolution, prove that "John's car is a lot of work to maintain." HINT: A proof using the four-part heuristic takes about six steps. There may be shorter proofs that do not use the heuristic.**

Answer

To Prove : $work(car-of(john))$

Negation : $\neg work(car-of(john))$,

Resolution:

1. $ent(john)$
2. $\neg ent(x) \vee wealthy(x)$
3. $\neg wealthy(x) \vee car(car-of(x), x)$
4. $\neg wealthy(x) \vee big(car-of(x))$
5. $\neg car(x, y) \vee \neg big(x) \vee work(x)$
6. $\neg work(car-of(john))$
7. $\neg car(car-of(john), y) \vee \neg big(car-of(john))$ From 6+5 {x/car-of(john)}
8. $\neg wealthy(john) \vee \neg big(car-of(john))$ From 7+3 {x/john}
9. $\neg ent(john) \vee \neg big(car-of(john))$ From 8+2 {x/john}
10. $\neg big(car-of(john))$ From 9+1 {}
11. $\neg wealthy(john)$ From 10+4 {x/john}
12. $\neg ent(john)$ From 11+2 {x/john}
13. ■ From 12+1 {}

2. **[30 points] Here's a partial PDDL description of a simplified planning problem for a trucking company moving cargo around the country. (The PDDL is partial because the goal is omitted.) t1 is a truck and c1 is an item of cargo. Assume ny and ral are the only two cities.**

Init(at(t1, ny) ∧ sit(c1, ny) ∧ city(ny) ∧ city(ral))
action(load(C, T, A),
PRECOND: sit(C,A) ∧ at(T,A)
EFFECT: ¬sit(C,A) ∧ in(C,T))
action(unload(C, T, A),
PRECOND: at(T,A) ∧ in(C,T)),
EFFECT: sit(C,A) ∧ ¬in(C,T))
action(drive(T, From, To),
PRECOND: city(From) ∧ city(To) ∧ at(T, From)
EFFECT: ¬at(T, From) ∧ at(T, To))

a. [15 points] Draw the first five levels of a plan graph for this problem -- initial state S_1, first action level A_1, second state level S_2, second action label A_2, third state level S_3. Draw the edges between the levels to connect literals and actions. Assume the goal (whatever it is) is not contained in S_1, S_2, or S_3. Don't forget, you will need a negated literal for every ground literal whose predicate symbol and arguments are in the lexicon but not in the initial state description. And don't forget the continuation actions.

Answer
The solution to this part can be found in the file named as '*PDDL_Assignment3.svg*'. You can open the file in Google chrome to see the answer.

b. [15 points] Identify with curved vertical edges the mutexes in this part of the graph. Label the mutexes with their type (NL, IE, I, IS, CN). Look for them in that heuristic order. For each pair, one type of mutex is enough; things are either mutex or not, and we don't need multiple reasons. Some mutex types don't occur at all.

Answer
The solution to this part can be found in the file named as '*PDDL_Mutex_Assignment3.pdf*'.

3. **[40 points] Here is the diabetic semantic network as shown in Lectures:**
Here is a Prolog program to encode this network and answer queries about it.
Re-implement this network in Java or C++. Allow for queries of the same sort as Prolog allows.

Answer
The solution to this is found in the file '*PrologSimulator.java*'. To run the program perform the following steps:-

i) javac PrologSimulator.java (Compile the program).
ii) java PrologSimulator (Execute the program).

Note: Type 'exit' to exit the program. Also all the answers from program come in one go, we don't need to type ';' to get all outputs.

4. **[30 points] (Bonus question) Here is a database of facts and rules. Write and consult a simple Prolog database which contains facts and rules representing this information.**

Creatures come in two types: humans and birds. One type of human is a man. One type of bird is a penguin. Freddie is a man. Arnie is a man. Lewis is a penguin.

a. (5 pts.) Draw this taxonomy as a graph, with ``creature'' at the root, and label the edges with AKO or ISA, whichever is appropriate.

Answer
The taxonomy graph is attached in the submission with the name '*Taxonomy_Graph.pdf'.* The graph highlighted in red the solution for the [c] part of this question.

b. **[10 points] Suppose these facts were represented by seven FOPL facts of the form edge(, ,). (Assume these facts are implemented; you don't have to write them.)**

Using as a toplevel rule head the syntax rel(SourceNode, RelationshipType, DestinationNode) and any other predicates you need, write a set of one or more rules to allow the inference that:

1. **Freddie is a man, Freddie is a human, and Freddie is a creature.**
2. **Arnie is a man, Arnie is a human, and Arnie is a creature.**
3. **Lewis is a penguin, Lewis is a bird, and Lewis is a creature.**

Your rules don't have to follow strict Prolog syntax, but it's probably easier if they do.

Answer
Considering the facts are as follows :-
edge(freddie, isa, man).
edge(arnie, isa, man).
edge(man, ako, human).
edge(human, ako, creature).
edge(lewis, isa, penguin).
edge(penguin, ako, bird).
edge(bird, ako, creature).

Inference Rules required to prove the relations are as follows :-
rel(SrcNode, Type, DestNode) :- edge(SrcNode, Type, DestNode).
rel(SrcNode, Type, DestNode) :- edge(SrcNode, isa, Node), rel(Node, ako, DestNode).
rel(SrcNode, Type, DestNode) :- edge(SrcNode, ako, Node), rel(Node, ako, DestNode).

c. **[5 points] Now add nodes and edges to the network to represent the knowledge that humans normally have two legs and birds can normally fly, but Freddie has one leg and penguins cannot fly.**

Using fact syntax such as property(<node>, legs, two) and property(<node>, fly, no), indicate which new facts will be necessary, and show in your network sketch from Part (a) where they should be added.

Answer
The file named as '*Taxonomy_Graph.pdf*' contains the answer to this question. The parts highlighted in red parts are the properties added to this graph.
The required properties can be represented as following facts.

property(Freddie, leg, 1)
property(human, leg, 2)
property(penguin, fly, no)
property(bird, fly, yes)

d. [10 points] Add rule(s) to allow inference that
(i) Lewis cannot fly,
(ii) Arnie has two legs.
(iii) Freddie has one leg.
Your new rules will need to use the new facts from Part (c).

Answer
The rules from part (b) will still remain and some additional rules will be required.
Lets say the predicate '*propertyValue*' is used to evaluate these facts, then corresponding new inference rules will be

```
propertyValue(Node, Property, Value) :- property(Node, Property, Value), !.
propertyValue(Node, Property, Value) :- rel(Node, isa, Node1),
                                        property(Node1, Property, Value).
propertyValue(Node, Property, Value) :- rel(Node, ako, Node1),
                                        property(Node1, Property, Value).
```

So the queries can be as follows:
(i) propertyValue(lewis, fly, no) : You should get answer as true.
(ii) propertyValue(arnie, leg, 2) : You should get answer as true.
(iii) propertyValue(freddie, leg, 1) : You should get answer as true.