Ankit Murarka amurark 200111393

Artificial Intelligence Assignment III

1. Consider the following English sentences.

   - Joe is a lawyer.
   - Lawyers are wealthy.
   - Wealthy people have big houses.
   - Big houses are a lot of work to maintain.

   1. Convert the sentences into first order predicate logic. Use the following lexicon:

```
Constants:  joe
Functions:  house-of(X)       -- X's house
Predicates: law(X)          -- X is a lawyer
            house(X, Y)       -- X is Y's house
            big(X)          -- X is big
            rich(X)      -- X is rich
            work(X)         -- X is a lot of work to maintain
```

⇨ First Order Predicate Logic(FOPL)

- Joe is a lawyer

law(Joe)

- Lawyers

$$\forall X(law(X) => rich(X))$$

- Wealthy People(X) have big houses(Y)

$$\forall X \forall Y\big((rich(X) \wedge house(Y,X)\big) => big(Y))$$

- Big houses are a lot of work to maintain

$$\forall Y\big(big(Y) \wedge \ \exists X\big(house(Y,X)\big) => work(Y)\big)$$

2. Convert the logic statements into CNF. &
3. Using resolution, prove that "Joe's house 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.

Conjunctive Normal Form(CNF)

1. law(joe)
2. $\neg(law(X)) \vee rich(X)$
3. $\neg(rich(X)) \vee \neg(house(Y,X)) \vee big(Y)$
4. $\neg\left(big(Y)\right) \vee \neg(house(Y,X)) \vee work(Y)$ ….Here the existential quantifier gets negated to a universal quantifier.
5. house(house-of(X),X) ….From the axiom

----

6. $\neg work(house-of(Joe))$ ….Negated conclusion

----

7. $\neg big(house-of(Joe)) \vee \neg house(house-of(Joe), X3)$ ⋯6+4 Substituting{house-of(Joe)/Y2}
8. $\neg rich(X2) \vee \neg house(house-of(Joe), X2) \vee \neg house(house-of(Joe), X3)$ ⋯7+3 Substituting { house-of(Joe)/Y1 }
9. $\neg law(Joe) \vee \neg house(house-of(Joe), Joe) \vee \neg house(house-of(Joe), X3)$ ⋯8+2 Substituting {Joe/X1 and Joe/X2}
10. $\neg house(house-of(Joe), Joe) \vee \neg house(house-of(Joe), X3)$ ⋯9+1
11. $\neg house(house-of(Joe), X3)$ ...10+5 Substituting {X4/Joe}
12. □ …11+5 Substituting {Joe/X3}

2. 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.

Solution: List of Mutexes categorized based on layers and types:

Layer: State1
This is the base layer, no mutexes in this layer otherwise we cannot proceed.

Layer: Action1
Mutex: Inconsistent Effect

at(t1,ny), drive(t1, ny, ral)
load(c1, t1, ny), sit(c1, ny)
drive(t1, ny, ral), $\neg at(t1, ral)$
load(c1, ty, ny), $\neg in(c1, t1)$

Mutex: Interference

load(c1, t1, ny) with drive(t1, ny, ral)

Layer: State2
Mutex: Negated Literals

¬at(t1, ny) with at(t1, ny)
sit(c1, ny) with ¬sit(c1, ny)
in(c1, t1) with ¬in(c1, t1)
at(t1, ral) with ¬at(t1, ral)

Mutex: Inconsistent Support

in(c1, t) and sit(c1, ny)
at(t1, ral) and at(t1, ny)
¬at(t1, ny) and ¬at(t1, ral)

Layer: Action2
Mutex: Inconsistent Effect

unload(c1,t1,ny) with load(c1, t1, ny)
unload(c1,t1,ny) with in(c1,t1)

at(t1,ny) with drive(t1,ny,ral)
at(t1,ny) with ¬$at(t1, ny)$
load(c1, t1, ny) with ¬in(c1,t1)
load(c1, t1, ny) with unload(c1,t1,ral)
¬$sit(c1, ny)$ with drive(t1, ral, ny)
drive(t1, ral, ny) with at(t1, ral)

drive(t1, ral, ny) with ¬at(t1, ral)
drive(t1, ral, ny) with drive(t1, ny, ral)
in(c1,t1) with unload(c1, t1, ral)
in(c1, t1) with ¬in(c1, t1)
drive(t1, ny, ral) with ¬at(t1, ral)
at(t1, ral) with ¬at(t1, ral)

Mutex: Interference

unload(c1, t1, ny) and unload(c1, t1, ral)
unload(c1, t1, ral) and drive(t1, ral, ny)
unload(c1, t1, ny) and drive(t1, ny, ral)
drive(t1, ny, ral) and load(c1,t1, ny)

Mutex: Competing Needs

unload(c1, t1, ral) and ¬in(c1, t1)
unload(c1, t1, ral) and ¬at(t1, ral)
unload(c1, t1, ny) and ¬at(t1, ny)
unload(c1, t1, ny) and ¬in(c1, t1)
¬at(t1, ny) and drive(t1, ny, ral)
¬at(t1, ral) and drive(t1, ral, ny)

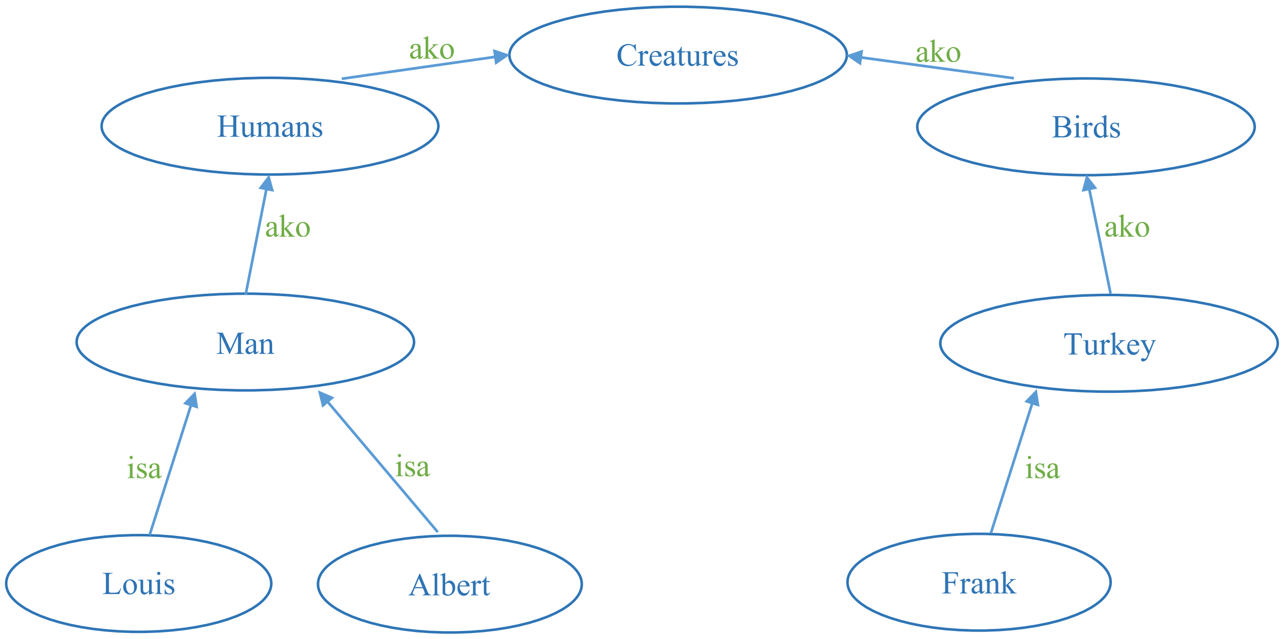
Layer: State3
Mutex: Negated Literals

sit(c1, ral) and ¬sit(c1, ral)
sit(c1, ny) and ¬sit(c1, ny)
in(c1, t1) and ¬in(c1, t1)
¬at(t1, ny) and at(t1, ny)
at(t1, ral) and ¬at(t1, ral)

3. Here is a database of facts and rules. Write a simple Prolog-style 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 turkey.
Louis is a man.
Albert is a man.
Frank is a turkey.
```

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



2. Suppose these facts were represented by seven FOPL facts of the form `edge(<sourceNode>, <linkType>, <destinationNode>)`. Implement these facts as Prolog facts.

   Using as a top-level 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. Louis is a man, Louis is a human, and Louis is a creature.
2. Albert is a man, Albert is a human, and Albert is a creature.
3. Frank is a turkey, Frank is a bird, and Frank is a creature.

Your rules should follow strict Prolog syntax, and should allow inference over hierarchies of **any** depth, not just the depth in this example.

Solution:

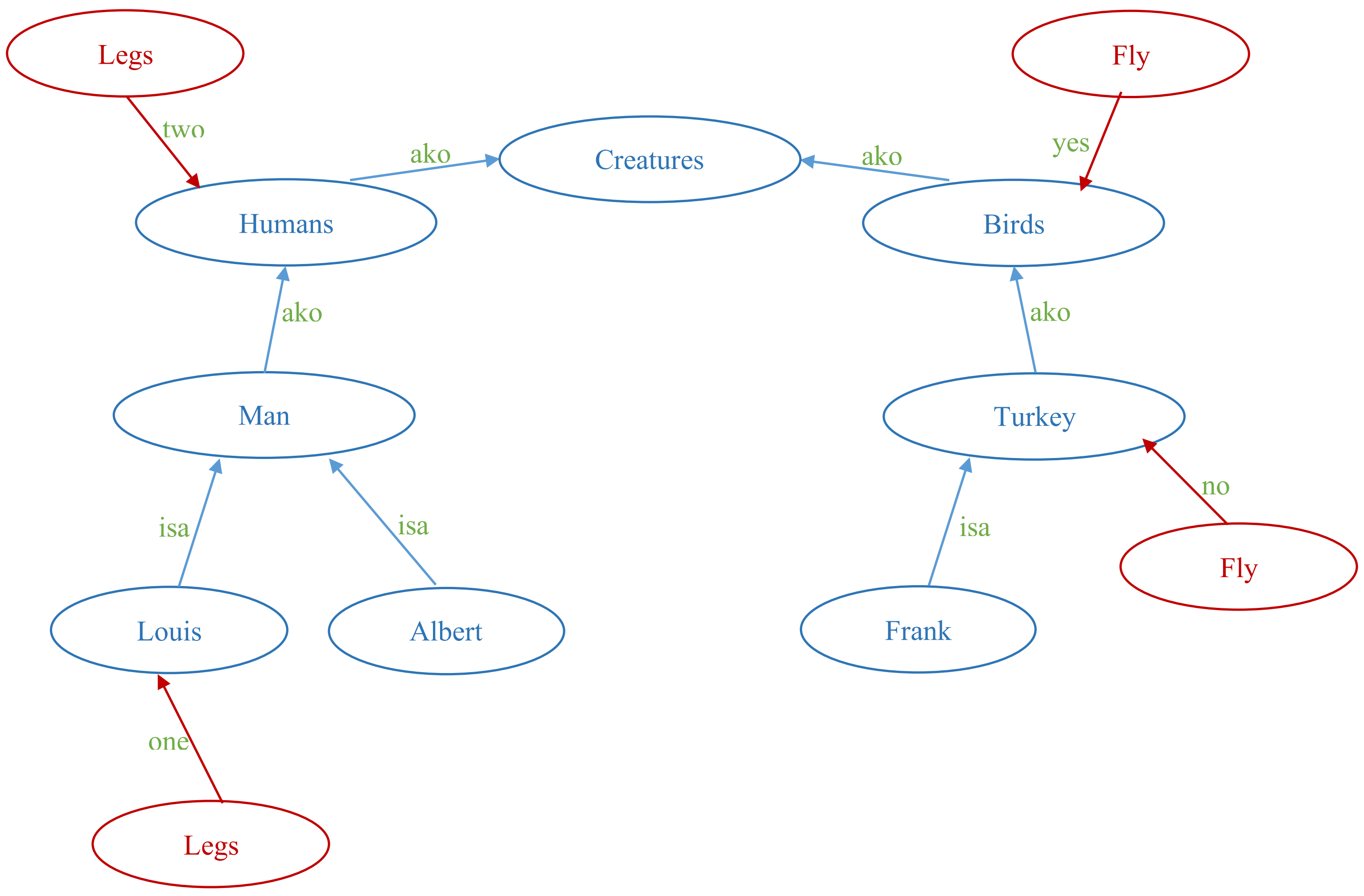
```
edge(bird, ako, creature).
edge(human, ako, creature).
edge(man, ako, human).
edge(turkey, ako, bird).
edge(albert, isa, man).
edge(louis, isa, man).
edge(frank, isa, turkey).
```

```
-   rel(SourceNode, RelationshipType, DestinationNode):-
    edge(SourceNode, RelationshipType, DestinationNode).
-   rel(SourceNode, isa, DestinationNode):-
            rel(SourceNode, isa, TempNode),
            rel(TempNode, ako, DestinationNode),!.


-   rel(SourceNode, RelationshipType, DestinationNode):-
            edge(SourceNode, ako, TempNode),
            rel(TempNode, RelationshipType, DestinationNode).
```

3. Now add nodes and edges to the network to represent the knowledge that humans normally have two legs and birds can normally fly, but Louis has one leg and turkeys 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.

Taxonomy graph indicating property facts



PROLOG CODE

```
property(human, legs, two).
property(bird, fly, yes).
property(louis, legs, one).
property(turkey, fly, no).
```

4. Add rule(s) to allow inference that (i) Frank cannot fly, (ii) Albert has two legs. and (iii) Louis has one leg. Your new rules will need to use the new facts from Part (c).

```
feature(Node, Property, Value):- property(Node, Property, Value).

feature(Node, Property, Value):-
    (property(Node, Property, _) ->
     (property(Node, Property, Value) ->
        true
     ;   false)
    ; rel(Node, RelationshipType, Node1),
     (property(Node1, Property, _) -> !,
       (property(Node1, Property, Value) ->
          true
      ;   false))).
```