Assignment #3 **Due: before 2345 T 17 OCT**

STUDENT NAME: AKSHAY NALWAYA

STUDENT ID: 200159155

The Assignment

1. [60 points] In planning, no-op actions are ones that have one precondition and one effect, where both are the same. In Graphplan, no-op actions are added, at every level, for each proposition that appears in the previous level.

1. [5 points] Explain why no-op actions are needed in Graphplan.

Ans.

One of the properties of planning graphs is that ***"Literals increase monotonically".*** This property will only hold true if, for every state, we carry forward the literals from previous state as it is not possible for all literals to be changed by some action. And hence, this is only possible by using *no-op* or *persistence* actions.
Also, some literals that were not used in current state might be useful for some action in next state and hence it is necessary to have all literals for every state.
Therefore, no-op actions are needed.

2. [5 points] In Graphplan, no-op actions are included when determining mutex relationships. Why is this done? Would Graphplan still work if it did not find those types of mutexes? Why or why not?

Ans.

No, Graphplan would not work if no-op actions are not involved while determining mutex relationships because, for mutexes condition like *"inconsistent effects",* we need to have a track of actions in the previous state and if we remove these actions then it would be a violation of what Graphplan is and its algorithm working. Thus, we would not find the mutex relationships.

3. [10 points] A robot needs to deliver mail in the Triangle between NCSU, UNC, and Duke. Use planning with GraphPlan to help the robot deliver mail to the different mail centers.

```
Init (At(M1, NCSU) ∧ At(M2, UNC) ∧ At(M3, Duke) ∧ Mail(M1) ∧
Mail(M2) ∧ Mail(M3) ∧ MailCenter(NCSU) ∧ MailCenter(UNC) ∧
MailCenter(Duke) ∧ Robot(Fred) ∧ At(Fred, NCSU) ∧
¬GripperFull(Fred))

Goal (At(M1, UNC) ∧ At(M2, Duke) ∧ At(M3, NCSU) ∧ At(Fred, NCSU))
Action (Go(b,s,e),
PRECOND: Robot(b) ∧ At(b, s) ∧ MailCenter(s) ∧ MailCenter(e)
EFFECT: ¬At(b, s) ∧ At(b, e))
```

```
Action(PickUp(b,l,m),
PRECOND: Robot(b) ∧ Mail(m) ∧ MailCenter(l) ∧ At(b, l) ∧ At(m, l) ∧
¬GripperFull(b)
EFFECT: GripperFull(b) ∧ GripperCarries(m) ∧ ¬At(m, l))
```

Given ONLY the predicates given in the other two actions (i.e. DON'T introduce any new predicates), write a schema for a PutDown(b,l,m) action that specifies preconditions and effects. Use the predicates Robot(b), MailCenter(l), and Mail(m) in your solution.

Ans.

SCHEMA FOR ACTION - PUTDOWN(b,l,m)

```
Action(PutDown(b,l,m),
PRECOND: Robot(b) ∧ MailCenter(l) ∧ Mail(m) ∧  At(b,l) ∧ ¬At(m,l) ∧
GripperFull(b) ∧ GripperCarries(m)
EFFECT: ¬GripperFull(b) ∧ ¬GripperCarries(m) ∧ At(m,l))
```

4. [10 points] Given the set of actions, formulate a plan that will reach the goal state from the start state.

Ans.

One plan to solve this problem:

```
[PickUp(Fred,NCSU,M1), Go(Fred,NCSU,UNC), PutDown(Fred,UNC,M1),
PickUp(Fred,UNC,M2), Go(Fred,UNC,Duke), PutDown(Fred,Duke,M2),
PickUp(Fred,Duke,M3), Go(Fred,Duke,NCSU), PutDown(Fred,NCSU,M3)]
```

5. [15 points] Suppose the initial state is ONLY

```
Init(At(M1, NCSU) ∧ At(Fred, NCSU) ∧ Robot(Fred) ∧ Mail(M1) ∧
MailCenter(NCSU) ∧ MailCenter(UNC) ∧ ¬GripperFull(Fred))
```

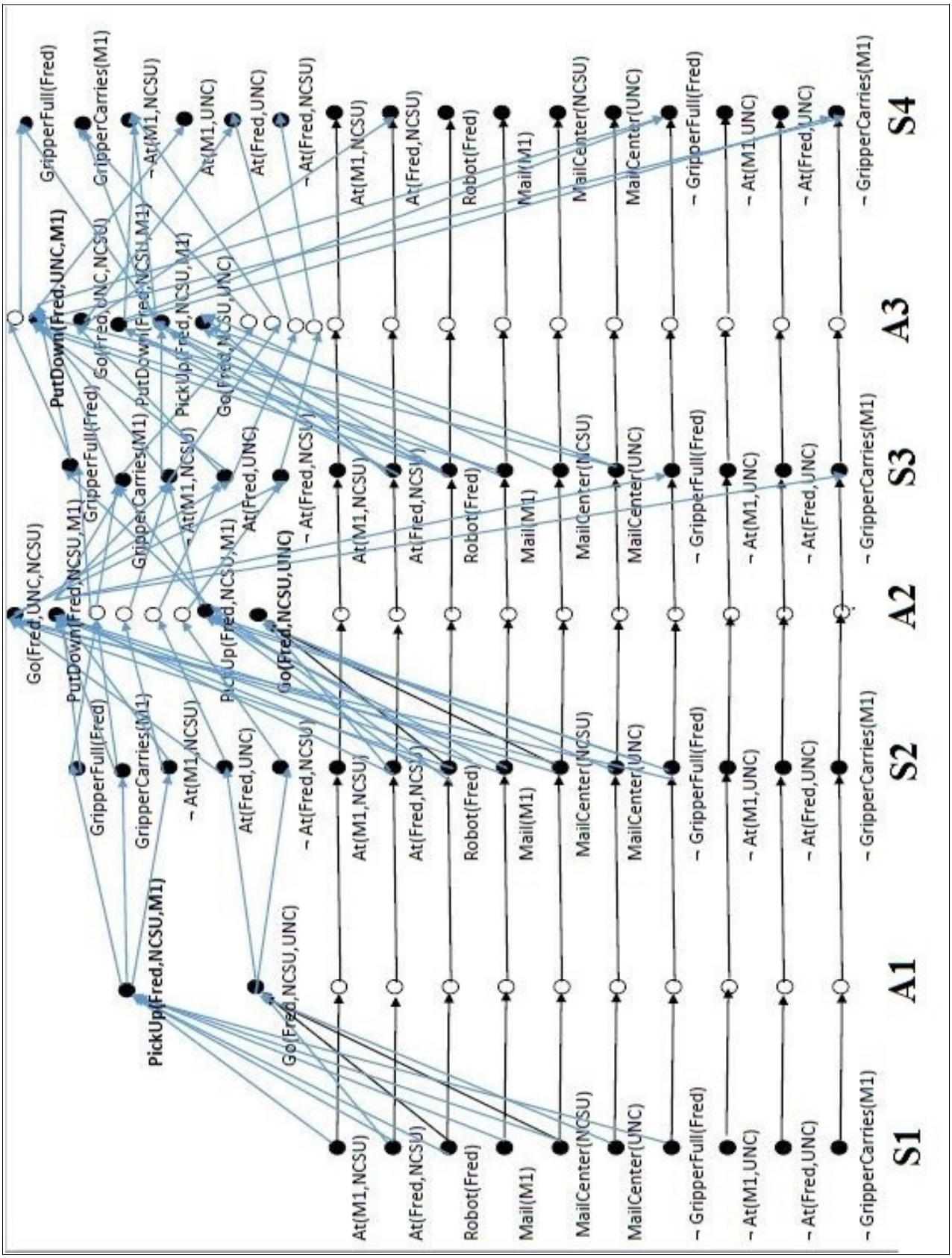
and the goal is ONLY

```
Goal(At(M1, UNC) ∧ At(Fred, UNC))
```

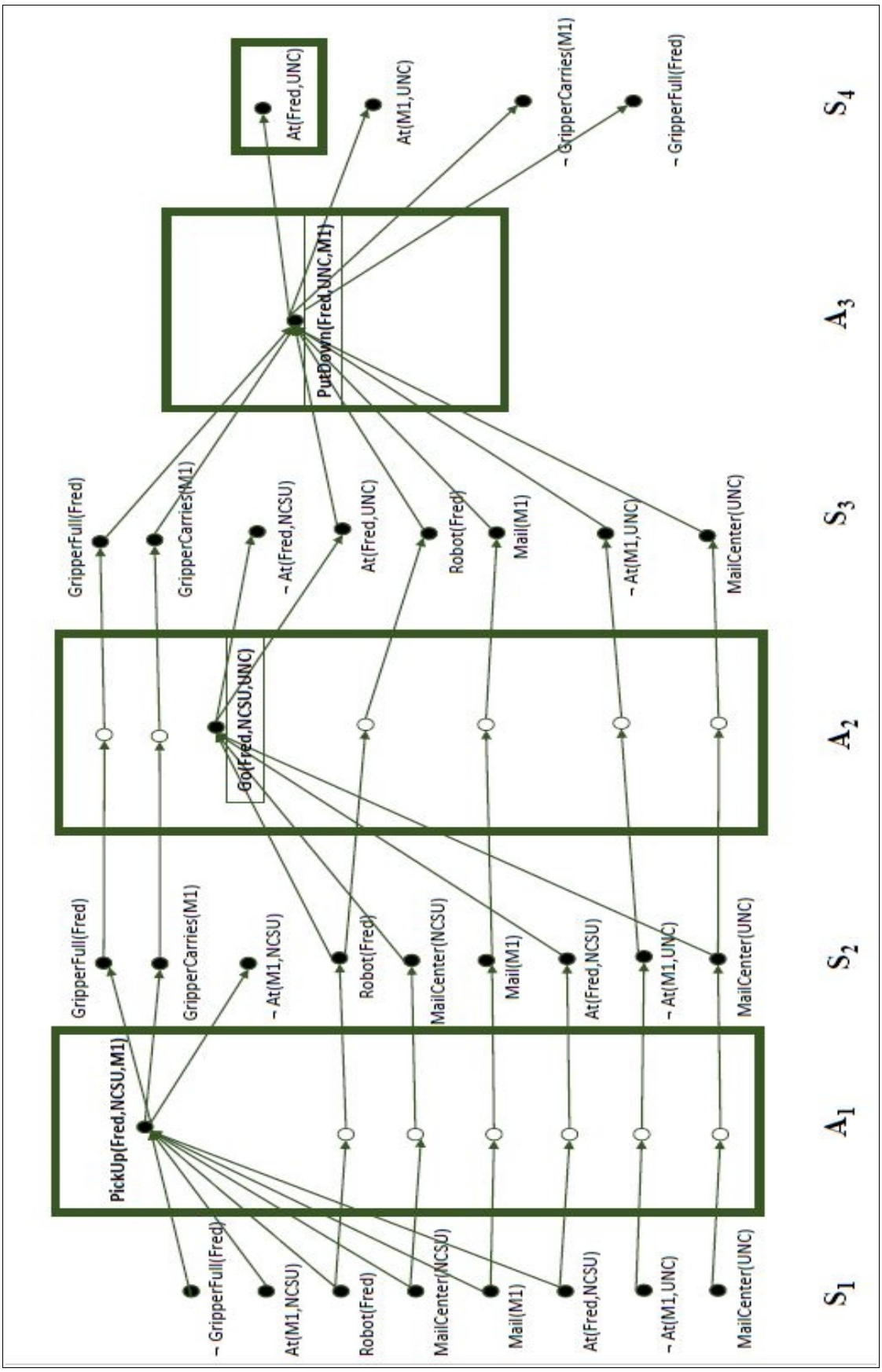
Draw the plan graph of this problem, starting from the initial state, and continue until you first reach the goal state. Then show the solution path. Don't forget, you will need a negated literal for every ground literal whose predicate symbol and arguments are in the lexicon but not in the initial state description.

Ans.

GRAPHPLAN OF PROBLEM:



SOLUTION GRAPH OF THE PROBLEM:



6. [15 points] Identify with curved vertical edges the mutexes in this graph. Label the mutexes with their type (NL, IE, I, CN, IS). 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 may don't occur at all.

Ans.

TYPES OF MUTEXES DETECTED:

Negated Literals (NL) -

- States At(M1,NCSU) and ¬At(M1,NCSU) at S2
- States At(Fred,NCSU) and ¬At(Fred,NCSU) at S2
- States GripperFull(Fred) and ¬GripperFull(Fred) at S2
- States GripperCarries(M1) and ¬GripperCarries(M1) at S2
- States At(Fred,UNC) and ¬At(Fred,UNC) at S3
- States At(M1,NCSU) and ¬At(M1,NCSU) at S3
- States GripperCarries(M1) and ¬GripperCarries(M1) at S3
- States At(Fred,NCSU) and ¬At(Fred,NCSU) at S3
- States GripperFull(Fred) and ¬GripperFull(Fred) at S3
- States At(Fred,UNC) and ¬At(Fred,UNC) at S4
- States At(M1,NCSU) and ¬At(M1,NCSU) at S4
- States GripperCarries(M1) and ¬GripperCarries(M1) at S4
- States At(Fred,NCSU) and ¬At(Fred,NCSU) at S4
- States GripperFull(Fred) and ¬GripperFull(Fred) at S4

Inconsistent Effects (IE) -

- Actions PickUp(Fred,NCSU,M1) and no-op of At(M1,NCSU) at A1
- Actions Go(Fred,NCSU,UNC) and no-op of ¬At(Fred,UNC) at A1
- Actions Go(Fred,NCSU,UNC) and no-op of ¬At(Fred,UNC) at A2
- Actions no-op of At(Fred,NCSU) and no-op of ¬At(Fred,NCSU) at A2
- Actions no-op of GripperCarries(M1) and no-op of ¬GripperCarries(M1) at A2
- Actions no-op of GripperFull(Fred) and PutDown(Fred,NCSU,M1) at A3
- Actions PickUp(Fred,NCSU,M1) and PutDown(Fred,NCSU,M1) at A3
- Actions PickUp(Fred,NCSU,M1) and no-op of At(M1,NCSU) at A3
- Actions Go(Fred,UNC,M1) and no-op of At(Fred,UNC) at A3
- Actions PutDown(Fred,UNC,M1) and no-op of ¬At(M1,UNC) at A3

Interference (I) -

- Actions Go(Fred,NCSU,UNC) and PickUp(Fred,NCSU,M1) at A1
- Actions PutDown(Fred,NCSU,M1) and no-op of ¬At(M1,NCSU) at A2
- Actions PickUp(Fred,NCSU,M1) and no-op of ¬GripperCarries(M1) at A2
- Actions Go(Fred,NCSU,UNC) and no-op of ¬At(Fred,NCSU) at A3
- Actions no-op of GripperFull(Fred) and no-op of ¬GripperFull(Fred) at A3

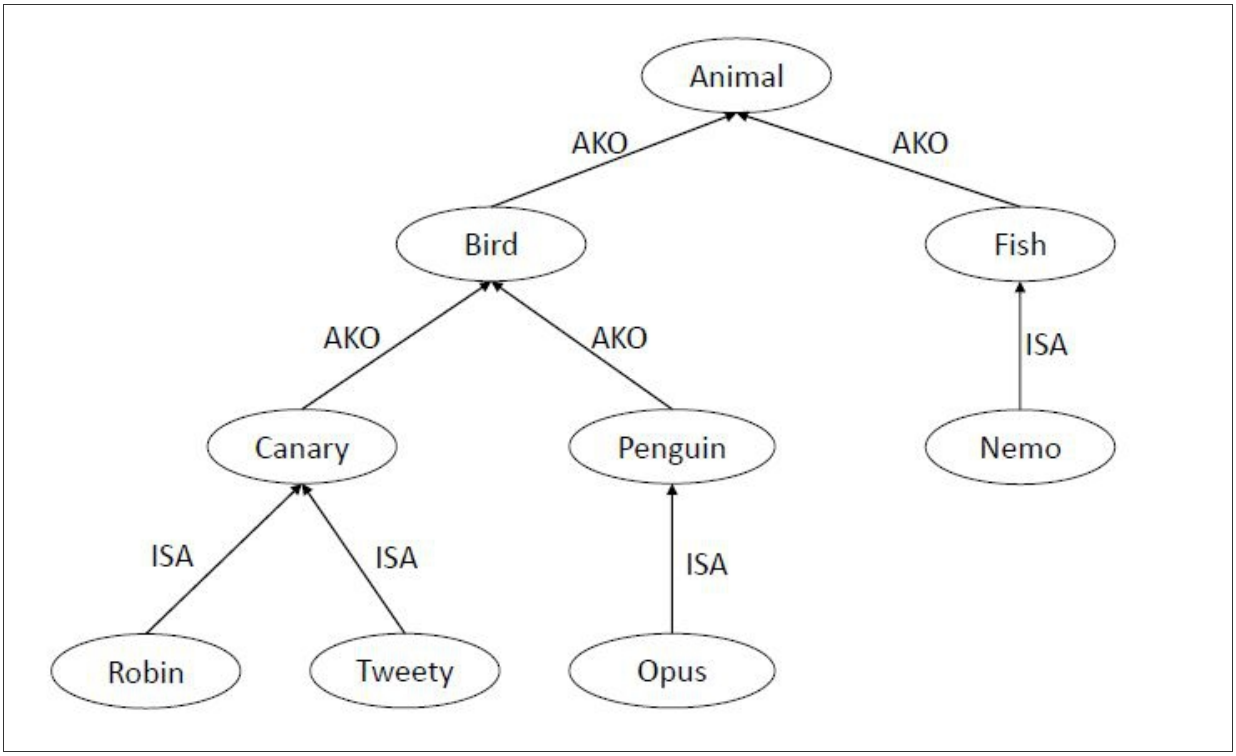
Competing Needs(CN) -

- Actions no-op of At(M1,NCSU) and no-op of ¬At(M1,NCSU) at A2
- Actions no-op of At(Fred,NCSU) and no-op of ¬At(Fred,NCSU) at A2
- Actions no-op of GripperFull(Fred) and no-op of ¬GripperFull(Fred) at A2
- Actions no-op of GripperCarries(M1) and no-op of ¬GripperCarries(M1) at A2
- Actions no-op of At(Fred,UNC) and no-op of ¬At(Fred,UNC) at A3
- Actions no-op of At(Fred,NCSU) and no-op of ¬At(Fred,NCSU) at A3
- Actions no-op of GripperCarries(M1) and no-op of ¬GripperCarries(M1) at A3
- Actions no-op of GripperFull(Fred) and no-op of ¬GripperFull(Fred) at A3
- Actions PickUp(Fred,UNC,M1) and PutDown(Fred,NCSU,M1) at A3

2. [40 points] Here is a database of facts and rules. Write a simple Prolog-style database which contains facts and rules representing this information.

```
Animals are of two types: bird and fish.
One type of bird is penguin.
Another type of bird is canary.
Opus is a penguin.
Tweety is a canary.
Robin is a canary.
Nemo is a fish.
```

1. **[5 points] Draw this taxonomy as a graph, with "animal" at the root, and label the edges with AKO or ISA, whichever is appropriate.**



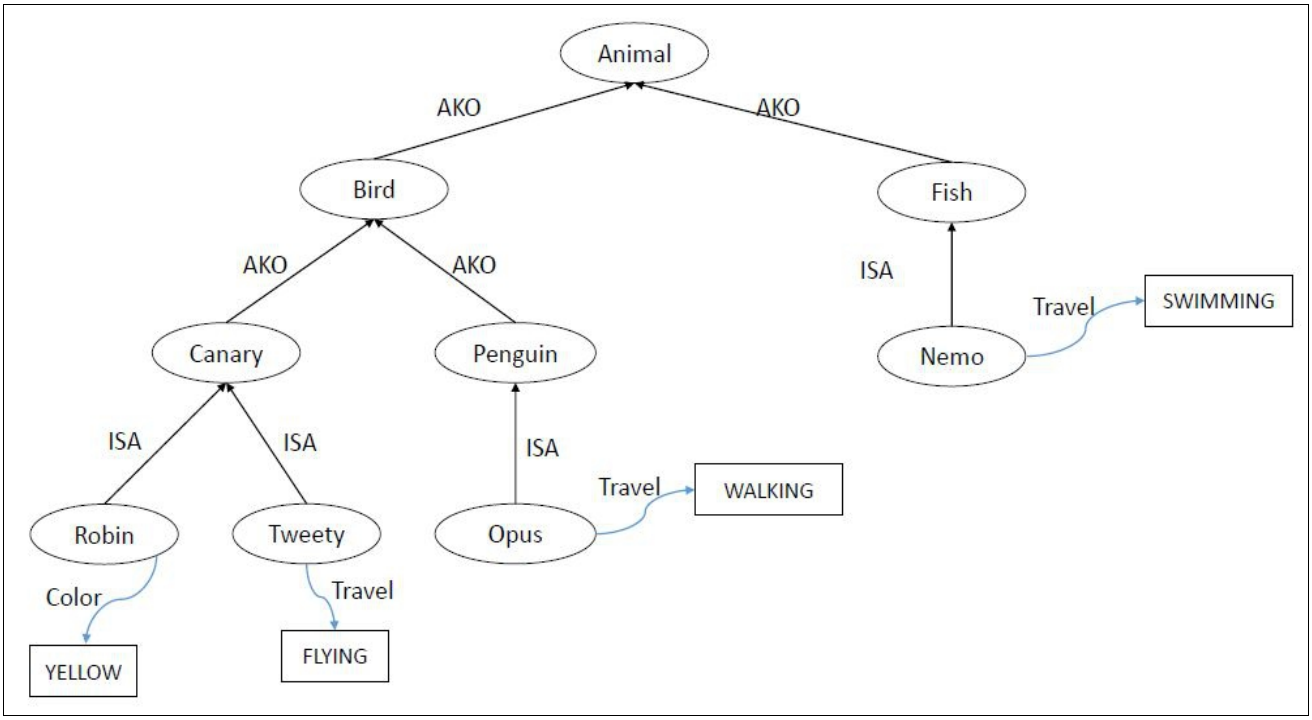
2. **[10 points] Suppose these facts were represented by seven FOPL facts of the form edge(<sourceNode>, <linkType>, <destinationNode>). Implement these facts as Prolog facts.**

   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. **Opus is a penguin, Opus is a bird, and Opus is an animal.**
   2. **Tweety is a canary, Tweety is a bird, and Tweety is an animal.**
   3. **Robin is a canary, Robin is a bird, and Robin is an animal.**
   4. **Nemo is a fish, Nemo is an animal.**

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

3. **[10 points] Now add nodes and edges to the network to represent the knowledge that birds normally travel by flying, and fishes travel by swimming, but penguins travel by walking. Also, a canary is normally yellow, but tweety is white. Using fact syntax such as property(<node>, travel, swimming) and property(<node>, color, yellow), indicate which new facts will be necessary, and show in your network sketch from Part (a) where they should be added.**



4. **[15 points] Add rule(s) to allow inference that (i) Opus travels by walking, (ii) Tweety travels by flying, (iii) Robin is yellow, and (iv) Nemo travels by swimming. Your new rules will need to use the new facts from above.**