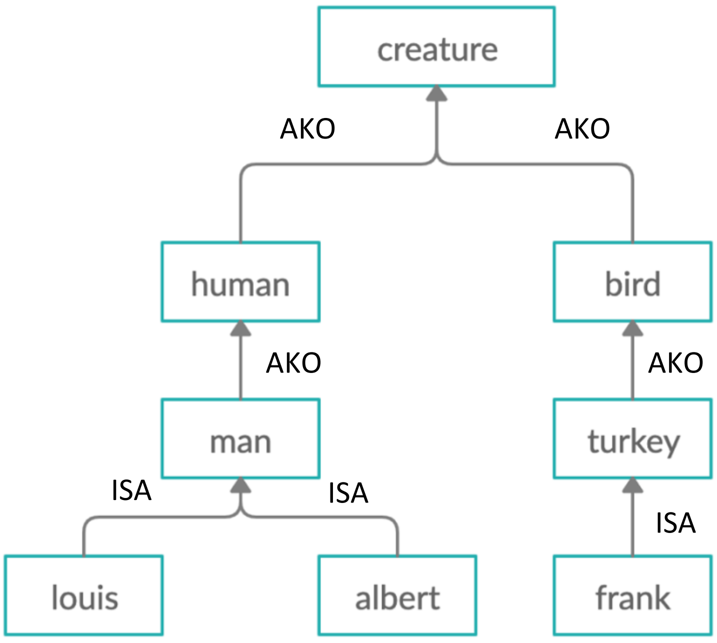
Assignment 4

Question1- 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.

Answer:



2. **Suppose these facts were represented by seven FOPL facts of the form edge(<source Node>, <link Type>, <destination Node>)Implement these facts as Prolog facts. Using as a top level rule head the syntax rel(Source Node, Relationship Type, Destination Node) 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.**

Answer: As given in the question FOPL PROLOG facts are written in the format like edge(<source Node>, <link Type>, <destination Node>). FOPL facts from above questions are written in **Question1.pl** file.
Facts are written like:

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

Rules are written as:

- rel(A, reltype, B):- edge(A, reltype, B).
- rel(A, reltype, B):- edge(A, reltype, C), rel(C, ako, B).
- rel(A, reltype, B):- edge(A, reltype, C), rel(C, isa, B).

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

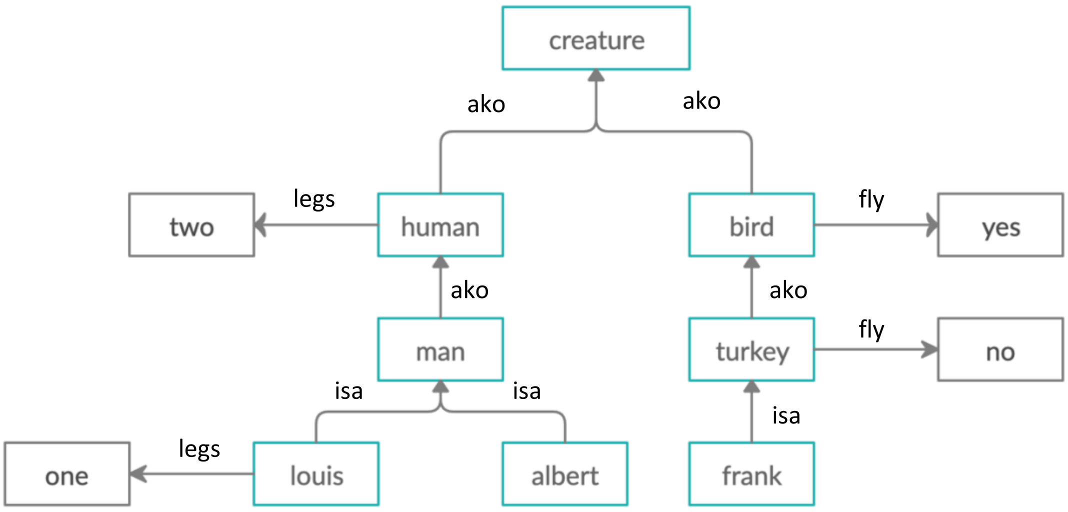
Answer: To allow inference for any depth we will have to use recursion in the rel(Source Node, Relationship Type, Destination Node) rule. It will find all the possible connection to the object been passed in the rel. Example: To find the reference louis isa human when we know louis is a man and man is a kind of a human we can write rel(A, reltype, B):- edge(A, reltype, C), rel(C, ako, B). Where A will be louis and B will be human and reltype will be isa. Similarly I have described rules like: rel(A, reltype, B):- edge(A, reltype, C), rel(C, isa, B) and rel(A, reltype, B):- edge(A, reltype, C). It is in file **Question1.pl**

4. **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.**

Answer: To add the properties to human and bird we can add new FOPL facts in our knowledge base. Using the template given in the question **property(<node>, legs, two)** for human and **property(<node>, fly, yes)** for birds. Below new facts were added:

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

Above new updated knowledge base is in file **Question1.pl**



In above diagram, we have added legs property to human and whichever object is a kind of human it will have the same properties. Like man will inherit properties of human and louis and albert will inherit properties from man. Value of that property can vary from object to object. For example Louis just have one leg and in legs property we have mentioned one, albert will have two legs. Same goes for bird. Birds have property of flying so turkey will also have property of flying but the value of that property will be no as turkey cannot fly and hence frank cannot fly.

5. **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).**

Answer: For inference of properties we added in above part number 4. I have created below rules:
hasPropoerty(A, ppt, val) :- property(Z, ppt, val) : This above rule will check if we are looking for property of human, louis, bird and turkey. Means objects that have a direct property edge to them and they are not depending upon other objects to read the property value.

hasPropoerty(A, ppt, val) :- edge(A, isa, Z) , property(Z, ppt, val) \+ property(A, Prop, _). : This rule will help to find property value of object who may or may not share same property value with their ancestors. For example the default value for man is having two legs but for louis it has one leg only. This rule will change the value for louis.

1. Frank cannot fly:
   Frank is a turkey and turkey have property value different than birds that is turkey cannot fly and hence frank cannot fly. Using the second rule turkey will override the value of property fly from yes to no and frank will use property value of turkey and answer will come true that frank cannot fly.

2. Albert has two legs:
   Albert is a man and man is a human and human have two legs. Man will inherit property value from human as it does not have separate property edge and albert will inherit property value from man as albert does not have separate property edge.

3. Louis have one leg:

Louis is a man and man is a human and human have two legs. Man will inherit property value from human as it does not have separate property edge but louis will not inherit property value from man as louis have separate property edge which means different value from its ancestors.

Question2: For this question consider the following problem description:

Suppose today is your spouse's birthday. You plan to cook dinner and prepare a wrapped present for her. You also want to clean up your apartment for the party. Thus, at the beginning, you have garbage in your apartment, your hand is clean, and the room is very quiet. You need to cook with your clean hand to make dinner. You need to be quiet to wrap present. You can carry the garbage, it will remove the garbage but you hand will be not clean. You can dolly the garbage, it will remove the garbage but the room won't be quiet.

1. **Provide a PDDL representation of the above problem specification.**

Answer: Init(Garbage(g) ⋀ CleanHands(h) ⋀ QuietApt(a))
Goal(Dinner(d) ⋀ Present(p) ⋀ ¬Garbage(g))

1. **Action(** Cook(h, d),
   PRECOND: CleanHands(h),
   EFFECT: Dinner(d))

2. **Action(** Wrap(r, p),
   PRECOND: QuietApt(a),
   EFFECT: Present(p))

3. **Action(** CarryGarbage(g, h),
   PRECOND: Garbage(g),
   EFFECT: ¬Garbage(g) ⋀ ¬CleanHands(h))

4. **Action(** DollyGarbage(g, a),
   PRECOND: Garbage(g),
   EFFECT: ¬Garbage(g) ⋀ ¬QuietApt(a))

2. **Apply the Graph plan algorithm to solve this problem showing the complete steps, mutex links, and the final plan. All levels, actions, variables and mutex links must be clearly labelled and their types shown. Draw the graph and the plan.**
   Answer: Below is the graph

   - Writing initial conditions S0
   - Writing all the Actions A0 whose preconditions are satisfied in S0 and writing persistent actions (No-Op) and writing their effect in S1
   - Find out the mutex relationships in layers A0 and S1
   - Find if we can achieve our goal from this level. Our goal is Dinner(d) ⋀ Present(p) ⋀ ¬Garbage(g)
   - We will fail while finding because if we start making ¬Garbage true with CarryGarbage we cannot make dinner true and if we use DollyGarbage then we cannot wrap the present. So, There is no way that these can occur in parallel.
   - We will create layer A1, where we will write all the actions whose preconditions are satisfied in S1 and persistent actions as well creating the layer S2 where we have all the effects from all actions in A1
   - Find all mutex relations in layer A1 and S2
   - Find if our goal is achievable from this graph. Again we will start making ¬Garbage true with CarryGarbage and then try to satisfy dinner. Dinner will not be possible able carry garbage so we will try to use persist dinner from previous layer. Now we try to satisfy present by using wrap action. We have a way to satisfy all the goal conditions.

Color description and type of link is given below:

🟩 No Operation actions and links

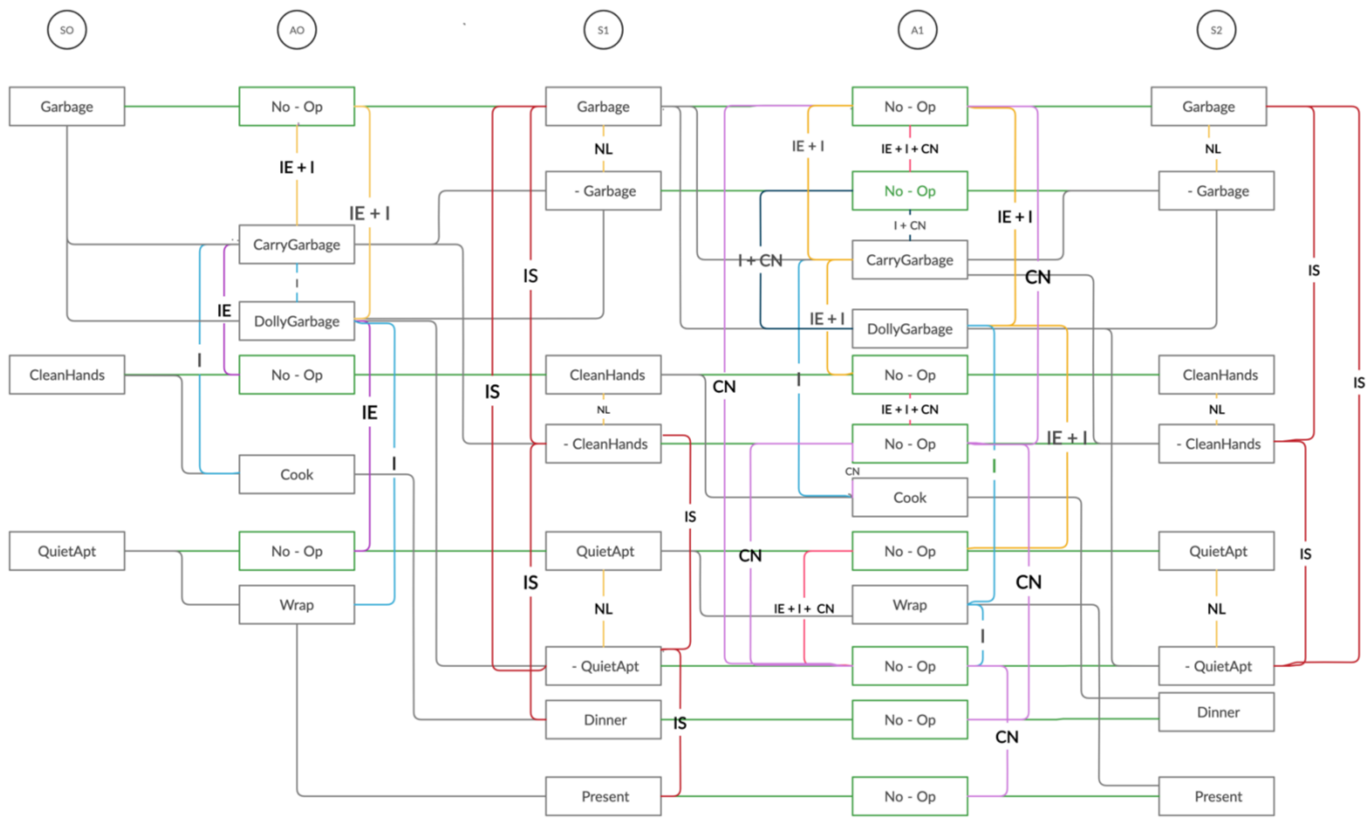
Links in Literals in layers S1 and S2

🟨 **NL** - Negated Literals 🟥 **IS** - Inconsistent Support

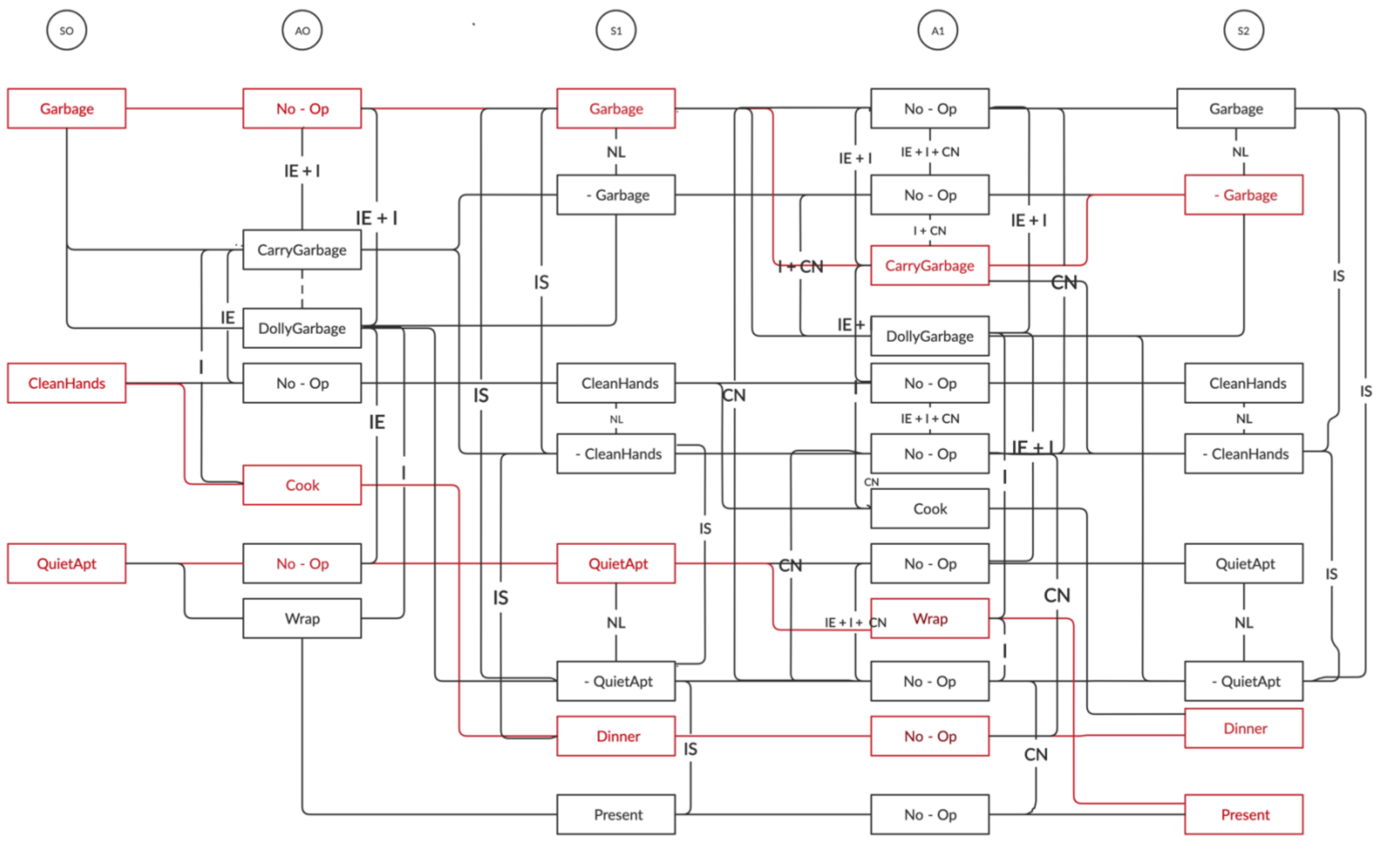
Links in Actions in layers A1 and A2

🟪 **IE** - Inconsistent Effects 🟦 **I** - Interference Actions 🟪 **CN** – Competing Needs

🟦 I + CN 🟨 IE + I 🟥 IE + I + CN



Below is the final plan I choose marked in red. **Cook – Wrap - CarryGarabage**



Explanation of this goal path is given in above points.