Assignment 4

Question 1:

1. (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:

Creatures come in two types: humans and birds.

One type of human is a man.

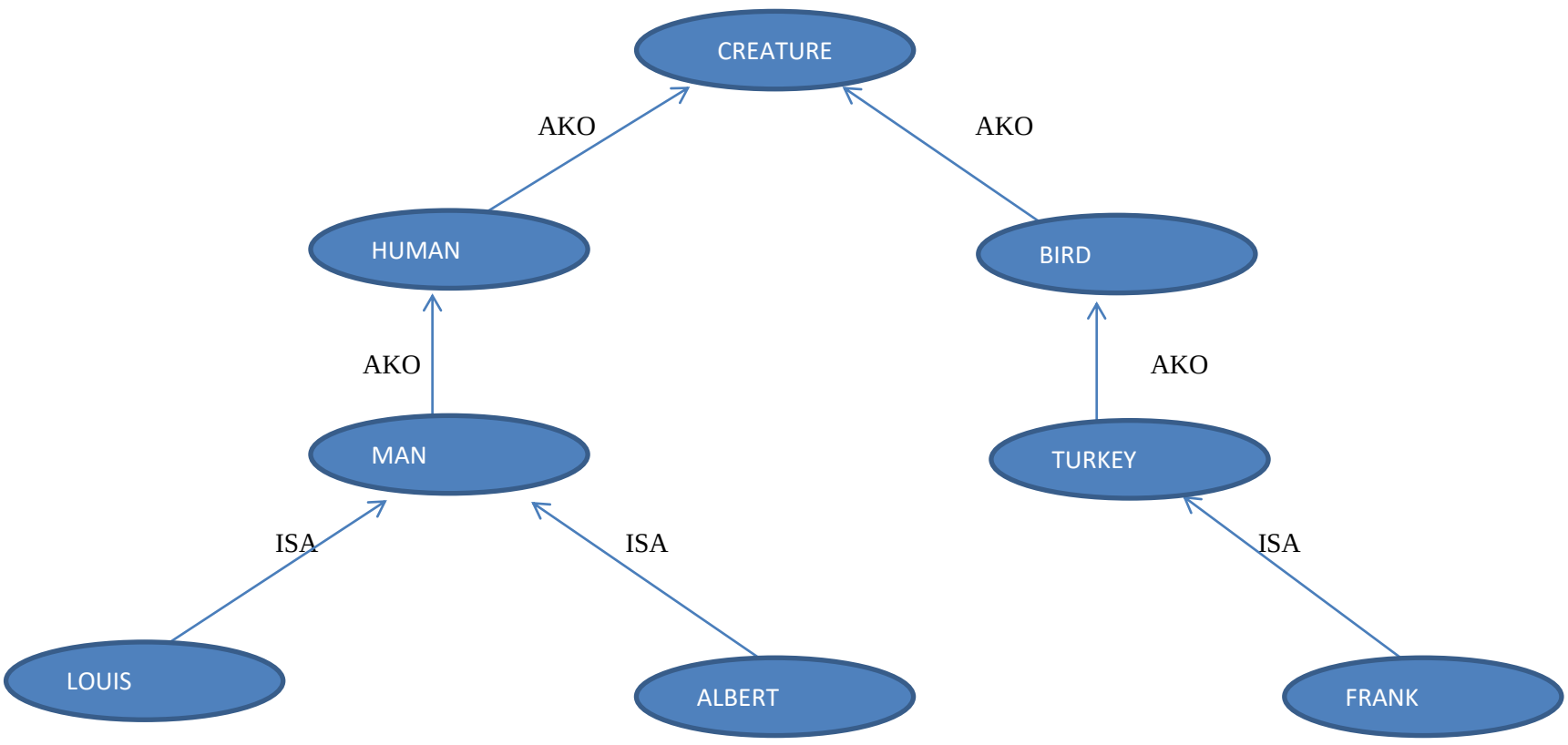
One type of bird is a turkey.

Louis is a man.

Albert is a man.

Frank is a turkey

Graph taxonomy:



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 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

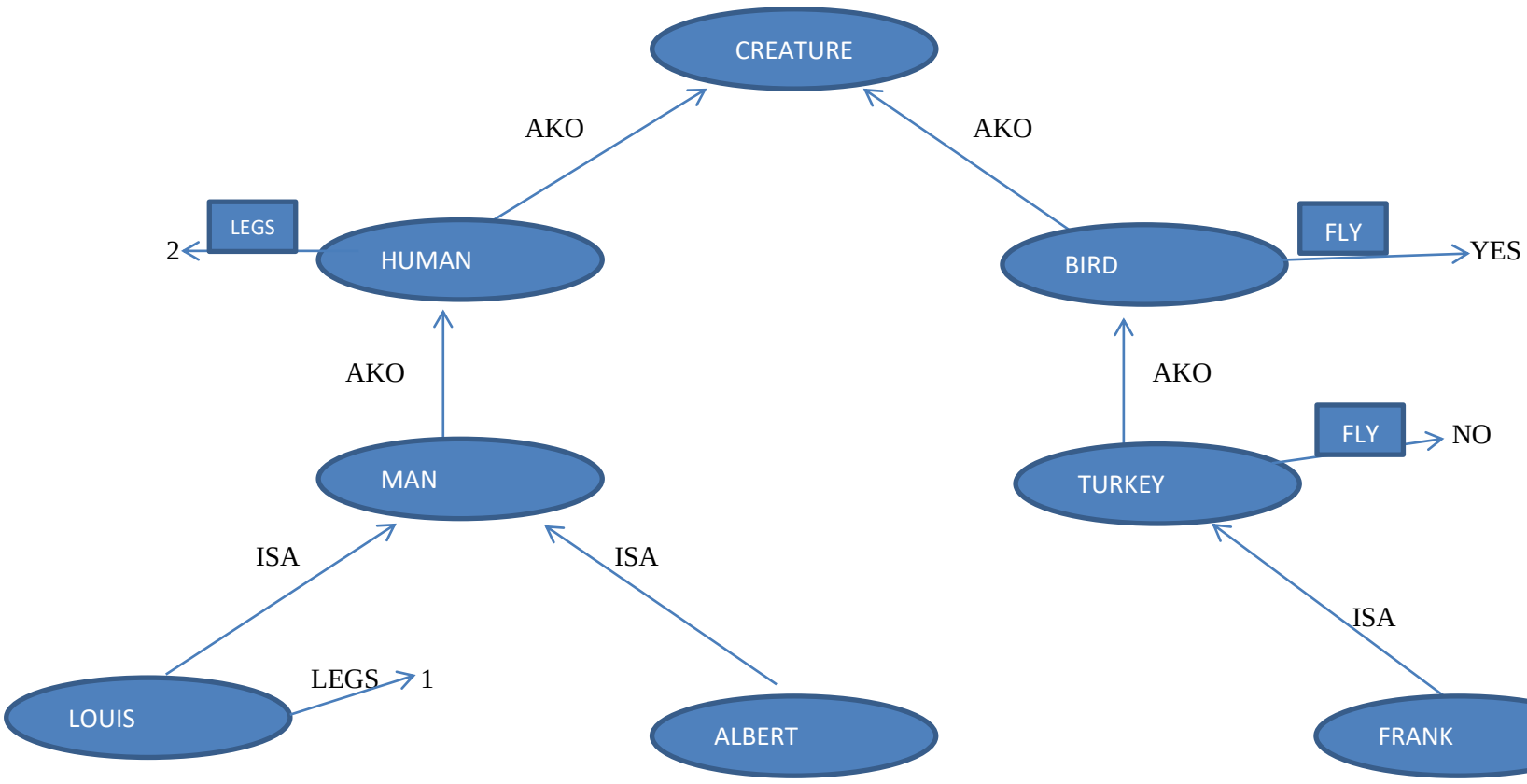
Attached as q1_b_c.pl

3.Your rules should follow strict Prolog syntax, and should allow inference over hierarchies of any depth, not just the depth in this example. 4. (5 pts.)

Attached as q1_b_c.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(, legs, two) and property(, fly, no), indicate which new facts will be necessary, and show in your network sketch from Part (a) where they should be added

Network search:



new facts:

property(human, legs, two).

property(bird,fly,yes).

property(louis,legs,one).

property(turkey,fly,no).

Prolog file:

Enclosed as q1_d.pl

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.

New rules:

prop(X,fly,yes):- property(X,fly,yes).

prop(X,fly,yes):-property(Z,fly,yes),rel(X,isa,Z),not(property(X,fly,no)), not(rel(X,isa,Y)),not(property(Y,fly,no)).

prop(X,legs,two):- property(X,legs,two).

prop(X,legs,two):-property(Z,legs,two),rel(X,isa,Z),not((property(X,legs,Y),not(Y=two))).

prop(X,legs,one):- property(X,legs,one).

prop(X,legs,one):-property(Z,legs,one),rel(X,isa,Z),not((property(X,legs,Y),not(Y=one))).

On queries:

* (i). frank cannot fly : prop(Frank, fly,yes) -> False

* (ii) Albert has two legs : prop (albert,legs,two) -> True

* (iii) Louis has one leg : prop (Louis, legs, two) -> False

Enclosed as:

q1_e.pl

Question 2

2. Plan graph 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.

a. Provide a PDDL representation of the above problem specification

2

a) PDDL representation of the problem.

Init (Garbage (G) ∧ CleanH (H) ∧ Quiet (R))

Goal (Dinner (d) ∧ Present (P) ∧ ¬ Garbage (g))

Action (cook (d),
PRECOND: cleanH(h)
EFFECTS: Dinner (d))

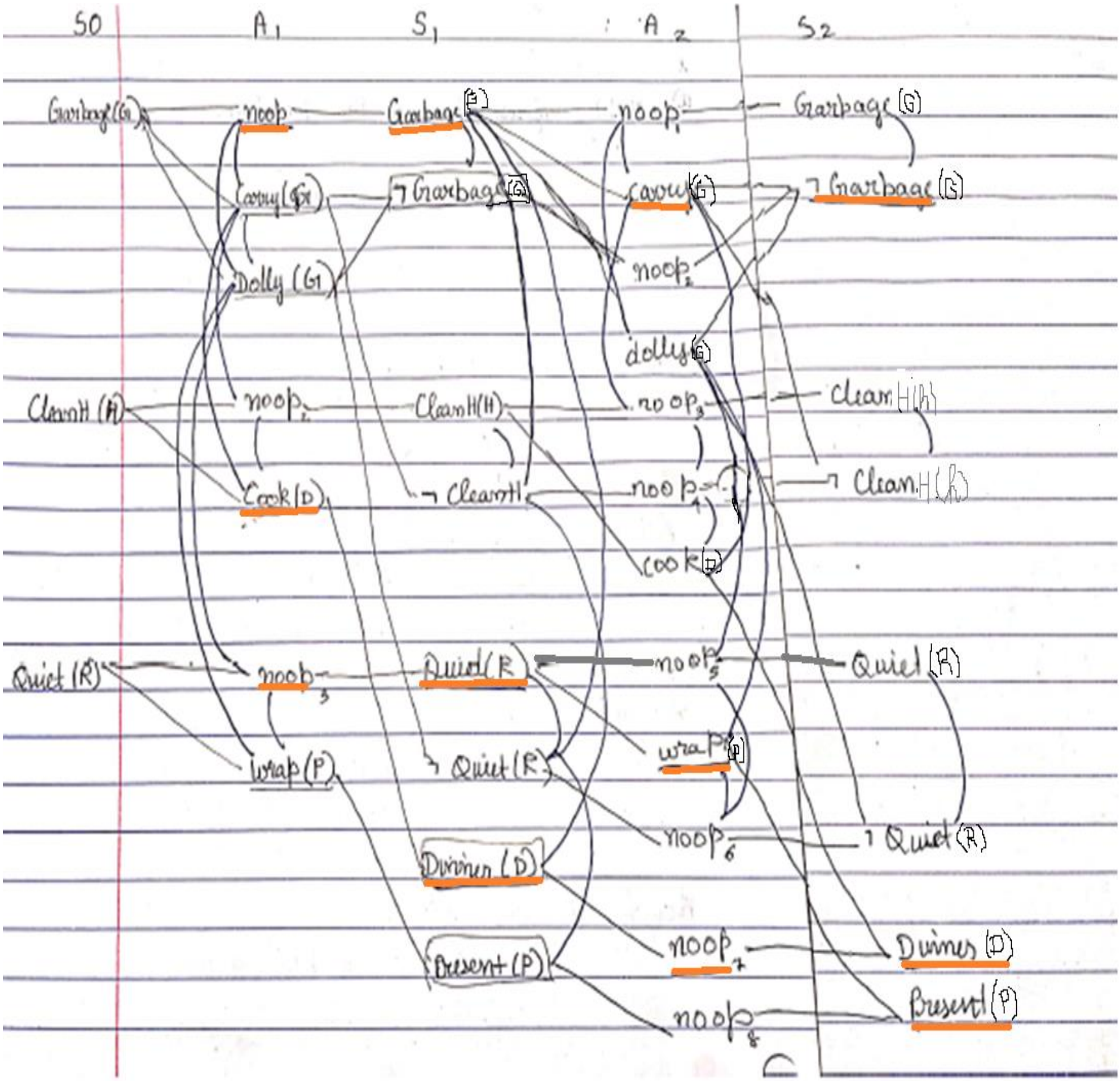
Action (wrap (P),
PRECOND: Quiet (R)
EFFECTS: Present (P))

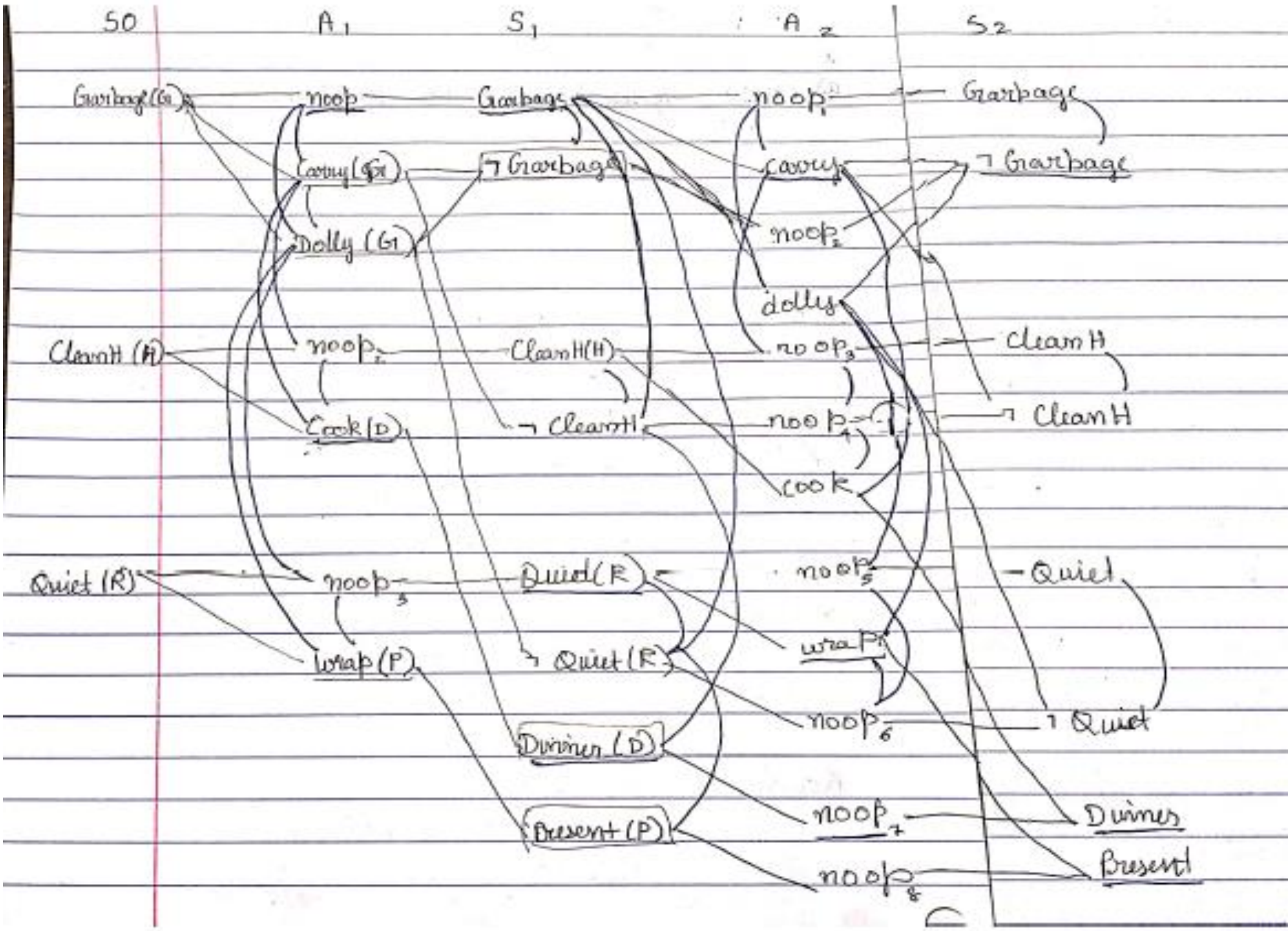
Action (carry (g),
PRECOND: Garbage (g)
EFFECT: ¬ cleanH(h) ∧ ¬ Garbage (g)

Action (dolly (g),
PRECOND: Garbage (g)
EFFECT: ¬ Quiet (R) ∧ ¬ Garbage (g)

b. Apply the Graphplan algorithm to solve this problem showing the complete steps,. All levels, actions, variables and mutex links must be clearly labeled and their types shown. Draw the graph and the plan

Graph:




S0
A1
S1
A2
S2
Garbage (G)
noop1
Garbage
noop1
Garbage
Carry (G)
¬ Garbage
Carry
¬ Garbage
Dolly (G)
noop2
dolly
CleanH (H)
noop3
CleanH (H)
noop3
CleanH
Cook (D)
¬ CleanH
noop4
¬ CleanH
cook
Quiet (R)
noop5
Quiet (R)
noop5
Quiet
Wrap (P)
¬ Quiet (R)
wrap
noop6
¬ Quiet
Dinner (D)
noop7
Dinner
Present (P)
noop8
Present

mutex links

mutex in A1:

IE:

- (noop$_1$ - carry) - (noop$_1$ - Dolly)
- (noop$_2$ - cook) - (noop$_3$ - wrap)

I:

(noop$_1$ - carry) (noop$_1$ - dolly) (carry - dolly)

(noop$_2$ - carry) (noop$_2$ - cook) (noop$_3$ - dolly)

(wrap - dolly). (carry - cook)

CN - None.

mutex in S1:

NL: (garb - ¬garb), (cleanH - ¬cleanH)
(Quiet - ¬Quiet)

IS: -(garb - ¬garb), (cleanH - ¬cleanH)
- (garb - ¬clean), (garb - ¬quiet)
- (¬clean - dinner), (quiet - ¬quiet)
- (¬quiet - present)

Mutexes and Plan (ctd):

At this step all actions in goal set are present and are not mutexed. → walk backwards → the underlined actions carry, dolly, cook & wrap are mutexed.

mutex in A2.

IE :

(noop1 - carry) (noop1 - noop2) (noop1 - dolly)
(noop3 - noop4) (noop3 - carry)
(noop5 - noop6) (noop5 - dolly)

I :

(noop1 - noop2) (noop1 - carry) (noop1 - dolly)
(~~clean~~ noop3 - noop4) (noop4 - carry) (noop6 - wrap)
(noop5 - noop6) (noop5 - dolly) (wrap - dolly) (noop4 - cook)

CN:

(noop1 - carry) (noop1 - noop2) (noop1 - dolly)
(noop3 - noop4) (noop4 - cook)
(noop6 - wrap) (noop5 - noop6)

mutex in S2 :

NL: (garb - ¬garb) (cleanH - ¬cleanH)
(quiet - ¬quiet)

Final plan:

- At S2 all the goal literals are non-mutexed : Dinner ^Present ^ neg(Garbage)
- We move back to the previous action level:
  - We pick the actions corresponding to states in S2 that are not mutexed :
    - Carry ^ wrap ^ noop7
- The states corresponding to the actions in level A1 that are not mutexed are:
    - Garbage ^ Dinner(d) ^Quiet(R)
- The actions in previous states corresponding to the states in S1 that are not mutexed are:
    - Noop1 ^ Cook(D) ^noop3

Hence the final action plan is:(underlined in orange)

To take action Cook(D) in action level 1.

To take action Carry(G) in action level 2,

To take action Wrap(P) in action level 2.