1

a)

¬Bot(B, loop\_on(1)) = {node(1), node(2), link(1, 1), link(1, 2), path(1, 1), path(1, 2), ¬loop\_on(1)}

Bot(B, loop\_on(1)) = ¬node(1) V ¬node(2) V ¬link(1, 1) V ¬link(1, 2) V ¬path(1, 1) V ¬path(1, 2) V loop\_on(1)

= loop\_on(1) <- node(1), node(2), link(1, 1), link(1, 2), path(1, 1), path(1, 2).

H1: loop\_on(X) <- path(Y, X). Or H1: loop\_on(X).

H2: loop\_on(X) <- path(X, X).

Both subsume bottom and fit the mode declaration but H1 would cover the negative example whereas H2 doesn’t.

=====

Alternatively, can H1 be {}, i.e. the empty set?

Or loop\_on(X)

b)

**Iteration 1: e+ = regulates(a, c)**

Step 1: find head of kernel set.

This is done by solving abductive task <B, Ab, IC> where B is background knowledge given, IC is the logical integrity constraints, and Ab is the set of all possible head atoms, given the modeh declarations.

For the heads, I get activates(a, b) and inhibits(b, c), as follows (performing :

← regulates(a, c)

← gene(G3), activates(a, G3), regulates(G3, c)

← activates(a, b), regulates(b, c)

∆ = {activates(a, b)} -- consistency phase goes here but is trivial

← regulates(b, c)

←inhibits(b, c)

∆ = {activates(a, b), inhibits(b, c)} -- consistency phase goes here but is trivial

□

Step 2: find body atoms of kernel set.

For the body atoms, I get link(a, b), link(b, c), diffStatus(b, c), sameStatus(a, b), sameStatus(c, b). I literally just enumerate all possible body atoms given the mode declarations and perform SLD on all of them. Maybe there’s a better way of doing this idk

Step 3. Induction to refine the hypothesis.

In this case the kernel set is already equal to the hypothesis they want.

Step 4 (?): Show the negative example isn’t covered.

Then we’re done. Only one iteration was necessary in this case.

b) ii) Not sure about this, please add suggestions. My guess:

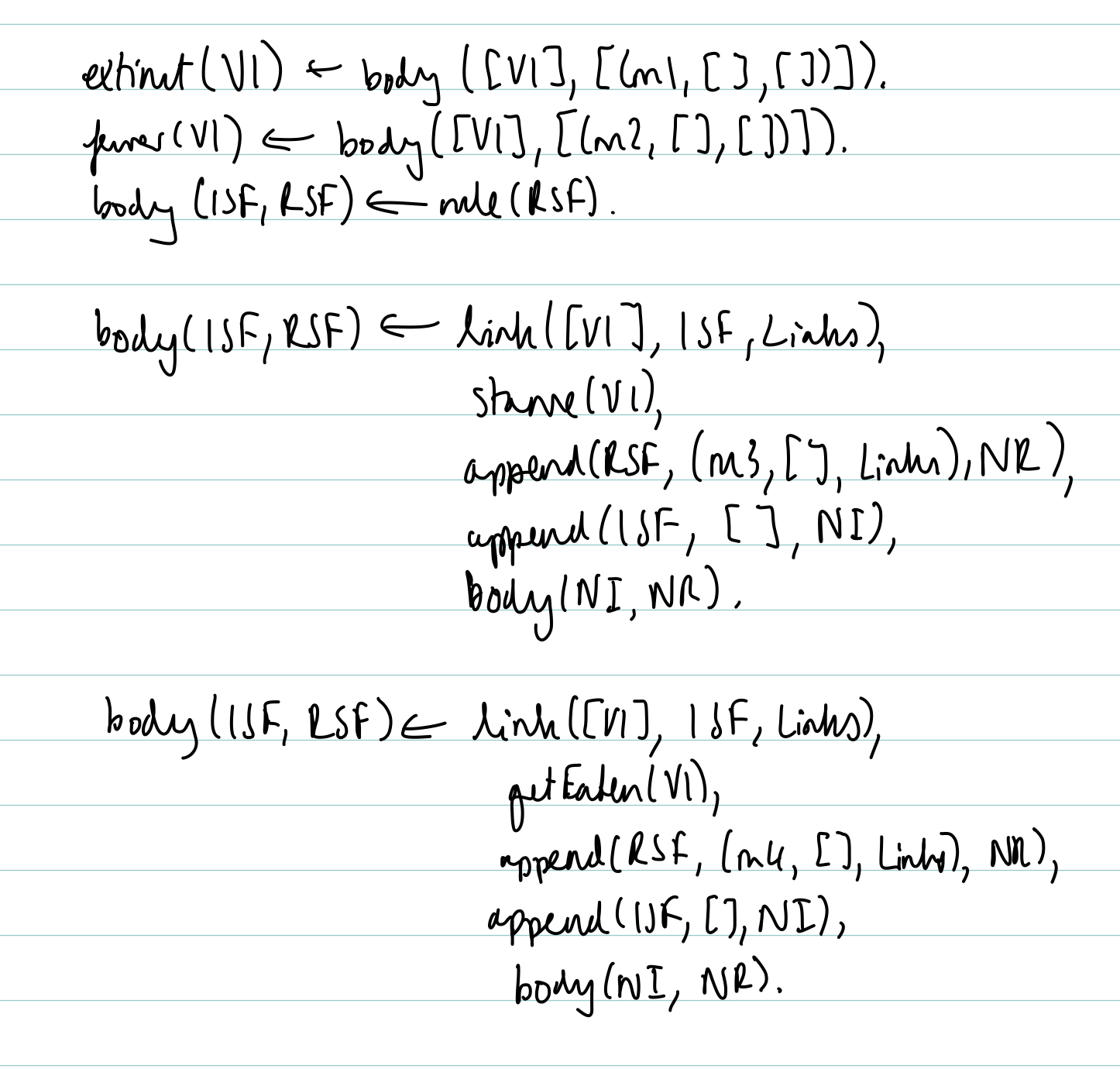
If regulates(a, b) is picked as the first seed, Progol5 fails, as it cannot compute multiple clauses from one seed example. If regulates(b, c) is picked as the first seed, Progol5 will derive the second clause in the first iteration and the first clause in the second iteration.

* Can you not just say here that regulates is used twice in B and start set is incomplete when learning concepts used multiple times in the background?

2

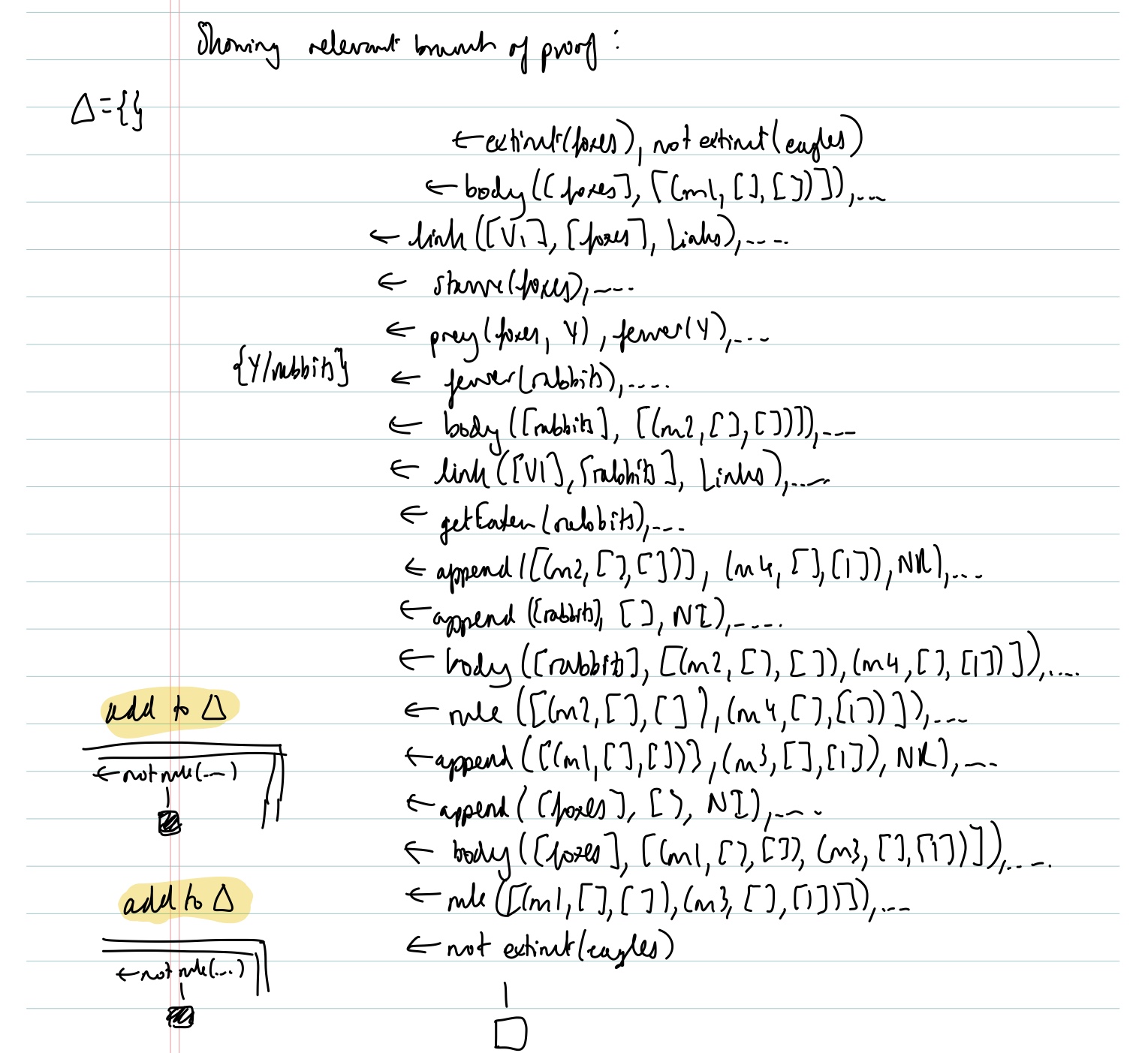
1. The kernel set K(B, e) is the most specific set of unground clauses that, together with the background knowledge, proves the positive seed example e while conforming with the language bias.

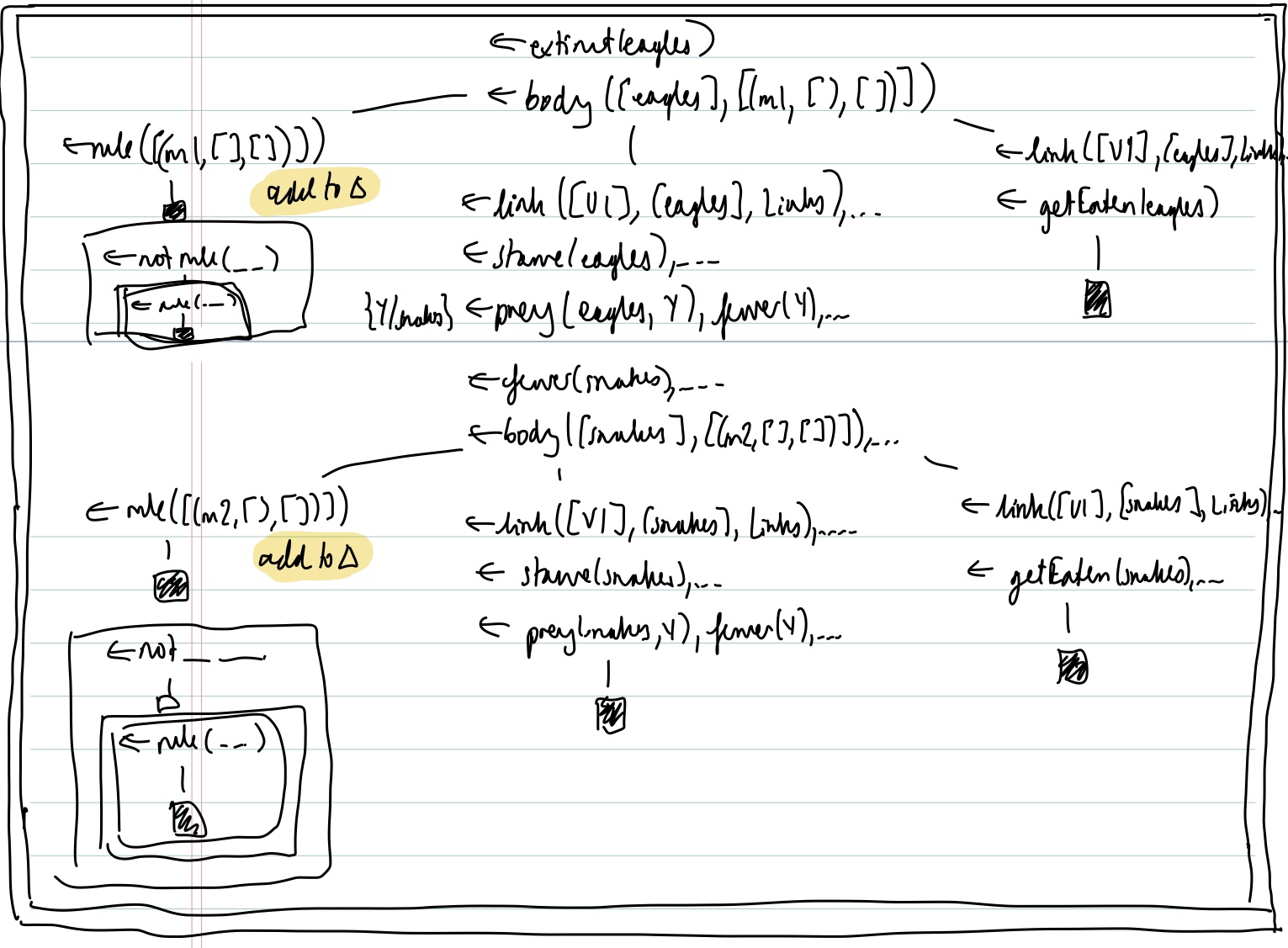
No, not possible using KSS. If we try to use the first two steps of HAIL to derive the kernel set, we get q(a) as the abductive solution in the first step (i.e., the head of ground(K(B, e)) ). We cannot derive any body predicates using this grounding, and so our hypothesis would just be q(X), but this covers the negative example too.

1. (i) Labelling extinct as m1, fewer as m2, starve as m3, getEaten as m4 (done 13/03/21)

(ii) Abductive solution is {rule([(m1, [], []), (m3, [], [1])]), rule([(m2, [], []), (m4, [], [1])]),

not rule([(m1, [], [])]), not rule([(m2, [], [])])}. Proof of relevant branch for this H given below





(iii) No, H is not derivable by KSS – in the first abductive step, we cannot obtain fewer as a head predicate (only abducible is e+ itself, extinct(foxes)). Also, neither of the body predicates can be proved using this grounding.

3ai

Facts = (person(p1), person(p2), food(nuts), food(bread), allergic(p1, nuts), allergic(p2, bread))

Answer sets:

AS1 = Facts, refuse(p1, nuts), refuse(p2, bread), refuse(p1,bread), refuse(p2, nuts)

AS2 = Facts, refuse(p1, nuts), refuse(p2, bread), eats(p1,bread), refuse(p2, nuts)

AS3 = Facts, refuse(p1, nuts), refuse(p2, bread), refuse(p1,bread), eats(p2, nuts)

AS4 = Facts, refuse(p1, nuts), refuse(p2, bread), eats(p1,bread), eats(p2, nuts)

3aii

There is at least 1 answer set, in this case AS4, which entails all of the positive example and none of the negative examples.

3aiii

Yes, since for all of the answer sets AS1..4, they entail the positive example and do not entail the negative example.

3bi

%Sk

eat(P,F) :- person(P), food(F), rule(1).

eat(P,F) :- person(P), food(F), not refuse(P, F), rule(2).

eat(P,F) :- person(P), food(F), not allergic(P, F), rule(3).

eat(P,F) :- person(P), food(F), not refuse(P, F), not allergic(P, F), rule(4).

3bii

%B

As stated in question

%Ab

{rule(1..4)}.

%Goal

goal :- eat(p1, bread), eat(p2, buts), not eat(p1, nuts).

:- not goal.

% Optimisation

#minimise[rule(1)=1, rule(2)=2, rule(3)=2, rule(4)=3].

3biii

AS1 u {rule(4), goal}

3ci

eat(P,F) :- not allergic(P,F), person(P), food(F).

3cii

E+ = { <{refuse(p2, nuts)}, {}> }

E- = { <{eat(p1, nuts)}, {}> }