31.

a

B union {not e} semantically entails not Bot(B, e)

H is derived by bottom generalisation iff H >= Bot(B, e)

b

i)

Not sure how to do, please help, specifically how to determine `Pos2` in the rule:

getTo(Pos1, FinalPos) <- room(Pos2), enter(Pos1, Pos2), goThrough(Pos2, FinalPos)

Here is a summary of how I did it:

**Abduction:** perform abduction to find ∆ such that B ∪ ∆ ⊨ e+. There are a number of different possible abductive solutions (∆’s) here, but the only one necessary in this case is {goThrough(o1, room1), goThrough(room1, goalRoom)}. I’m pretty sure you just use the one abductive solution here, but I’ve asked on piazza about this: <https://piazza.com/class/k0r3c960aegko?cid=99>

Thus, the heads to use are goThrough(o1, room1) and goThrough(room1, goalRoom).

**Deduction:** the relevant body atoms to prove here are outside(o1, room1), doorOpen(o1, room1), archway(room1, goalRoom), room(room1) and room(goalRoom)

Thus we have (leaving grounded for now):

goThrough(o1, room1) <- outside(o1, room1), doorOpen(o1, room1), room(room1)

goThrough(room1, goalRoom) <- archway(room1, goalRoom), room(room1), room(goalRoom)

So our kernel set is

goThrough(P1, P2) <- outside(P1, P2), doorOpen(P1, P2), room(P2)

goThrough(P1, P2) <- archway(P1, P2), room(P1), room(P2)

**Induction:** Use the try/use encoding of the hypothesis space described in the slides (this is quite long so I won’t write it all out), and then perform abduction over this, together with the background knowledge, with the goal of e, and abducibles of all the use(\_, \_) predicates you’ve defined in the encoding. Should be straightforward to derive the necessary hypothesis.

Inductive step:

For the kernel set above, I got T to be:

T = {

goThrough(P1,P2) ← use(1,0), try(1,1,P1,P2), try(1,2,P1,P2), try(1,3,P1,P2)

try(1,1,P1,P2) ← not use(1,1)

try(1,1,P1,P2) ← use(1,1), outside(P1, P2)

try(1,2,P1,P2) ← not use(1,2)

try(1,2,P1,P2) ← use(1,2), doorOpen(P1, P2)

try(1,3,P1,P2) ← not use(1,3)

try(1,4,P1,P2) ← use(1,3), room(P2)

goThrough(P1,P2) ← use(2,0), try(2,1,P1,P2), try(2,2,P1,P2), try(2,3,P1,P2)

try(2,1,P1,P2) ← not use(2,1)

try(2,1,P1,P2) ← use(2,1), archway(P1, P2)

try(2,2,P1,P2) ← not use(2,2)

try(2,2,P1,P2) ← use(2,2), room(P1)

try(2,3,P1,P2) ← not use(2,3)

try(2,3,P1,P2) ← use(2,3), room(P2)

}

Ab = {

use(1,0), use(1,1), use(1,2), use(1,3), use(2,0), use(2,1), use(2,2), use(2,3),

not use(1,0), not use(1,1), not use(1,2), not use(1,3), not use(2,0), not use(2,1), not use(2,2), not use(2,3)

}

IC = {

← use(1,0), not use(1,0)

← use(1,1), not use(1,1)

← use(1,2), not use(1,2)

← use(1,3), not use(1,3)

← use(2,0), not use(2,0)

← use(2,1), not use(2,1)

← use(2,2), not use(2,2)

← use(2,3), not use(2,3)

}

Then resolve <B⋃T, Ab, IC> (by doing SLD on e+)

You’ll get the abductive solution:

∆ = {

use(1,0), not use(1,1), use(1,2), not use(1,3),

use(2,0), use(2,1), not use(2,2), not use(2,3)

}

Which corresponds to given H

ii)

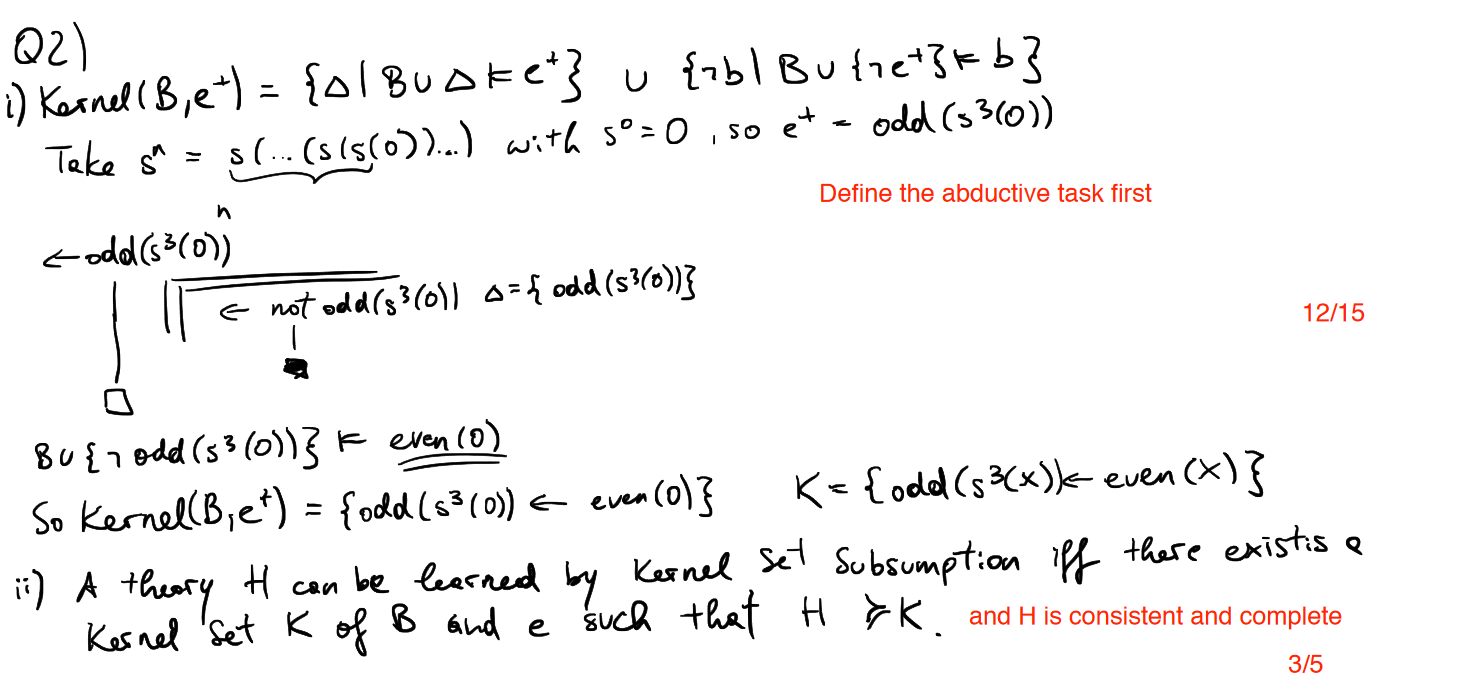
No. Progol5 only works if head predicate used once.

2

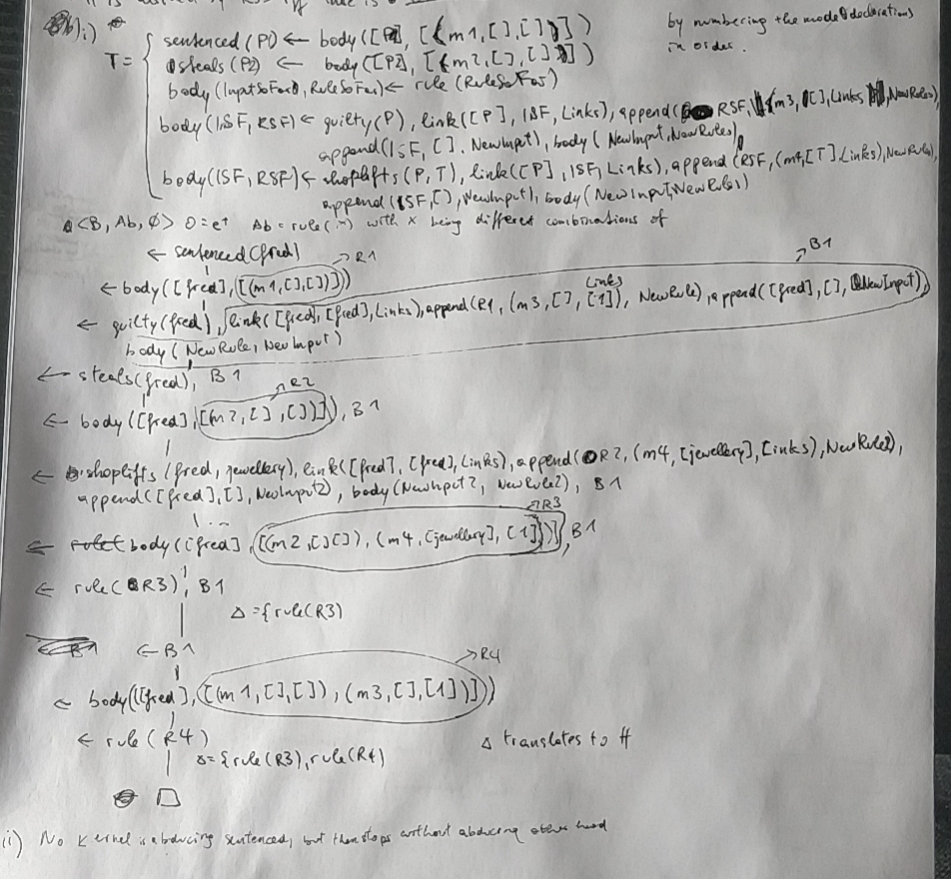
a)

CW1 Q2:

Abductive task is <B, AB, {}> with Ab = set of instances of predicates of madeh, and Observation = e+



b)



3

a

i) A1: F (Facts) U

{male(p1)

king(p1)

male(p2)

entertainer(p2)}

----

A2: F U

{male(p1)

king(p1)

female(p2)

entertainer(p2)}

---

A3: F U

{female(p1)

male(p2)

entertainer(p2)}

---

A4: F U

{female(p1)

female(p2)

entertainer(p2)}

ii)

We can see from above that at least 1 answer set have {king(p1)} and doesn’t have {king(p2)}

iii)

Yes. We have all answer set containing the {entertainer(p2)} not all answer set containing {king(p2)}

iv)

<B, {<{king(p1}, {king(p2)}>}, {<{},{entertainer(p2)}>, <{king(p2)},{}>}>

b

i)

entertainer(P) :- person(P)

king(P) :- person(P)

entertainer(P) :- person(P), male(P)

entertainer(P) :- person(P), job(J), has\_job(P, J)

king(P) :- person(P), male(P)

king(P) :- person(P), job(J), has\_jobs(P, J)

entertainer(P) :- person(P), job(J), male(P), has\_job(P, J)

king(P) :- person(P), job(J), male(P), has\_job(P, J)

ii)

% B

% Skeleton rules

entertainer(P) :- person(P), rule(1)

king(P) :- person(P), rule(2)

entertainer(P) :- person(P), male(P), rule(3)

entertainer(P) :- person(P), job(J), has\_job(P, J), rule(4, J)

king(P) :- person(P), male(P), rule(5)

king(P) :- person(P), job(J), has\_jobs(P, J), rule(6, J)

entertainer(P) :- person(P), job(J), male(P), has\_job(P, J), rule(7, J)

king(P) :- person(P), job(J), male(P), has\_job(P, J), rule(8, J)

% generate Hypothesis

{rule(1..3), rule(4, monarch), rule(4, jester), rule(5), rule(6, monarch), rule(6, jester) andsoon}

goal :- king(p1), not king(p2)

:- not goal

#minimize[rule(1..3)=1, rule(4, J)=2, rule(5) = 2 andsoon].

iii)

{rule(8, monarch), rule(4, jester), goal} U (A1 or A2)

4

Don’t think this is covered anymore.