Done on the 11/3/20. Please highlight any mistakes.

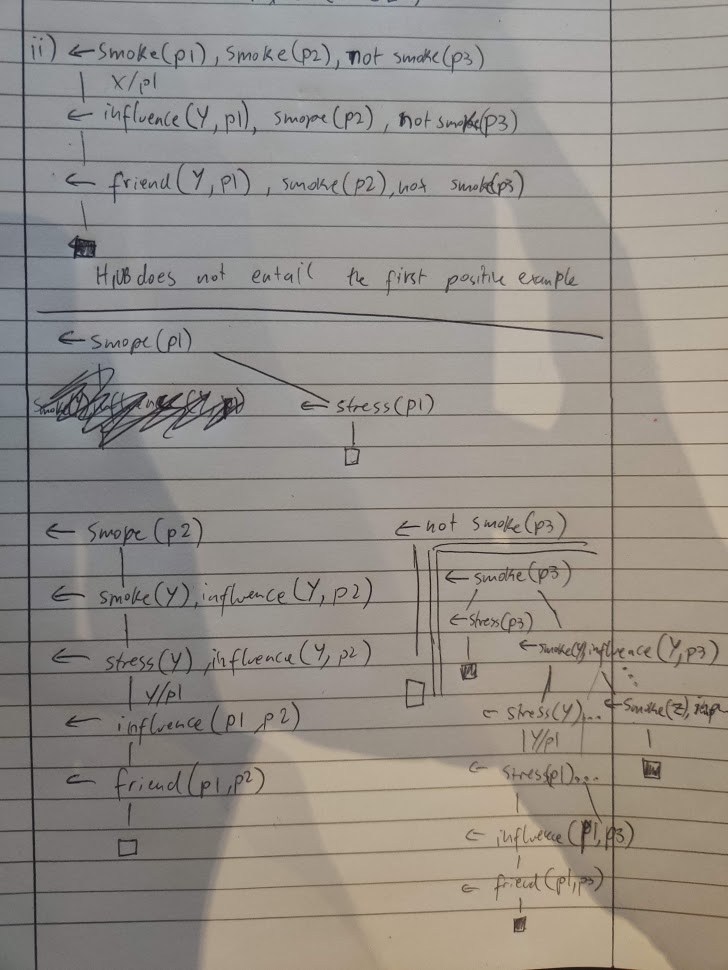
1. a)

i) The learning from entailment task means we have to find a hypothesis H such that BUH entails every positive example and does not entail any of the negative examples.

ii) H1 is NOT an inductive solution.

E+\_1 isn’t covered and E-\_1 is covered

H2 IS an inductive solution, E+ covered and E- not.



b) i)

HAIL step 1: PIck positive example do(d1, medication)

Perform abduction with do(d1, medication) as goal

do(d1, medication)

I

hasRole(d1, R), permitted(R, medication), granted(d1, medication).

I

Unify R with doctor

permitted(doctor, medication), granted(d1, medication)

Abducibles = permitted(doctor, medication)

(Insert consistency check here)

I

granted(d1, medication)  
 Abducibles = permitted(doctor, medication), granted(d1, medication)

(Insert consistency check here)

I

[]

Step 2: Using B U neg E+, deduce possible grounded body declarations.

assigned(diagnosis, doctor), assigned(medication, nurse), assigned(labTest, intern)

trusted(d1), trusted(i1)  
 assigned(medication, doctor), assigned(labTest, nurse), assigned(labTest, doctor)  
  
Step 3: Create Kernel set:

permitted(R, A) <- assigned(A, R).

granted(S, A) <- trusted(S).

Step 4: Inductive step:3你

Transform kernel set

permitted(R, A) <- use(1, 0), try(1, 1, R, A).

try(1, 1, R, A) <- not use(1,1).

try(1, 1, R, A) <- use(1,1), assigned(A, R).

granted(S, A) <- use(2, 0), try(2, 1, S, A).

try(2, 1, S, A) <- not use(2, 1).

try(2, 1, S, A) <- use(2,1), trusted(S).

Now perform abductive proof with all positive and negative examples as goal.

Abducibles are all valid use rules

<- do(d1, medication), not do(n1, medication), not do(i1, medication)

I

<- hasRole(d1, R), permitted(R, medication), granted(d1, medication). (Plus negations)

I

<- permitted(doctor, medication), granted(d1, me我吧……6yjdication) + negations

I

<- use(1, 0), try(1, 1, doctor, medication), granted(d1, medication) + negations

(Abduce use(1,0)  
 (Perform consistency check)

I  
<- try(1, 1, doctor, medication), granted(d1, medication) + negations

(We’ll use the right try because the Q asks to prove H rather than find H)  
<- use(1, 1), assigned(medication, doctor), granted(d1, medication) + negations

(Abduce use(1,1), perform consistency check)

<- assigned(medication, doctor), granted(d1, medication) + negations  
 I

<- subRole(R1, doctor), assigned(medication, R1) + rest

I

<- assigned(medication, nurse) + rest

I

<- granted(d1, medication) + rest

I

<- use(2,0), try(2, 1, d1, medication) + rest

(Abduce use(2,0), perform consistency check)

I

<- try(2,1, d1, medication) + rest

I

<- use(2,1), trusted(d1), + rest

(Abduce use(2,1), perform consistency check)

I

<- trusted(d1) + rest

I

<- not do(n1, medication) + rest

(Open NAF box)

I <- do(n1, medication)

I I

I <- hasRole(n1, R), permitted(R, medication), granted(n1, medication)

| <- permitted(nurse, medication), granted(n1, medication)

| (We skip to use(1,1))

| <- assigned(medication, nurse), granted(n1, medication)

| (This works)

| <- granted(n1, medication)

| (Skip to use(2,1)  
 | <- trusted(n1)

| X (this fails)

<- not do(i1, medication)  
 (Repeat but this one fails on use(1,1), assigned(medication, intern)

I

[]

ii) Can this be computed by Progol5

Virgin Progol5 cannot ever compete with chad HAIL

StartSet is incomplete

It is not possible as we can only choose to perform SLD on one mode head at a time, and deducing not\_permitted would require granted, therefore it isn’t possible

Progol5 can only compute 1 clause per hypothesis, and 2 are required to answer this one.

1. a)

i) We assign the mode declarations m1 to m4

We create the Top set T = {

preferredVehicle(B1) <- body([B1], [(m1, [], [])]).

peopleCarrier(B1) <- body([B1], [(m2, [], [])]).

body(Inputs, Rules) <- rule(Rules).

body(Inputs, Rules) <- se , link(Inputs, [B1], Links),  
 append(Rules, (m3, [N], Links), NewRules),

append(Inputs, [], NewInputs),

body(NewInputs, NewRules).

body(Inputs, Rules) <- economical(B1), link(Inputs, [B1], Links),

append(Rules, (m4, [], Links), NewRules),

append(Inputs, [], NewInputs),

body(NewInputs, NewRules).

}

Now we perform abduction on <T U B> goal is preferredVehicle(ford), not preferredVehicle(honda).

Abducibles are all possible ground combinations of rule

A = rule([m1, [], []]), rule([(m2, [], [])]]... etc etc

<- preferredVehicle(ford) + rest

<- body([ford], [(m1, [], [])] + rest

<- economical(ford), link([ford], [ford], [1]) + rest

<- hybrid(ford), peopleCarrier(ford) + rest

<- peopleCarrier(ford) + rest

<- body([ford], [(m2, [], [])]) + rest

<- seats(ford, X) … + rest

(Unify X with 8)  
 <- link([ford], [ford], [1]) + rest

(NewINputs = [ford], newRules = [(m2, [],[]), (m3, [8], [])])  
 <- body(NewI, newR) + rest

<- rule(newR) + rest

(Abducibles = rule([(m2, [],[]), (m3, [8], [])]))

(Insert consistency check here)

(Pick up from the previous rule)

<- link([ford], [ford], [1]) + rest

(newInputs = [ford], newRules = [(m1, [], []), (m4, [], [])])

<- body(NewI, newR)

<- rule(newR)

(Abducibles = rule([(m2, [],[]), (m3, [8], [])]), rule( [(m1, [], []), (m4, [], [])])))  
 (Insert consistency check)  
 Finish

Convert rules back into actual rules, giving us H

ii) There is a more general solution for T

~~Replace PeopleCarrier(X) <- seats(X, 8) with peopleCarrier(X) <- seats(X, Y).~~

~~It is more general than H as it theta-subsumes H (Y=8) and it is complete as it covers every positive example, and consistent as it does not cover any negative examples.~~

Above is incorrect as the generalisation does not respect the mode declaration (second argument of seats has to be a constant)

Replace PeopleCarrier(X) <- seats(X, 8) with peopleCarrier(X) <-

b)

i) To generate the KS, do abduction with goal preferredVehicle(ford), which gives this same term as the only head. The possible bodies are seats(ford, 8) and economical(ford), but the deductive SLD step would not give you economical(ford), since it requires peopleCarrier(), which isn’t in B.

Hence the KS is {preferredVehicle(X) <- seats(X, 8)}, which is not theta-subsumed by H. So H is not derivable by KSS.

~~Then, the answer would generate a general solution preferredVehicle(X), which is not consistent, as it covers the negative examples.~~

ii) If we delete peopleCarrier(X) from the economical(X) rule, we are able to deduce economical(X), giving a complete and consistent solution using KSS, which is preferredVehicle(X) <- economical(X). (Ignoring peopleCarrier)

3.a)

i) Say F = {q(1,2), q(2,1), t(1), t(2)}

Answer sets = {

F U {p(1, 1), p(2, 2)},

F U {r(1,1), r(2, 2)},   
F U {r(1,1), p(2, 2)},

F U {r(2, 2), p(1, 1)}}

}

ii) Yes, because at least one answer set entails p(1,1) and doesn’t entail q(1,1)

iii) No, because not every answer set entails p(1,1)

b) 2021 – not covered

i) Yes, it bravely respects o because there is at least one set of answer sets for the orderings where the left has a lower cost than the right

ii) No, because you could buy(a), buy(b) on the left and buy(b) on the right, giving a higher cost for the left side of the ordering.

C. i) Not possible, since the empty set would be a valid solution to any examples you could put.

E^+ = {p}, E^- = null

B U H gives 1 answer set i.e. {p}.

ii) Its not possible, because the Hypothesis gives answer sets {p} and {q}, which do not intersect, therefore you cannot give examples for a cautious learning task

iii) It is possible, giving E+ = <{p}, %>, <{q}, %> and E- = <%, {p, q}>, then the empty set isn’t valid and neither are single atoms.

iv) No, since r isn’t anywhere else, you get the same answer sets as the empty set, therefore any examples you could give, the answer set would be an optimal answer.

v) Yes, with E+ = <<{p, r}, %>, { r }> E- = <<{r}, {p}>, {r}>.

4.

a)

i) Skeleton rules {

// ... means to specify the types of the input vars (no need to do this for constant C)

// T1 can only be used by r() after it is outputted by q()

p(T2,C) :- ...

p(T2,C) :- q(T1), ...

p(T2,C) :- q(T1), r(T1, T2), ...

p(T2,C) :- q(T1), q(T1’), ...

p(T2,C) :- q(T1), q(T1’), r(T1’,T2), ....

p(T2,C) :- q(T1), q(T1’), r(T1,T2), ....

~~p(T, C) :- q(T1), r(T1, C), t1(T1), t2(T), t2(C).~~

~~p(T, C) :- q(T1), r(T1, T), r(T1, C), t1(T1), t2(T), t2(C).~~

}

ii) Encoding:

Insert background Knowledge

p(T, C) :- t2(T), t2(C), rule(1, C).

p(T, C) :- q(T1), t1(T1), t2(T), t2(C), rule(2, C).

p(T, C) :- q(T1), r(T1, T), t1(T1), t2(T), t2(C), rule(3, C).

~~p(T, C) :- q(T1), r(T1, C), t1(T1), t2(T), t2(C), rule(4, C).~~

~~p(T, C) :- q(T1), r(T1, T), r(T1, C), t1(T1), t2(T), t2(C), rule(5, C).~~

{rule(1, a), rule(1, b), rule(2, a), rule(2, b)....you know}  
 goal :- p(a,b), not p(b,b).

:- not goal.

#minimize(rule(1, C)=1, rule(2,C)=2, rule(3,C)=3, ~~rule(4,C)=3, rule(5,C)=4~~)

iii) Add weak constraints:

:~ rule(1, C).[1@1, rule(1, C)]

:~ rule(2, C).[2@1, rule(2, C)]

:~ rule(3, C).[3@1, rule(3, C)]

~~:~ rule(4, C).[3@1, rule(4, C)]~~

~~:~ rule(5, C).[4@1, rule(5, C)]~~

:~ not p(a,b).[1@1, p(a,b)]

:~ p(b,b).[2@1, p(b,b)]

b) i) EVID: The unconditional probability of evidence e.

MARG: The conditional probability of query q succeeding given evidence e P(q | e).

MPE: The world with the highest probability where query q succeeds given the evidence. Arg max P(q | e)

ii) 8 worlds. (combinations of the probabilistic facts being true or false)

For buyTicket(ale) to be true, (sale V voucher) ^ likeTheatre(ale) needs to be true.

This occurs in the words w1, w2, w3 = {1, 0, 1}, {0, 1, 1} and {1, 1, 1}

P(buyTicket(ale)=true) = P(w1) + P(w2) + P(W3) = 0.6\*0.7\*0.8 + 0.4\*0.3\*0.8 + 0.6\*0.3\*0.8 = 0.336 + 0.096 + 0.144 = 0.576.

iii) They are NOT pairwise incompatible, since the traces for both branches overlap and are not mutually exclusive.

We would get {sale, likeTheatre(ale)} and {voucher, likeTheatre(ale} but they both are part of the word {sale, voucher, likeTheatre(ale)}.