This is just my attempt, so pls feel free to correct all the errors lol

1.a.i.

S1= yes  
S2= no – rule 3 doesn’t hold  
S3= no – rule 2 doesn’t hold  
S4= yes  
S5= yes

1.a.ii.

S1= no - {a,b,c} doesn’t entail b, not c OR a, not c  
S2= no – not a model  
S3= no – not a model  
S4= yes – X entails b, not c and a, not c  
S5= yes – X entails not a

1.a.iii.

S1= no – M(PX) = {} =/= {a,b,c}  
S2= no – M(PX) = {c} =/= {}  
S3= no – M(PX) = {} =/= {a}  
S4= no - M(PX) = {} =/= {a,b}  
S5= yes - M(PX) = {c} = {c}

1.b.i.

b <-- a

{a} is no longer stable – as M(P{a}) = {a,b}. {} is still not a stable model, and therefore every other potential model would either have to contain b or c, and therefore wouldn’t be subset minimal and so wouldn’t be a stable model

1.b.ii.

p <-- not q

q <-- not p

Stable models: {a,p}, {a,q}, {b,p}, {b,q}, {c,p}, {c,q}

1.c.

Let P be a normal logic program, let X be a stable model of P

Assume Y ⊆ X and Y ⊨ P.

We need to show Y = X:

First we can show Y ⊨ PX :

Since Y ⊆ X, PX ⊆ PY

Since Y ⊨ P, Y ⊨ PY

Hence, since PX ⊆ PY , Y ⊨ PX

Now we know Y ⊨ PX, i.e. Y is a model of the reduct PX

Since X = M(PX), X is the least model of PX, and so X ⊆Y

Since Y ⊆ X and X ⊆Y, X = Y

2.a.i

% Rule 1

ruleBodySat(r1(X)) :- tharg(X).

ruleHeadFires(r1(X)) :-

ruleBodySat(r1(X)),

not -blue(X),

not -ruleHeadFires(r1(X)).

blue(X) :- ruleHeadFires(r1(X)).

% Rule 2

ruleBodySat(r2(X)) :- veg(X), tharg(X).

ruleHeadFires(r2(X)) :-

ruleBodySat(r2(X)),

not -green(X),

not -ruleHeadFires(r2(X)).

green(X) :- ruleHeadFires(r2(X)).

% Rule 3

ruleBodySat(r3(X)) :- hum(X), tharg(X).

ruleHeadFires(r3(X)) :-

ruleBodySat(r3(X)),

not -red(X),

not -ruleHeadFires(r3(X)).

red(X) :- ruleHeadFires(r3(X)).

% Exception

-ruleHeadFires(r1(X)) :- ruleBodySat(r2(X)).

-ruleHeadFires(r1(X)) :- ruleBodySat(r3(X)).

-ruleHeadFires(r2(X)) :- ruleBodySat(r3(X)).

% Extra Info

-red(X) :- blue(X).

-red(X) :- green(X).

-blue(X) :- green(X).

-blue(X) :- red(X).

-green(X) :- red(X).

-green(X) :- blue(X).

2.a.ii

Bravely – Zargon is green and red

Cautiously – Nothing re. colour

2.a.iii

Concludes he is red

2.b.i

Answer set of P1: {p}, {q}

Answer sets of P2: {p}, {q}

Equivalent

No strongly equivalent

Check by adding

q :- p.

p :- q.

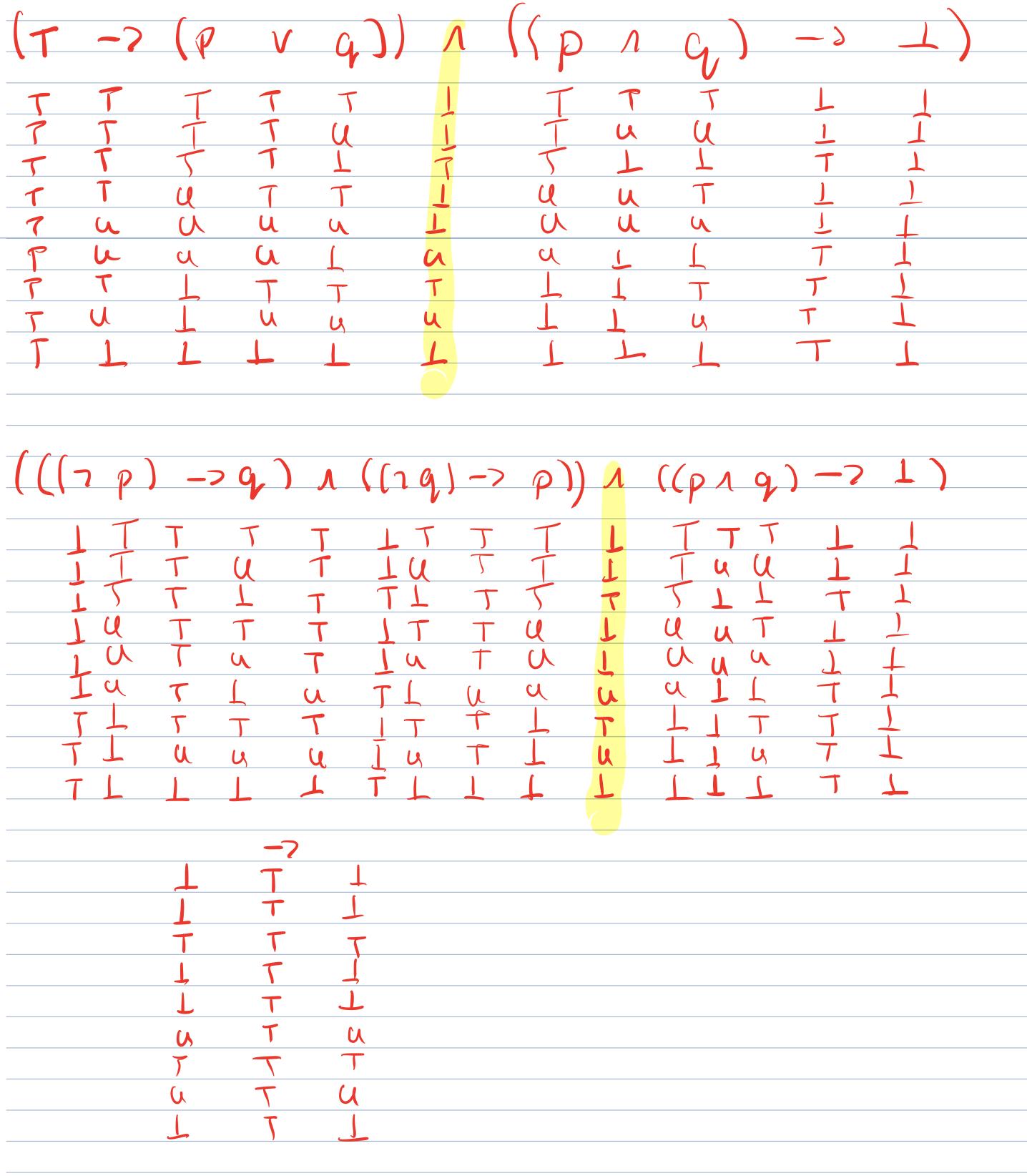
2.b.ii

Answer set of P1: {p}, {q}

Answer sets of P2: {p}, {q}

Equivalent

They are strongly equivalent (since both programs give the same top level values, L -> R and R -> L will just be all true in both cases)



Note: in the lecture notes, usually ¬p is written as (p -> ⊥) in HT logic. But ¬ is still defined in the notes, so it should be fine.

3.a

Looks like 3+4 are similar to our course but using a different language than C+