Create horn clause and forward chaining to assert all statements that can be derived from it

R1 : ~P11

R2 : B11 <=>(P12 v P21)

R3: B21 <=> (P11 v P22 V P31)

R4: ~B11

R5: B21

Answer:

Horn clause means at most 1 positive literal

H1 : ~P11

R2 : B11 ⬄ (P12 v P21)

(B11 => (P12 v P21)) ^ ((P12 v P21) => B11)

= (~B11 v (P12 v P21)) ^ (~(P12 v P21) v B11)

= (~B11 v P12 v P21) ^ ((~P12 ^ ~P21) v B11)

= (~B11 v P12 v P21) ^ (~P12 v B11) ^ (~P21 v B11)

R3: B21 ⬄ (P11 v P22 V P31)

(B21 => (P11 v P22 V P31)) ^ ((P11 v P22 V P31)=>B21)

= (~B21 v (P11 v P22 V P31)) ^ (~(P11 v P22 V P31)v B21)

= (~B21 v P11 v P22 v P31) ^ ((~P11 ^ ~P22 ^ ~P31)v B21)

= (~B21 v P11 v P22 v P31) ^ (~P11 v B21) ^ (~P22 v B21) ^ (~P31 v B21)

R4: ~B11

R5: B21

So horn clauses so formed are :

1. ~P11
2. P12 v P21=> B11
3. (P11 v P22 V P31)=>B21
4. ~B11
5. B21

Then using forward chaining we have, nothing can be derived directly from the clauses got but we can derive using other methods to get some result.

Using P12 v P21=> B11, we can apply contrapostive

~B11 => ~(P12 v P21).

We can use forward chaining now

Using ~B11 , we have ~(P12 v P21) is true

which implies ~P12 and ~P21 means no pit in P12 and no pit in 21 adding extra 2 statements to KB :

R6: ~P12

R7: ~P21