3.1

**Using the following algorithem from calss-material**

unify(A,B): Substitution | Fail

Initialization: sub: Substitution = {} // Empty substitution

equations: Equation[] = (A = B)

1. While (equations is not empty):

2. Let equation\_1 = pop(equations)

3. Let eq'\_1 = equation\_1 ○ sub

4. If one side in eq'\_1 is a variable X:

4.1 If the other side is not the same variable: i.e., eq'\_1 = {X = term}

4.2 then sub = sub o {X = term}

4.3 else if the other side is the same variable: i.e., eq'\_1 = {X = X}

4.4 continue

5. else if both sides in eq'\_1 are atomic, then:

6. if both sides are the same constant symbol then continue, else return FAIL.

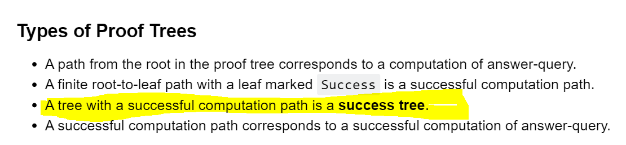
7. else if the predicate symbols and the number of arguments are the same: eq'\_1 = (p(t\_1,...,t\_n) = p(s\_1, ..., s\_n)):

8. split eq'\_1 into equations: equations = equation U (t\_i = s\_i) for i=1..n, continue.

9. else return FAIL.

1. Unify [ t ( s ( s ) , G , H , p, t ( E ) , s ) ,  
    t ( s ( H ) , G , p , p, t ( E ) , K ) ]  
   initialization: s={}, Equation = { t ( s ( s ) , G , H , p, t ( E ) , s ) = 5 t ( s ( H ) , G , p , p, t ( E ) , K )}
   1. equation\_1= t ( s ( s ) , G , H , p, t ( E ) , s ) = 5 t ( s ( H ) , G , p , p, t ( E ) , K )  
      eq’\_1 = equation\_1s= equation\_1{}= equation\_1  
      case7 is true:  
      Equations = [][s(s)=s(H), G=G,H=p,p=p,t(E)=t(E), K=s]= [s(s)=s(H),H=p, K=s]
   2. equation\_1= s(s)=s(H)  
      eq’\_1 = equation\_1s= equation\_1{}= equation\_1  
      case7 is true:  
      Equations= equations]H=s[ == [H=p, K=s]]H=s[= FAIL   
      H can’t be equal to p and to s simultaneously
2. Unify [ g ( c , v ( U ) , g , G , U , E , v ( M ) ) ,   
    g ( c , M , g ,v ( M ) , v ( G ) , g , v ( M ) ]  
   initialization: s={}, Equation = [g ( c , v ( U ) , g , G , U , E , v ( M ) ) = g ( c , M , g ,v ( M ) , v ( G ) , g , v ( M ) ]
   1. equation\_1 = Equations.pop() = g ( c , v ( U ) , g , G , U , E , v ( M ) ) = g ( c , M , g ,v ( M ) , v ( G ) , g , v ( M )   
      eq’\_1 = equation\_1{}= equation\_1   
      case7 is true: splitting equation to little eqatuins  
      Equations =[M=v(U),G=v(M),U=v(G),E=g,]
   2. equation\_1 = {M=v(U)}  
      eq’\_1 <- equation\_1{}= equation\_1  
      case 4.1 is true:  
      s<- {M=v(U)}
   3. equation\_1<- G=v(M)  
      eq’\_1 = equation\_1s= G=v(v(U))  
      case 4.1 is true:  
      s<- {M=v(U), G=v(v(U))}
   4. equation\_1<- U=v(G)  
      eq’\_1= U=v(G)s = U=v(v(v(U) FAIL-Circular definition is not allowed
3. Unify [ s ( [ v | [ [ v | V ] | A ] ] ) ,   
    s ( [ v | [ v | A ] ] ) ]  
   initialization: s={}, Equation = [s ( [ v | [ [ v | V ] | A ] ] ) = s ( [ v | [ v | A ] ] ) ]
   1. equation\_1 <- Equations.pop() = s ( [ v | [ [ v | V ] | A ] ] ) = s ( [ v | [ v | A ] ] )  
      eq’\_1 = equation\_1{}= equation\_1   
      case7 is true: splitting equation to little equations  
      Equations <- [[ v | [ [ v | V ] | A ] ]= [ v | [ v | A ] ]]
   2. equation\_1 <- Equations.pop() = [ v | [ [ v | V ] | A ] ]= [ v | [ v | A ] ]  
      eq’\_1 = equation\_1{}= equation\_1   
      case7 is true: splitting equation to little equations  
      Equations <- [ [ v | A ]= [ [ v | V ] | A ]]
   3. equation\_1 <- Equations.pop() = [ [ v | A ]= [ [ v | V ] | A ]  
      eq’\_1 = equation\_1{}= equation\_1   
      case7 is true: splitting equation to little equations  
      Equations <- [v=[v|V]] – FAIL- atomic value ‘v’ can’t be equal compound value

Q3.3

* 1. תמונה של העץ למעלה
  2. {X=zero, Y=s(zero)}, {X=s(zero}, Y=zero}
  3. There is a success path from root to leaf, therefore it’s a success tree.   
     As written in class material  
     
  4. All paths from root to leavess are finite, therefor it’s a finite tree