<u>Assignment 5</u>

<u>Submittors: Moran Neptune 208474544 and Bar Perlman 305026882</u>

<u>Part 1: Theoretical Questions:</u>

2. C.
?- female(Y), relatives(joseph, Y)

<u>3.</u>
<u>A.</u> Unify[**mn**(ttt,M,mn,V,HO,mn,br(M)), **mn**(ttt,br(HO),mn,br(M),b r(V),mn,br(M))]
      1. s = {}
        Eqs = [mn(ttt,M,mn,V,HO,mn,br(M)) = mn(ttt,br(HO),mn,br(M),b
            r(V),mn,br(M))]
      2. s = {}
        Eqs = [~~ttt=ttt~~, M=br(HO), ~~mn=mn~~, V=br(M), HO=br(V), ~~mn=mn~~,
           ~~br(M)=br(M)~~]
      3. s = {} · {M=br(HO)} = {M=br(HO)}
        Eqs = [V=br(br(HO)), HO=br(V)]
      4. s = {M=br(HO)} · { V=br(br(HO))}
        Eqs = [HO=br(br(br(HO)))]

לכן, ההצבה לא אפשרית, הקשירות (HO=br(br(br(HO)) הינה מעגלית.
<u>B.</u> Unify[**m**(M,N), **n**(M,N)]

מכיוון שלא מדובר באף בדיקה מהאלגוריתם (משום שלא מדובר ב-
משתנה וקבוע, או בשני המשוואות יש אטומים שווים, ומשום שהסמל של שניהם שונה
(m <--> n)
אזי נחזיר שגיאה.
<u>C.</u> Unify[**kmr**(pt(pt),V,pt,g,kmr(TF),pt), **kmr**(pt(V),V,pt,g,kmr(TF),HO)]
      1. s = {}
        Eqs = [kmr(pt(pt),V,pt,g,kmr(TF),pt) = kmr(pt(V),V,pt,g,kmr(TF),HO)]
      2. s = {}
        Eqs = [pt(pt)=pt(V), ~~V=V~~, ~~pt=pt~~, ~~kmr(TF)=kmr(TF)~~, pt=HO]
      3. s = {} · {pt(pt)=pt(V)} = {pt(pt)=pt(V)} = {pt=V}
        Eqs = [V=HO]
      4. s = {pt=V} · {V=HO} = {pt=HO}
        Eqs = []
      5. s = {pt=HO}
<u>D.</u> Unify[**mn**([T]), **mn**(T)]
      1. s = {}
        Eqs = [mn([T]) = mn(T)]
      2. s = {}
        Eqs = [[T]=T]
      לכן, ההצבה לא אפשרית, הקשירות T=[T] הינה מעגלית.

<u>E.</u> Unify[**g**([br | [KK | K]]), **g**([[br | KK] | K])]

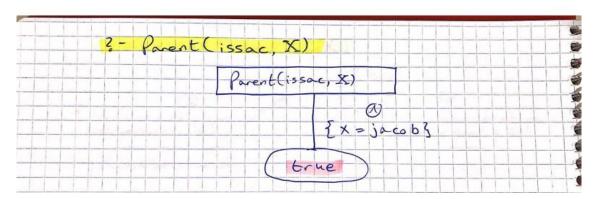   1. s = { }
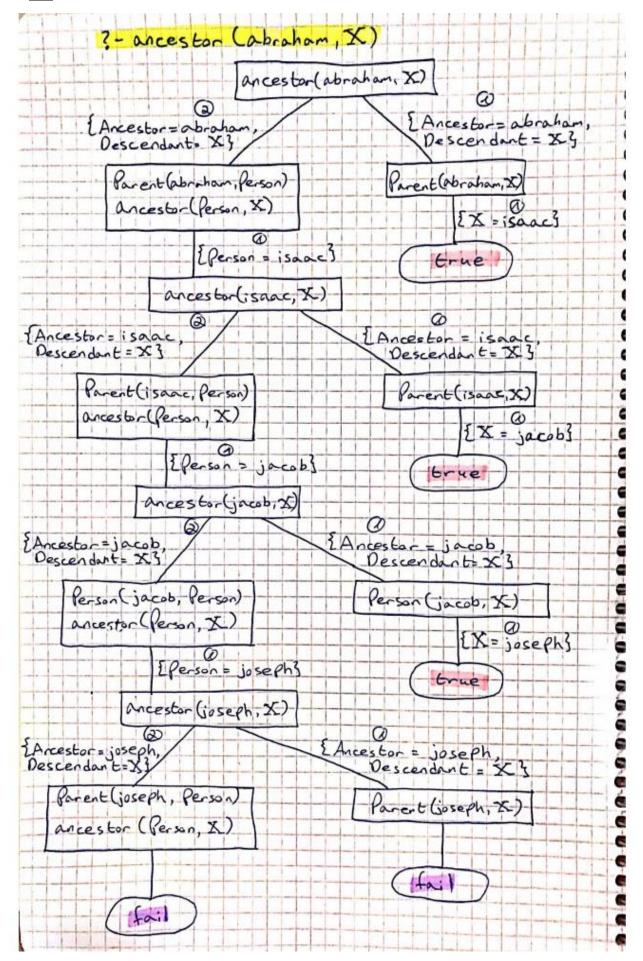   Eqs = [**g**([br | [KK | K]]) = **g**([[br | KK] | K])]
      2. s = { }
         Eqs = [[br | [KK | K]] = [[br | KK] | K]]
      3. s = { } · {br=[br|KK]}
         Eqs = []KK|k]=K]

. לכן, ההצבה לא אפשרית, הקשירות [br=[br|KK   הינה מעגלית .

.4

:A.1

?- ancestor (abraham, X)

ancestor(abraham, X)

② {Ancestor = abraham, Descendant = X}

① {Ancestor = abraham, Descendant = X}

Parent(abraham, Person)
ancestor(Person, X)

Parent(abraham, X)

① {X = isaac}

true

① {Person = isaac}

ancestor(isaac, X)

② {Ancestor = isaac, Descendant = X}

① {Ancestor = isaac, Descendant = X}

Parent(isaac, Person)
ancestor(Person, X)

Parent(isaac, X)

① {X = jacob}

true

① {Person = jacob}

ancestor(jacob, X)

② {Ancestor = jacob, Descendant = X}

① {Ancestor = jacob, Descendant = X}

Person(jacob, Person)
ancestor(Person, X)

Person(jacob, X)

① {X = joseph}

true

① {Person = joseph}

Ancestor(joseph, X)

② {Ancestor = joseph, Descendant = X}

① {Ancestor = joseph, Descendant = X}

Parent(joseph, Person)
ancestor(Person, X)

Parent(joseph, X)

fail

fail

<u>B:</u> 1. X= Jacob

X = Isaac .2

X = Jacob

X = Joseph

<u>C:</u> These trees are finite, because they are not infinite - there is not an infinite route,
.no route is from the version of: p(X) :- p(Y), q(X, Y)

<u>D:</u> These trees are success proof trees. Because, there is at least one success route in
.each tree

<u>A5..</u>

SC --> AtomC | CompoundC | lambdaC

AtomC --> NumberC | BooleanC | StringC

CompoundC --> regularFormC

'regularFormC --> '(' StringC SC* ')

'lambdaC --> '(' 'lambda' '(' StringC* ')' '(' SC* ')' '('SC*')' ')

NumberC --> <Number**>**

BooleanC --> #t | #f

StringC --> <String**>**