PPL – Assignment 4

2.1) הגנרטורים G1, G2 **שווים** אם לכל מספר טבעי n≥1 הערך ה-n-י שמחזיר G1 שווה לערך ה-n-י שמחזיר G2.

2.3)

נגדיר:

הערך ה-n-י שמחזיר Fib1: Fib1n.

הערך ה-n-י שמחזיר Fib2: Fib2n.

*נוכיח את הטענה:*

**לכל מספר טבעי n≥1 הערך ה-n-י שמחזיר Fib1 שווה לערך ה-n-י שמחזיר Fib2.**

נסמן:

הוכחה באינדוקציה על n:

בסיס: עבור n=1 – תנאי עצירה של Fib1, כלומר: Fib11 = 1.

*אז אכן מתקיים:* Fib11 = Fib21 = 1.

הנחת האינדוקציה: נניח שעבור כל מס טבעי n>1 מתקיים Fib1n = Fib2n*.*

צעד האינדוקציה: נוכיח Fib1n+1 = Fib2n+1.

נשים לב ש- מקיימים את המשוואה: x2 – x – 1 =0.

*לכן עבור כל* n>1 *מקיימים את המשוואות הבאות:*

אז סה"כ קיבלנו:

מ.ש.ל

3.1) טענה: הפרוצדורה append$ שקולה-CPS לפרוצדורה append.

נוכיח כי לכל רשימה lst1 שאורכה מספר טבעי n ולכל continuation (יסומן cont) מתקיים:

(append$ lst1 lst2 cont) = (cont (append lst1 lst2))

נסמן את אלגוריתם applicative-eval כ-a-e.

נוכיח באינדוקציה על n:

בסיס האינדוקציה: עבור n=0 (כלומר המקרה בו lst1 רשימה ריקה):

a-e[ (append$ lst1 lst2 cont) ] =>\* a-e[ (cont lst2) ]

a-e[ (cont (append lst1 lst2)) ] =>\* a-e[ (cont lst2) ]

הנחת האינדוקציה: נניח שעבור רשימה lst1 שאורכה n>0 מתקיים:

(append$ lst1 lst2 cont) = (cont (append lst1 lst2))

צעד האינדוקציה: נוכיח כי עבור רשימה באורך n+1 הטענה מתקיימת.

a-e[ (append$ (cons x lst1) lst2 cont) ] =>\* a-e[ (append$ lst1 lst2 (lambda (v) (cont (cons x v)))) ]

מהנחת האינדוקציה:

a-e[ (append$ lst1 lst2 (lambda (v) (cont (cons x v)))) ] =

a-e[ (lambda (v) (cont (cons x v))) (append lst1 lst2) ] =>\* a-e[ (cont (cons x (append lst1 lst2))) ]

מצד שני:

a-e[ (append (cons x lst1) lst2) ] = a-e [ (cons x (append lst1 lst2)) ]

a-e[ (cont (append (cons x lst1) lst2)) ] = a-e[ (cont (cons x (append lst1 lst2))) ]

אז בסך הכל קיבלנו:

a-e[(append$ (cons x lst1) lst2 cont)] =>\* a-e[(cont (cons x (append lst1 lst2)))]

a-e[(cont (append (cons x lst1) lst2))] =>\* a-e[(cont (cons x (append lst1 lst2)))]

אז לכל רשימה lst1 שאורכה מספר טבעי n ולכל continuation (יסומן cont) מתקיים:

(append$ lst1 lst2 cont) = (cont (append lst1 lst2))

ולכן הפרוצדורה append$ שקולה-CPS לפרוצדורה append.

מ.ש.ל

5.1)

a) unify[t(s(s), G, s(U), p, t(K), s),

t(s(G), G, K, p, t(K), U)]

s(s) , s(G) 🡺 s = G

s(U) , K 🡺 K = s(U)

s , U 🡺 s = U

substitution = { U = s, G = s, K = s(s) }

b) unify[p([[W | V] | [V | k]]),

p([[v | V] | W])]

W , v 🡺 v=W

W , [V | k] 🡺 W = [V | k]

Subtitition { v = [V |k] } is not possible since v is a constant and [V | k] is composite.