Assignment 6

Q.1

(b) נוכיח באינדוקציה על n, גודל הרשימה lst1

הוכחה:

מקרה בסיס: n=0. במקרה זה lst1 היא הרשימה הריקה, לכן לפי מקרה הבסיס הפונקציה append$ תחזיר (cont lst2), זאת בדומה לפונקציית append שבמקרה הבסיס תחזיר את lst2 כנדרש.

הנחה: נניח כי לכל lst1 באורך n הטענה מתקיימת, נוכיח עבור כל רשימה lst1 באורך n+1:

תהי lst1 ונניח כי אורכה n+1. צ"ל: (append$ lst1 lst2 cont) = (cont (append lst1 lst2)).

$$(append\$\ lst1\ lst2\ cont) \underset{calc}{\Longrightarrow} \left(append\$\ (cdr\ lst1)\ lst2\ \left(lambda\ (res)\ \left(cont\ (cons\ (car\ lst1)\ res)\right)\right)\right)$$

$$\xRightarrow[induction]{} \left(\left(lambda\ (res)\ \left(cont\ (cons\ (car\ lst1)\ res)\right)\right)\ (append\ (cdr\ lst1)\ lst2)\right)$$

$$\xRightarrow[substitution]{} \left(cont\ \left(cons\ (car\ lst1)\ (append\ (cdr\ lst1)\ lst2)\right)\right)$$

$$\xRightarrow[reverse\ calc]{} \left(cont\ (append\ lst1\ lst2)\right)$$

Q.2

(d) The reduce1-lzl is the same as the regular reduce taught in class and will be effective when we'll want a final answer of finite list.
The reduce2-lzl will be effective when we want a partial reduction of a list or when the activation is on infinite list.
The reduce3-lzl will be effective when we'll want the history of the reduction.

(g) The pi-sum implementation taught in class calculate the approximate of pi by using reduction for each number of the approximation. For example, for approximate 10 we need to calculate the approximate of 9, 8 until the first. In lazy list, we can compute the next element in a faster way because we always have access to the previous element.

Q.3

3.1

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

ניצור משוואה: $t(s(s),G,s,p,t(K),s) = t(s(G),G,s,p,t(K),U)$

ונריץ את אלגוריתם פתרון המשוואות:

<u>אתחול</u>

אוסף המשוואות: $[t(s(s),G,s,p,t(K),s) = t(s(G),s,G,s,p,t(K),U)]$
הצבה: {}

בחירת משוואה מהאוסף: $t(s(s),G,s,p,t(K),s) = t(s(G),G,s,p,t(K),U)$

'פרשנות' המשוואה על ידי הפעלת ההצבה הנוכחית על שתי אגפיה:
$t(s(s), G, s, p, t(K), s) = t(s(G), G, p, s, t(K), U)$

'מקרה ג' – שני האגפים הם ביטויים מורכבים בעלי אותו מבנה (אותו פרדיקט, אותו מספר פרמטרים)
← פירוק למשוואת קטנות יותר

אוסף המשוואות: $[s(s) = s(G), G = G, s = s, p = p, t(K) = t(K), s = U]$
הצבה: {}

בחירת משוואה מהאוסף: $s(s) = s(G)$
'פרשנות' המשוואה על ידי הפעלת ההצבה הנוכחית על שתי אגפיה: $s(s) = s(G)$

'מקרה ג' – שני האגפים הם ביטויים מורכבים בעלי אותו מבנה (אותו פרדיקט, אותו מספר פרמטרים)
← פירוק למשוואת קטנות יותר

אוסף המשוואות: $[G = G, s = s, p = p, t(K) = t(K), s = U, s = G]$
הצבה: {}

בחירת משוואה מהאוסף: $t(K) = t(K)$
'פרשנות' המשוואה על ידי הפעלת ההצבה הנוכחית על שתי אגפיה: $k = k$

אוסף המשוואות: $[s = U, s = G]$
הצבה: {}

'מקרה ב' – אחד הצדדים הוא משתנה לוגי
← הוספת המשוואה להצבה: $\{\} \circ \{U = s\} = \{U = s\}$

אוסף המשוואות: $[s = G]$
הצבה: $\{U = s\}$
בחירת משוואה מהאוסף: s=G

'מקרה ב' – אחד הצדדים הוא משתנה לוגי
← הוספת המשוואה להצבה: $\{s = G\} \circ \{U = s\} = \{U = s, G = s\}$

אוסף המשוואות: []
הצבה: $\{U = s, G = s\}$

← המאחד הכללי ביותר (mgu) הוא ההצבה $\{U = s, G = s\}$
$unify[t(s(s), G, s, p, t(K), s), t(s(G), G, s, p, t(K), U)] = t(s(s), s, s, p, t(K), s)$

b. $unify\left(p\left([v|[V|w]]\right), p\left([[v|V]|W]\right)\right)$

ניצור משוואה: $p\left([v|[V|w]]\right) = p\left([[v|V]|W]\right)$
ונריץ את אלגוריתם פתרון המשוואות:

<u>אתחול</u>

אוסף המשוואות: $[p\left([v|[V|w]]\right) = p\left([[v|V]|W]\right)]$
הצבה: {}

בחירת משוואה מהאוסף: $p\left([v|[V|w]]\right) = p\left([[v|V]|W]\right)$
'פרשנות' המשוואה על ידי הפעלת ההצבה הנוכחית על שתי אגפיה:

$p([v|[V|w]]) = p([[v|V]|W])$

'מקרה ג' – שני האגפים הם ביטויים מורכבים בעלי אותו מבנה (אותו פרדיקט, אותו מספר פרמטרים)
← פירוק למשוואת קטנות יותר

אוסף המשוואות: $\left[[v|[V|w]] = [[v|V]|W]\right]$
הצבה: {}

בחירת משוואה מהאוסף: $[v|[V|w]] = [[v|V]|W]$
'פרשנות' המשוואה על ידי הפעלת ההצבה הנוכחית על שתי אגפיה: $[v|[V|w]] = [[v|V]|W]$

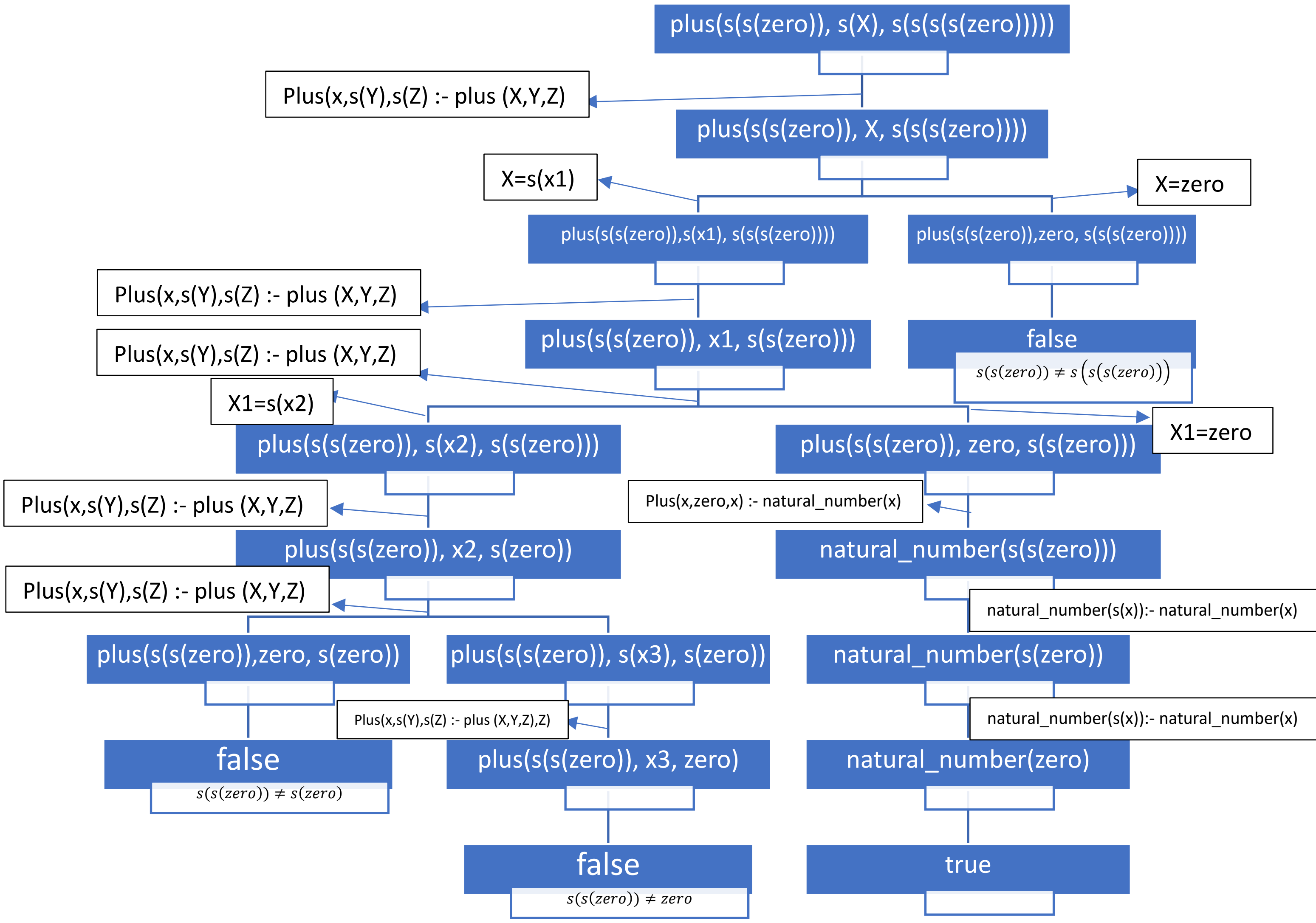
'מקרה ג' – שני האגפים הם ביטויים מורכבים בעלי אותו מבנה (אותו פרדיקט, אותו מספר פרמטרים)
← פירוק למשוואת קטנות יותר

אוסף המשוואות: $[v = [v|V], [V|w] = W]$
הצבה: {}

בחירת משוואה מהאוסף: $v = [v|V]$
'פרשנות' המשוואה על ידי הפעלת ההצבה הנוכחית על שתי אגפיה: $v = [v|V]$

נשים לב שמשתנה הטיפוס משמאל נמצא במשתנה הטיפוס מימין ולכן לפי האלגוריתם קיבלנו שגיאה.

3.3


plus(s(s(zero)), s(X), s(s(s(s(zero)))))
Plus(x,s(Y),s(Z) :- plus (X,Y,Z)
plus(s(s(zero)), X, s(s(s(zero))))
X=s(x1)
X=zero
plus(s(s(zero)),s(x1), s(s(s(zero))))
plus(s(s(zero)),zero, s(s(s(zero))))
Plus(x,s(Y),s(Z) :- plus (X,Y,Z)
plus(s(s(zero)), x1, s(s(zero)))
false
$s(s(zero)) \neq s(s(s(zero)))$
Plus(x,s(Y),s(Z) :- plus (X,Y,Z)
X1=s(x2)
X1=zero
plus(s(s(zero)), s(x2), s(s(zero)))
plus(s(s(zero)), zero, s(s(zero)))
Plus(x,s(Y),s(Z) :- plus (X,Y,Z)
Plus(x,zero,x) :- natural_number(x)
plus(s(s(zero)), x2, s(zero))
natural_number(s(s(zero)))
Plus(x,s(Y),s(Z) :- plus (X,Y,Z)
natural_number(s(x)):- natural_number(x)
plus(s(s(zero)),zero, s(zero))
plus(s(s(zero)), s(x3), s(zero))
natural_number(s(zero))
Plus(x,s(Y),s(Z) :- plus (X,Y,Z),Z)
natural_number(s(x)):- natural_number(x)
false
$s(s(zero)) \neq s(zero)$
plus(s(s(zero)), x3, zero)
natural_number(zero)
false
$s(s(zero)) \neq zero$
true