עבודת בית 5

מגישים: עומרי גולדברג 208938985, נעה חרדון 312164668

Question 1:

1.1 Recursive to Iterative CPS Transformations

b. proof that append$ is CPS-equivalent to append:

- (define append$
- (lambda (lst1 lst2 cont)
- (if (empty? lst1)
- (cont lst2)
- (append$ (cdr lst1)
-     lst2
-     (lambda (cdr-lst1) //new-cont
-         (cont (cons (car lst1) cdr-lst1))))))

נוכיח זאת באמצעות אינדוקציה על אורך הרשימה הראשונה (lst1).

<u>מקרה בסיס:</u> כאשר אורך הרשימה lst1 הוא 0
במקרה זה מה שיתקבל מהפונקציה append הוא (append '() lst2) = lst2 ובמקביל מהפונקציה append$ יתקבל (append$ lst1 lst2 cont) = (cont lst2)

<u>צעד:</u> נניח שהטענה נכונה עבור רשימה lst1 באורך n, נוכיח עבור רשימה lst1 באורך n+1

- על פי הקוד הערך של (append lst1 lst2) הוא: (cons (car lst1) (append (cdr lst1) lst2))
- על פי הקוד הערך של (append$ lst1 lst2 cont) הינו (append$ (cdr lst1) lst2 new-cont) כאשר new-cont מסומן כהערה בlambda בשתי השורות האחרונות בקוד של append$.

מכיוון ש- (cdr lst1) זוהי רשימה שאורכה n = n + 1 – 1 ניתן לומר כי הנחת האינדוקציה מתקיימת ומכך נובע ש- (new-cont (append (cdr lst1) lst2)) = (append$ (cdr lst1) lst2 new-cont).

כעת נחליף את הערך new-cont בערכו האמיתי ונקבל את הדבר הבא:

(cont (cons (car lst1) (append (cdr lst1) lst2))) = (append$ (cdr lst1) lst2 new-cont)

כעת נשתמש במה שציינו קודם ונוכל לומר כי: (cont (append lst1 lst2)) = (append$ lst1 lst2 cont)

Question 2:

d. **<u>reduce1-lzl</u>** is useful for finite lazy lists, otherwise we will get into an infinite loop, for example we can use reduce1-lzl for calculating the sum of numbers in a finite lazy list.
<u>reduce2-lzl</u> is useful for infinite lazy lists with a specific number of elements, for example we can use reduce2-lzl for calculating the sum of the first 5 numbers in a infinite series.
<u>reduce3-lzl</u> is useful when we don't need the whole reduced values of the list immediately, we can use reduce3-lzl when we want to access separately to elements.

g. **advantage of pi generator**: we can divide our calculations in a way which we can "save" old computations (getting the n+1 item from the lazy list requires the generation of one item and adding it to the previous sum, as opposed to calculating the whole approximation with pi-sum). It's helpful in situations where we don't know how accurate we need to be and might need to improve our calculations later on.

disadvantage of pi generator: if we know the accuracy, in pi generator it will require multiple calls to the lazy list instead of calling pi-sum once. In that case pi-sum is more efficient.

Question 3:

3.1 Unification

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

step 1: $s\ =\ \{\}, equations\ =[t(s(s),G,s,p,t(K),s)\ ,t(s(G),G,s,p,t(K),U)]$

שני הפרדיקטים שווים בשם ובמספר הארגומנטים לכן נשווה בין כל זוג ארגומנטים מתאימים במיקום.

step 2: $s\ =\ \{\}, equations\ =\ [s(s)=s(G),\ G=G,\ s=s,\ p=p,\ t(K)=T(K),\ s\ =\ U]$

step 3: $s\ =\ \{\}\circ\{s(s)=s(G)\}\ =\ \{s(s)=s(G)\}=\{s=G\}, equations\ =\ [U=s]$

step 4: $s=\{G=s\}\circ\{U=s\}=\{U=s,G=s\}, equations=[\quad]$

step 5: $s\ =\ \{G=s,U=s\}$

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

step 1: $s\ =\ \{\}, equations\ =\ [p([v\ |\ [V\ |\ W]])\ ,p([[v\ |\ V]\ |\ W])]$

step 2: $s\ =\ \{\}, equations\ =\ [[v\ |\ [V\ |\ W]]\ ,[[v\ |\ V]\ |\ W]]$

שני הפרדיקטים שווים בשם ובמספר הארגומנטים לכן נשווה בין כל זוג ארגומנטים מתאימים במיקום.

step 3: $s\ =\{\}\circ\{v\ =\ [v\ |\ V]\}, equations\ =\ [[V\ |\ W]\ =\ [W]]$

we got a fail because the placement is impossible, we receive that $v\ =\ [v\ |\ V]$ is cyclic.

3.3 Proof tree

$?-plus(s(s(zero)), s(X), s(s(s(s(zero))))).$

$\{X1 = s(s(zero)), s(Y1) = s(X),$

$s(Z1) = s\left(s\left(s(s(zero))\right)\right),$ %p2

$Z1 = s\left(s(s(zero))\right)\}$

$plus(s(s(zero)), Y1, s(s(s(zero))))$

$\{X2 = s(s(zero)), X = zero,$

$X4 = s(s(s(zero)))\},$ %p1

Fail

$X2 = s(s(zero)) \neq s(s(s(zero))) = X4$

$\{X2 = s(s(zero)), s(Y2) = Y1,$

$s(Z2) = s(s(s(zero)))\},$ %p2

$plus(s(s(zero)), Y2, s(s(zero)))$

$\{X3 = s\left(s(zero)\right), Y2 = s(Y3),$

$s(Z3) = s(s(zero))$ %p2}

$plus(s(s(zero)), Y3, (s(zero)))$

$\{X4 = s\left(s(zero)\right), Y3 = s(Y4),$

$s(Z4) = s(zero)$ %p2}

$plus(s(s(zero)), Y4, zero)$

Fail

$s(s(zero)) \neq zero$

$\{X3 = X4 = s(s(zero)), Y2 = zero,$ %p1}

$natural\_number(s(s(zero)))$

$\{X4 = s(zero)\},$ %n2

$natural\_number(s(zero))$

$\{X5 = zero\},$ %n2

$natural\_number(zero)$

%n1

true

בסופו של דבר נקבל כי

$X = s(zero)$