**Proof append$**

**טענה:** הפרוצדורה append$ שקולה cps לפרוצדורה append. כלומר, עבור כל רשימה lst1 ועבור כל רשימה lst2 ולכל continuation אשר יסומן ע"י cont מתקיים :

**הוכחה:** כיוון שהפרוצדורה append היא רקורסיבית, ההוכחה מתבצעת ע"י שימוש באינדוקציה.

* בסיס האינדוקציה:
* הנחת האינדוקציה: עבור רשימה lst1 באורך k כך ש , כלומר
* צעד האינדוקציה: יהא ,

עבור lst1 באורך n

מהנחת האינדוקציה, נקבל:

**שאלה 2d**

* נשתמש בreduce1-lzl כאשר נתונה לנו רשימה עצלה שאנחנו יודעים שהגודל שלה הוא סופי ונרצה להפעיל פונקציית reduce לחישוב ערך מצטבר בעזרת פונקצייה מסויימת.
* נשתמש בreduce2-lzl כאשר נרצה לחשב ערך מצטבר של רשימה, אך לא נדע בהכרח כי הגודל שלה הוא סופי, ולכן נגביל את מספר האיברים.

בנוסף, נשתמש כאשר נרצה לחשב ערך מצבר על n האיברים הראשונים ברשימה.

* נשתמש בreduce3-lzl כאשר נרצה לדעת מהו הערך המצטבר בחישוב של רשימה עבור מספר אינדקסים, ולכן נקרא לפונקצייה.

הפונקצייה תחזיר לנו רשימה עצלה שמכילה חישוב חדש על כל אחד מאיברי הרשימה, כאשר החישוב מבוסס על פונקציית הreducer שקיבלנו בקריאה לreduce3-lzl

**שאלה 2g**

* יתרונות-

1. יתרון ברור במימוש הפונקציה generate-pi-approximation על פני המימוש pi-sum שראינו בכיתה הוא שניתן להחליט איזה קירוב לערך של נרצה, וכך לחסוך חישובים מיותרים ולהקטין את זמן הריצה.
2. בעזרת שימוש בפונקציה זו נוכל לדעת את הקירוב שנרצה עבור כל שלב בתהליך, ולא רק את התוצאה הסופית.

* חסרונות-
  + בזבוז זיכרון עבור רשימה גדולה, במקום זיכרון עבור ערך החזרה יחיד שמחושב בpi-sum.

**שאלה 3-**

סעיף א'-

**unify[t(s(s), G, s, p, t(K), s), t(s(G), G, s, p, t(K), U)]**

שלב א': initialize

sub: {}

Equitation:

|  |
| --- |
|  |

שלב ב'

sub: {}

Equitation:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

שלב ג'

sub: {}

Equitation:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |

שלב ד'

sub: {}

Equitation:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

שלב ה'

sub: {}

Equitation:

|  |
| --- |
|  |
|  |
|  |
|  |

שלב ו'

sub: {}

Equitation:

|  |
| --- |
|  |
|  |
|  |

שלב ז'

sub: {}

Equitation:

|  |
| --- |
|  |
|  |
|  |

שלב ח'

sub: {}

Equitation:

|  |
| --- |
|  |
|  |

שלב ט'

sub:

Equitation:

|  |
| --- |
|  |

שלב י'

sub:

Equitation:

סעיף ב'

שלב א': initialize

sub: {}

Equitation:

|  |
| --- |
|  |

שלב ב'

sub: {}

Equitation:

|  |
| --- |
|  |

שלב ג'

sub: {}

Equitation:

|  |
| --- |
|  |
|  |

שלב ד'

נקבל את המשוואה הבאה:

נשים לב כי המשוואה לא עונה על אף אחד מהתנאים באלגוריתם, ולכן נכנס אל הelse הסופי שמחזיר fail.

**סעיף 3.3**

נמספר את המשוואות:

plus(X, zero, X) :- natural\_number(X). %p1

plus(X, s(Y), s(Z)) :- plus(X, Y, Z). %p2

natural\_number(zero). %p3

natural\_number(s(X)) :- natural\_number(X). %p4

Diagram

Description automatically generated