:1.1.b

נוכיח כי לכל n טבעי וכל cont -פונקציית המשך אשר תסומן ע"י c מתקיים :

(append$ lst1 lst2 c) =(c (append lst1 lst2)

נוכיח באינדוקציה על האורך של lst1.

בסיס: n = 0.

(append$ ‘() lst2 c) =>(c lst2) =(c (append ‘() lst2)

יהי n .

הנחת האינדוקציה: לכל k<n טבעי כך ש |lst1|=k הטענה מתקיימת.

צעד: |lst1|=n

(append$ lst1 lst2 c) => (append$ (cdr lst1) lst2 (lambda (res) (c (cons (car lst1) res)))

מהנחת האינדוקציה נקבל :

(lambda (res) (c (cons (car lst1) res)))((append (cdr lst1) lst2)) =>

(c (cons (car lst1) (append (cdr lst1) lst2))) = (c (append lst1 lst2))

:2.d

Reduce-lz1 – נוכל להשתמש במקרה זה כאשר אין חשש לריצה אינסופית על הרשימה העצלה שאנו מקבלים כקלט. במקרה זה ידוע לנו בוודאות כי קיים סוף לרשימה ולכן בהכרח לא נגיע לריצה אינסופית ונסיים את הריצה.

Reduce-lz2 – כאשר אין אנו בטוחים אם נקבל כקלט רשימה עצלה סופית נרצה להשתמש במקרה זה. ע"פ מימוש פונקציה זו היא תרוץ מספר סופי של פעמים לבחירתנו. לכן בנוסף לחשש מרשימה עצלה אינסופית נוכל להשתמש בה כאשר נרצה להגביל את מספר הצעדים.

Reduce-lz3 – נרצה להשתמש בפעולה זו כאשר אנו בטוחים כי לא נקבל רשימה עצלה אינסופית כקלט (בהתאם ל-reduce-lz1) ובנוסף נשתמש בפונקציה זו כאשר נרצה לראות כפלט את כל שלבי החישוב עד הגעה לתשובה הסופית.

:2.g

יתרון – מכיוון שמימשנו ע"י רקורסיית זנב לא יהיה מצב של עומס על המחסנית מכיוון שכל פעם יהיה במחסנית איבר אחד לעומת המימוש pi-sum שעומס ברקורסיה ראש ולפיכך עלול להיווצר מצב של stack overflow עקב עומס של קריאות רקורסיביות.

חיסרון – במימוש נאלצנו להשתמש בבנייה מורכבת של פונקציות עזר המשתמשות ברשימות עצלות הדורשות מימוש דקדקני ומורכב (כלומר, מבנה נותנים חדש). דבר זה הינו מסובך למימוש ומהווה מקור נוסף לשגיאות טיפוס. בנוסף הקוד עלול להיות קשה לקריאה עבור גורם צד ג' זאת בניגוד לפונקציה pi-sum שהינה אינטואיטיבית לכתיבה והבנה.

: Unification – 3.1

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

- S = { }

  A*s = t(s(s), G, s, p, t(K), s)

  B*s = t(s(G), G, s, p, t(K), U)

- S = { G = s }
  A*s = t(s(s), s, s, p, t(K), s)

  B*s = t(s(s), s, s, p, t(K), U)

- S = { G = s , U = s }
  A*s = t(s(s), s, s, p, t(K), s)
  B*s = t(s(s), s, s, p, t(K), s)

הביטוי a תקין וההצבה ה-MGU היא S = { G = s, U = s }

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

- S = { }
  A*s = p([v | [V | W]])
  B*s = p([[v | V] | W])

  FAIL – v != [v | V] not the same structure.

: Proof tree – 3.3