פתרון שאלת PROC – 2024ב – א2

השינויים הנדרשים:

1. נרחיב את הדקדוק של השפה (lang.scm) כדי לכלול את הדקדוקים החדשים:

      (expression

        ("(" expression (arbno expression) ")")

        call-exp)

      (expression

        ("overload proc" (arbno "[" (separated-list identifier ",") "]" "->" expression ";"))

        overload-proc-exp)

כמו כן, נמחוק את ביטוי ה-proc-exp הישן.

1. נגדיר טיפוס נתונים חדש: overload-proc-val

(define-datatype expval expval?

(num-val

(value number?))

(bool-val

(boolean boolean?))

**(overload-proc-val**

**(ids (list-of identifier?))**

**(bodies (list-of expression?))**

**)**

)

כשהטיפוס החדש מכיל רשימת ids – הפרמטרים ורשימת bodies – תוכן הפרוצדורות.

1. נגדיר פונקציות עזר רלוונטיות בקובץ interp.scm:

* פונקציה ראשונה – extend-env\* - מרחיבה את הסביבה באופן רקורסיבי עם יותר ממשתנה וכריכה אחת.

  (define extend-env\*

    (lambda (params args env)

      (if (null? params)

          env

          (extend-env\* (cdr params) (cdr args)

                      (extend-env (car params) (car args) env)))))

* פונקציה שנייה – zip – יוצרת רשימה חדשה שמורכבת מזוגות של המקומות המתאימים של 2 הרשימות שניתנו לה

(define zip

(lambda (lst1 lst2 result)

(if (or (null? lst1) (null? lst2)) result

(zip (cdr lst1) (cdr lst2) (cons (cons (car lst1) (car lst2)) result))

))  
)

* פונקציה המחלצת מתוך expval את הערכים של overload-proc עם שימוש בcases

  (define expval->overload-proc

    (lambda (v)

      (cases expval v

        (overload-proc-val (ids bodies)

          (list ids bodies))

        (else (expval-extractor-error 'overload-proc v))

      )))

* הפונקציה המחלצת מתוך רשימה zipped את הפונקציה הרלוונטית לקריאה לפי כמות הפרמטרים שניתנו

  (define get-overload-proc

    (lambda (zipped rands-length)

      (cond

        ((null? zipped)

        (eopl:error 'get-overload-proc "No matching overload found"))

        ((equal? (length (car (car zipped))) rands-length)

        (car zipped)) ; return the (param-list . body) pair

        (else

        (get-overload-proc (cdr zipped) rands-length)))))

* הטיפול ב-overload-proc-exp – בניית ביטוי מתאים

(overload-proc-exp (ids bodies)

(overload-proc-val ids bodies))

* הטיפול ב-call-exp החדש

        (call-exp (rator rands)

          (let\* ((overload-proc (expval->overload-proc (value-of rator env)))

                (ids          (car overload-proc))

                (bodies       (cadr overload-proc))

                (zipped       (zip ids bodies '()))

                (args         (map (lambda (x) (value-of x env)) rands))

                (matched-pair (get-overload-proc zipped (length rands)))

                (extended-env (extend-env\* (car matched-pair) args env)))

            (value-of (cdr matched-pair) extended-env)))

הסבר כללי:

1. מחשבים בעזרת value-of את הrator ומתוכו מחשבים ביטוי overload-proc-val. הערך היוחזר הוא רשימה שמורכבת מה-ids וה-bodies ולכן מחלצים אותם על ידי car ו- cadr.
2. משתמשים בפרוצדורה zip בשביל ליצור רשימה שבה המשתנים של כל פונקציה כרוכים לbody שלה – הרשימה zipped.
3. מחשבים את הפרמטרים שניתנו לפונקציה בקריאה לcall-exp עם value-of, מרחיבים את הסביבה עם הרשימה שיצרנו באמצעות map של ערכים אלו עם extend-env\* רקורסיבי על הרשימה
4. משתמשים בפונקציה get-overload-proc כדי להשיג את הפונקציה הרלוונטית על סמך כמות הפרמטרים שניתנו בעת הקריאה
5. ובסוף – מחשבים את הערך של body של הפונקציה הרלוונטית עם הסביבה המורחבת שיצרנו