EOPL Pset 4

Alireza Habibzadeh `99109393`

2

از آنجایی که که زبان letrec را هم با تغییر دادن همین زبان let ساختیم، یک کلک منطقی آن است که ابتدا همان تغییرات را اعمال کنیم تا زبان به letrec تبدیل شود، سپس در این زبان جدید که دارای توابع بازگشتی است فاکتوریل را به سادگی پیاده کنیم. البته باید استفاده از تعریف توابع را تنها به خودمان محدود کنیم تا دقیقا خواسته‌ی سوال ارضا شود.

اما روش دیگر این است که مانند سایر expressionها مثل `diff-exp` ، تایپ جدیدی به نام مثلا `fact-exp` به زبان اضافه کنیم.

$$\text{diff-exp} \quad : \quad Exp \rightarrow Exp$$

ابتدا تابع را در زبان racket برای استفاده‌ی خودمان تعریف می‌کنیم: (یا از تابع آماده‌ی racket استفاده می‌کنیم.)

```
#lang racket

(define (! n)
  (if (= n 0) 1
      (* n (! (sub1 n)))))
```

تغییر در تعریف datatype expression

```
(define-datatype expression expression?
  (fact-exp
    (exp1 expression?))
  ; .
  ; .
  ; . other expressions
  )
```

تغییر در تابع `value-of`

در واقع این جایی است که مقدار فاکتوریل واقعا محاسبه می‌شود. در همه‌ی برنامه تا قبل از evaluation فاکتوریل‌ها به صورت lazy `fact-exp` باقی می‌مانند.

```
#lang racket
```

```
(define value-of
  (λ (exp env)
    (cases expression exp
      (diff-exp (exp1)
        (let ([val1 (value-of exp1 env)])
          (let ([num1 (expval->num val1)])
            (num-val (! num1)))))
      ; .
      ; .
      ; . other expressions
      )))
```