Pár příkladů na Scheme - řešení

m@rtlin

14. června 2013

Verze 1.1.1

Abstrakt

Na přání studentů předmětu *Paradigmata Programování 1* na *Katedře informatiky Přírodovědecké fakulty Univerzity Palackého v Olomouci* jsem sestavil tento skromný seznam příkladů k procvičení ke zkoušce.

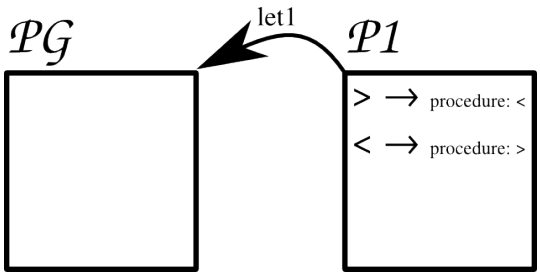
O případných chybách nebo jiných připomínkách mě prosím informujte. Za správnost neručím.

1 Prostředí a vazby

Napište, na co se vyhodnotí následující výrazu a rozkreslete vazby mezi vzniklými prostředími.

Příklad 1

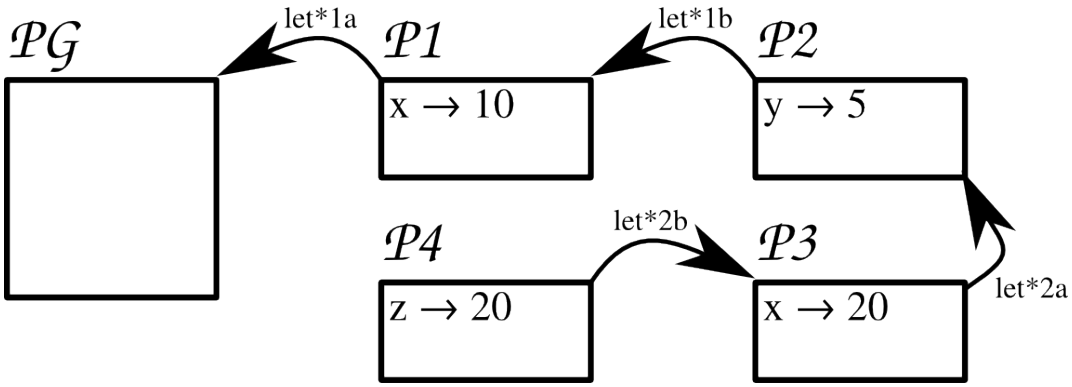
```
(let1 ((> <)
       (< >))
   (< 4 8))
```



V prostředí $P1$ se vyhodnotí tělo $let1$ a to tedy na hodnotu $\#f$.

Příklad 2

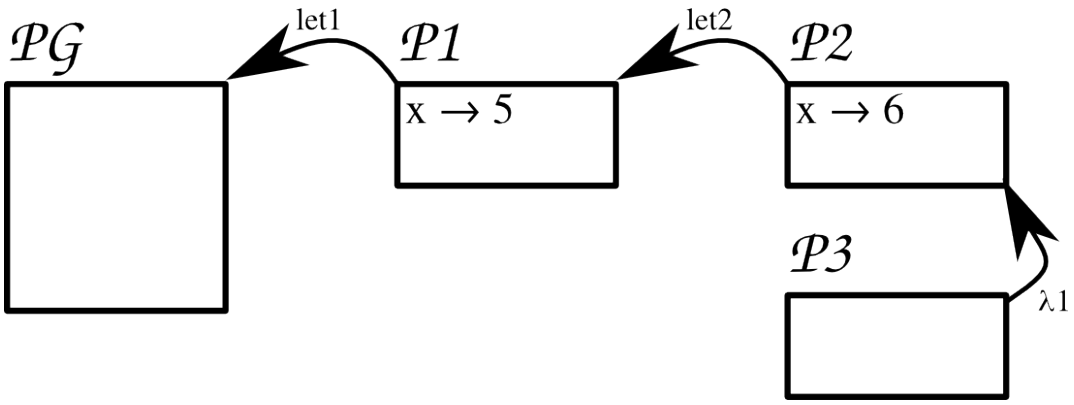
```
(let*1 ((x 10)
        (y 5))
  (let*2 ((x 20)
          (z x))
    (+ y z)))
```



V prostředí $P4$ se vyhodnotí tělo a to na hodnotu 25.

Příklad 3

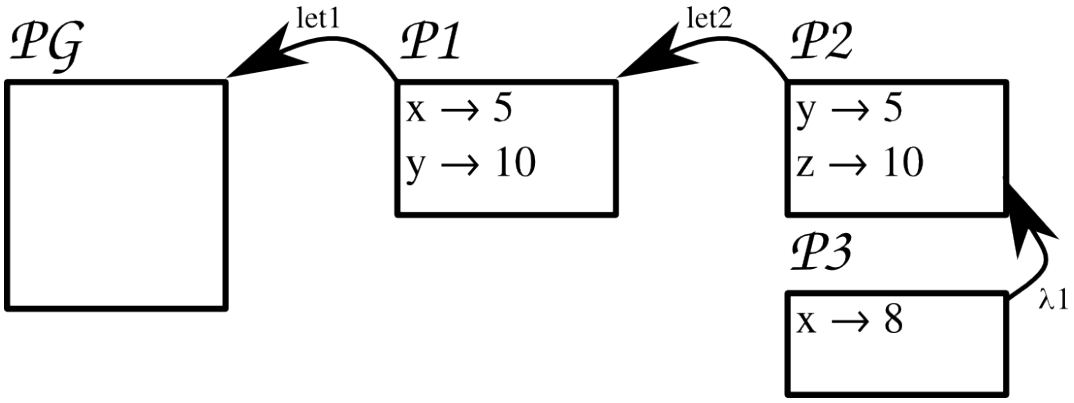
```
((let1 ((x 5))
   (let2 ((x (+ x 1)))
      (lambda1 ()
        (- x 10)))))
```



Tělo *let*2 se vyhodnotí na $\lambda_1 =< (), (- \ x \ 10), P2 >$, jejímž zavoláním vznikne $P3$ a vyhodnotí se na -4.

Příklad 4

```
(let1 ((x 5)
       (y 10))
   ((let2 ((y x)
           (z y))
      (lambda1 (x)
        (list x y z)))
    8))
```

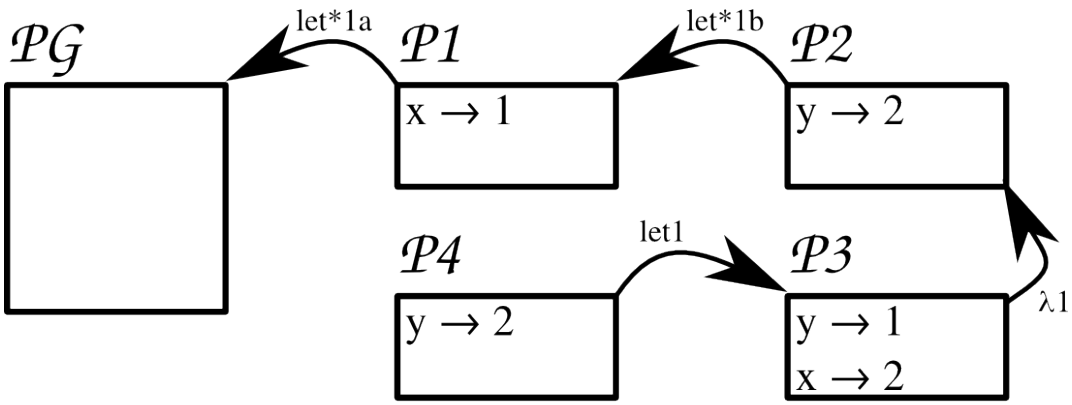


Tělo *let*2 se vyhodnotí na $\lambda_1 =< (x), (list \ x \ y \ z), P2 >$, jejímž zavoláním vznikne $P3$ a její tělo se v tomto prostředí vyhodnotí na (8 5 10).

Příklad 5

```
(let*1 ((x 1)
        (y 2))
   ((lambda1 (y x)       ;pozor, zde jsou x a y prohozene
     (+ y
        (let1 ((y x))
          (* y x))))
    x
```

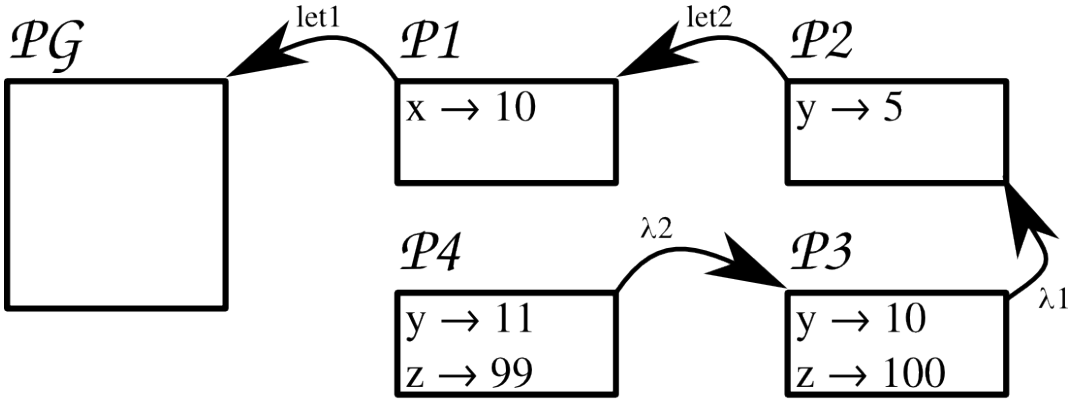
```
y))
```



Tělo let^*1 se vyhodnotí na $\lambda_1 =< (yx), (+\ y\ (let1\ ((y\ x))\ (*\ y\ x))), P2 >$, jejímž zavoláním vznikne $P3$ a v tomto prostředí se vyhodnocujej jeí tělo. Ta si ještě navíc vytvoří prostředí $P4$ a následně vrátí výsledek 5.

Příklad 6

```
(let1 ((x 10))
  ((let2 ((y 5))
    (lambda1 (y z)
      ((lambda2 (y z)
        (cons y z))
       (+ x 1)
       (- z 1))))
    x
    (* 10 x)))
```



Tělo $let2$ se vyhodnotí na $\lambda_1 =< (y\ z), ((lambda2\ (y\ z)\ (cons\ y\ z))), P2 >$, jejímž zavoláním vznikne $P3$ a v něm vyhodocením těla λ_1 vznikne $\lambda_2 =< (yz), (cons\ y\ z), P3 >$, která je inhed zavolána a vyhodnotí se na $(11\ .\ 99)$.

2 Boxové notace

Zakreslete logickou a fyzickou reprezentaci výsledku vyhodnocení následujících výrazů v boxové notaci.

Příklad 1

```
'(4 . ((8. 9) (10 . (3 . (8 . 9)))))
```

Vyhodnotíme:

```
(4 . ((8 . 9) (10 . (3 . (8 . 9)))))
```

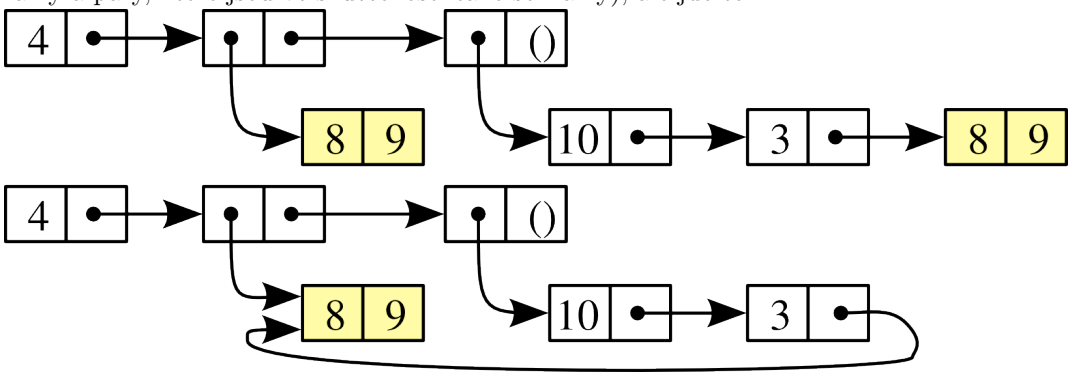
Nyní by bylo poněkud vhodné si kód přepsat buď do „seznamové“

```
(4 (8 . 9) (10 3 8 . 9))
```

nebo „párové notace“

```
(4 . ((8. 9) . ((10 . (3 . (8 . 9))) . ()))
```

Pokud tak neučiníte, máte práci mírně komplikovanější (hádají se vám tam seznamy a páry, které jsou ve skutečnosti také seznamy), ale jde to:



Příklad 2

```
(cons (cons (cons 4
                  8)
            (cons 6
                  (cons 1
                        5)))
      (cons (cons 1
                  5)
            (cons 4
                  8)))
```

Zde byl výraz velmi přehledně zapsán a tudíž by neměl být problém s tím, že se v něm ztratíte. Vzhledem k tomu, že obsahuje pouze a jenom příkazy *cons*, je teoreticky možné provést vykreslení diagramu přímo (jedno volání *cons* vytvoří jeden pár).

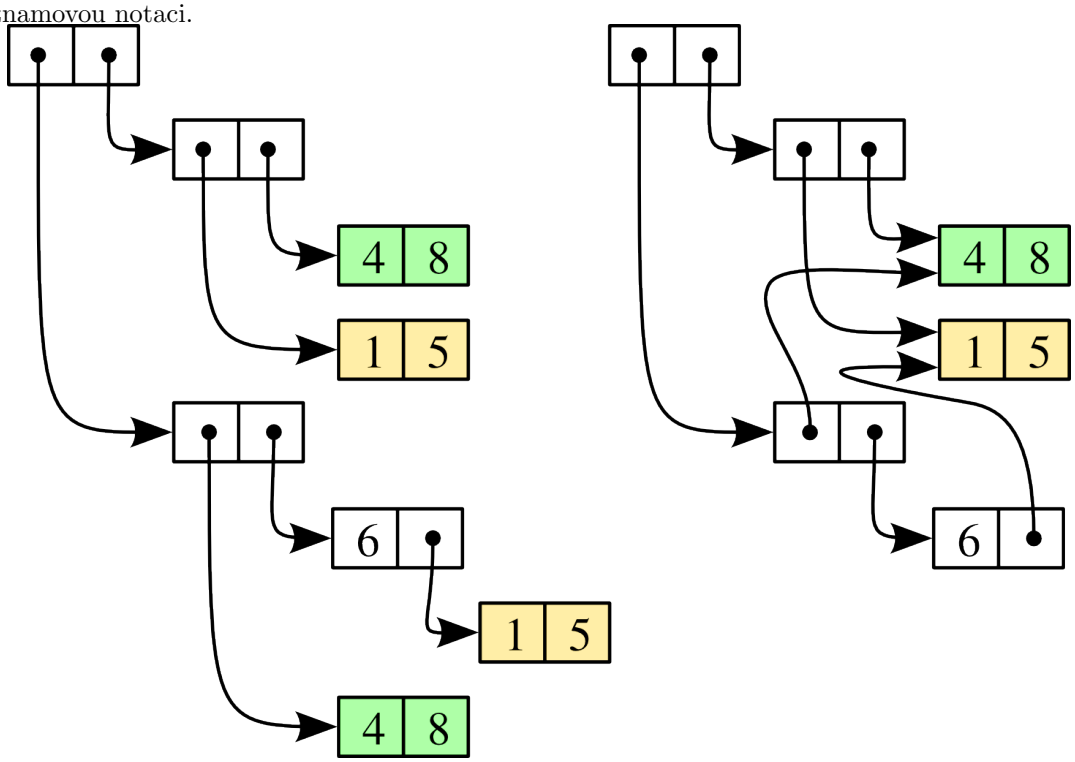
Každopádně vyhodnocení výrazu dopadne následovně (při zachování struktury původního výrazu):

```
(((4 .
   8) .
  (6 .
   (1 .
    5))) .
 ((1 .
   5) .
  (4 .
   8)))
```

Po přepsání pak:

```
(((4 . 8) . (6 . (1 . 5))) . ((1 . 5) . (4 . 8)))
```

Již na první pohled je vidět, že výsledek neobsahuje žádné seznamy (ted neobsahuje žádné prázdné seznamy - ()) a je proto víc než zbytečné uvažovat seznamovou notaci.



Příklad 3

```
(list (list list (list (list list) list) (list list)
  (list (list (list list) list)) list (list) (list (list))) list)
```

Zde je přepis víc než nutný:

```
(list (list list
            (list (list list)
                  list)
            (list list)
```

```
            (list (list (list list)
                        list))
            list
            (list)
            (list (list)))
      list)
```

Nyní můžeme výraz vyhodnotit. Pro jednoduchost používám notaci $\#l$ pro $\#procedure : list$:

```
((#l
  ((#l)
   #l)
  (#l)
  (((#l)
    #l))
  #l
  ()
  ((())
 #l)
```

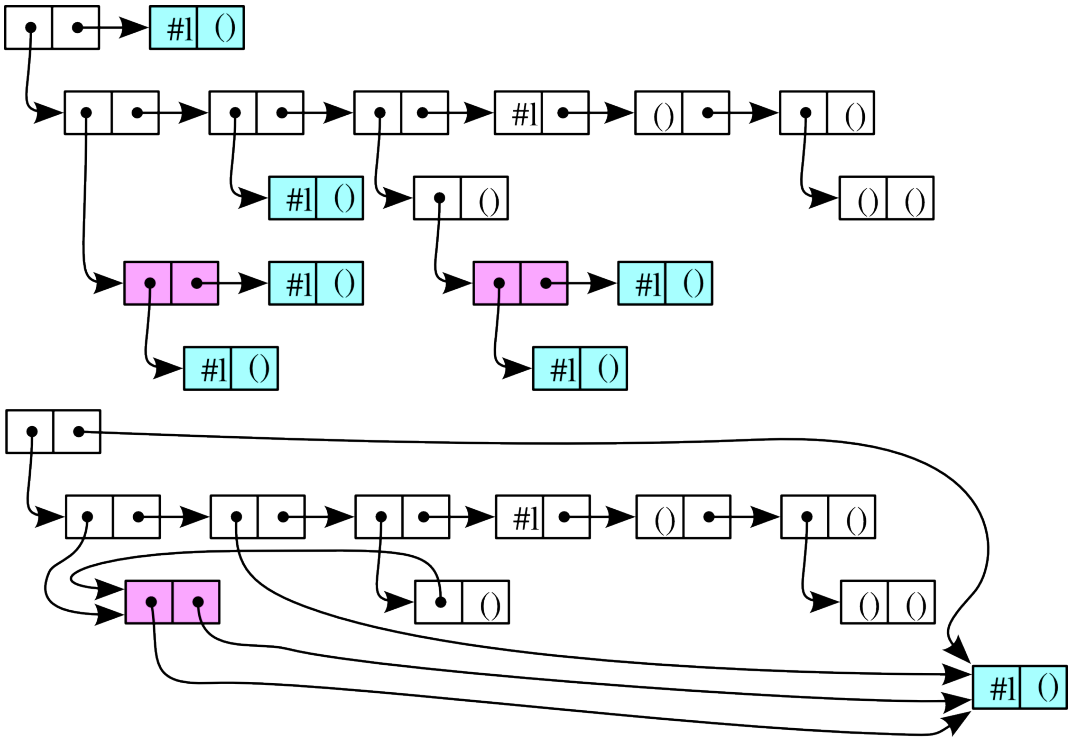
Nyní je vhodné si povšimnout, že se jedná a o dvouprvkový seznam, jehož prvním prvkem je sedmiprvkový seznam. Lehce názorněší je tento přepis:

```
((#l
  ((#l) #l)
  (#l)
  (((#l) #l))
  #l
  ()
  ((())
 #l)
```

Pro úplnnost doplním, jak by výraz vypadal zapsán na jeden řádek:

```
((#l ((#l) #l) (#l) (((#l) #l)) #l () ((())) #l)
```

Překreslení do boxové notace se udělá jednoduše, pokud si představíte, jak jsou seznamy do sebe povnořovány, a uvědomíte, jak se zakresluje seznam:



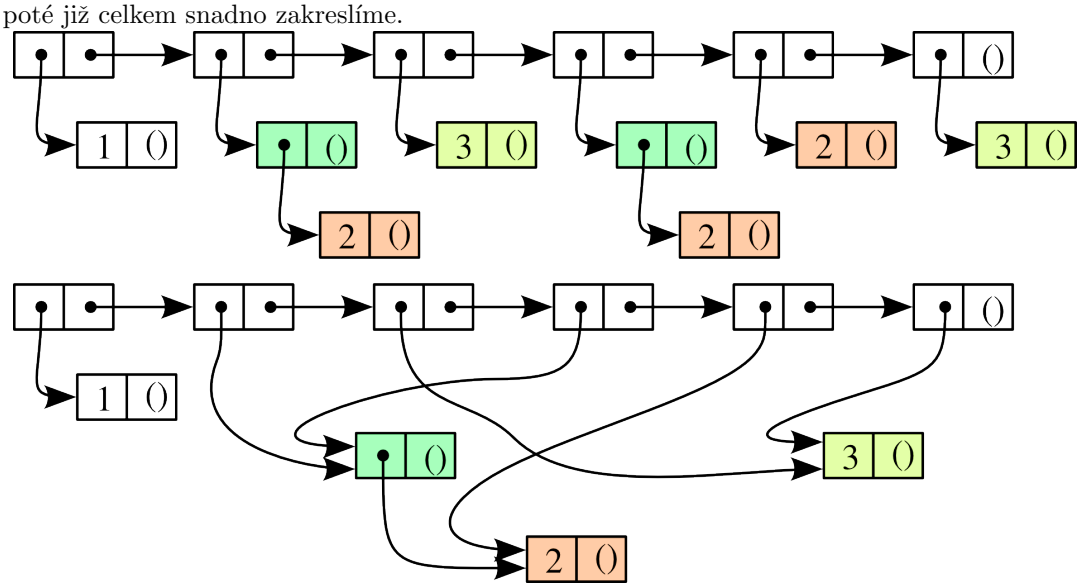
Příklad 4

```
(append (map list (list 1 (cons 2 (list)) 3 '(2))) (list (list 2) '(3)))
```

Vyhodnotíme.

```
((1) ((2)) (3) ((2)) (2) (3))
```

A poté již celkem snadno zakreslíme.



Příklad 5

```
`((4) ,(list 6 5 4) '(4) ,@(map list 6 5 4) (list 4))
```

Vyhodnocení:

```
((4) (6 5 4) (quote (4)) (6) (5) (4) (#1 4))
```

Opět zakreslíme vlemi snadno: