

Мутиращи операции

Трифон Трифонов

Функционално програмиране, 2023/24 г.

29 ноември 2023 г.

Тази презентация е достъпна под лиценза Creative Commons Признание-Некомерсиално-Споделяне на споделеното 4.0 Международен 

Мутиращи операции в Scheme

Мутиращите операции в Scheme позволяват въвеждането на **странични ефекти**.

Преглед:

- `set!` — промяна на оценка, свързана със символ
- `set-car!`, `set-cdr!` — промяна на компоненти на точкови двойки
- `begin` — последователност от действия
- `open-input-file`, `open-output-file` — работа с файлове
- `read`, `write`, `display` — вход и изход

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>
 - В противен случай — **грешка!**
- Примери:
 - (define a 2) $a \rightarrow 2$
 - (set! a 5) $a \rightarrow 5$
 - (define (sum x) (begin (set! a (+ a x)) a))
 - (sum 10) $\rightarrow 15$
 - (sum 10) $\rightarrow 25$
 - **губи се референциалната прозрачност!**

Пример: текуща сметка

```
(define (make-account sum)
  (lambda (amount)
    (if (< (+ amount sum) 0)
        (display "Insufficient funds!\n")
        (set! sum (+ sum amount)))
    sum))
```

- `(define account (make-account 100))`
- `(account 20) → 120`
- `(account -50) → 70`
- `(account -150) → "Insufficient funds"`
70

Промяна на компоненти (set-car!, set-cdr!)

- (set-car! <двойка> <израз>)
- (set-cdr! <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- Примери:
 - (define p (cons (cons 1 2) (cons 3 4)))
 - (set-car! p 7)
 - $p \rightarrow (7 . (3 . 4))$
 - (set-cdr! p '(5 3))
 - $p \rightarrow (7 5 3)$
 - (set-cdr! (cdr p) p)
 - $p \rightarrow (7 5 7 5 7 5 \dots)$