Scheme

Markus Läll

Veidi ajalugu

- Scheme on üks kahest suuremast Lisp'i dialektist.
- Lisp'i lõi John McCarthy pärast 1956. aastal AI-teemalisel "Dartmouth Summer Research Project"il käimist.
- Lisp'i iImplementeerimine algas 1958.
- Scheme tegid Gerald Jay Sussman ja Guy L. Steele'i soovist mõista paremini Carl Hewitt’i tegutsejate mudelit (*actor model*)
- Sellele järgnes hulk artikleid aastatel 1975-1978 koondnimega *The Lamda Papers* (veebis vabalt saadaval).

...pikemalt

- McCarthy: "The history of LISP" (1979)
- Steel, Gabriel: "The Evolution of Lisp" (1991?)

Standardiseerimine

R[n]RS = Revised[n] Report on the Algorithmic Language Scheme

Versioonid:

- 1978: R(1)RS (Steele)
- 1985: R2RS
- 1986: R3RS
- 1991: R4RS
- 1998: R5RS, praegu enim implementeeritud
- 12.08.2007: R6RS

Pakettide lülitumine standardisse toimub läbi SRFI'de (Scheme Requests for Implementation)

Koodi ülesehitus, tehete järjekord

- Kõik on esitatud liht- või liit- ehk nn s-avaldiste ehk *s-expression* 'itena,
- mis võivad olla üksteise sees => puu kuju.
- Iga liitavaldise esimene element on protseduur/funktsioon.
- Seega on avaldised esitatud prefiks kujul:
  - (*fn1* arg1 arg2 (fn2 arg3 arg4 ...) ...)
  - (+ 1 2 3 (* 4 5)) => 26
- Tehete järjekord on olemas niivõrd, kuivõrd see on esitatud sulgudega,
- üleliigsed sulud on keelatud.
- Kõik liit-avaldised on esitatud sulgude sees.
- Süntaks on BNF'is formaliseeritav.

Muutujad, omistamine, protseduurid

- Muutujad on "väärtuste asukohad" ja
- pole seotud tüüpidega;
- samas, kuna nad on asukohad, siis on nad ka "poole tee pealt" muudetavad.
  - (define kuus 6)
  - (set! kuus 7)
  - (+ kuus 7) => 14
- Kõik protseduurid on kulisside taga esitatud läbi lambda-vormide:
  - (lambda <vars> <exps>)
  - Viimase exp'i väärtus tagastatakse.
- Protseduurid:
  - (lambda (arg) (+ 1 arg))
  - ((lambda (arg) (+ 1 arg)) kuus) => 8
  - (define lisaÜks (lambda (arg) (+ 1 arg)))
  - (define (lisaÜks arg) (+ 1 arg))

Andmed

Kõik väärtused on objektid.
Tüübid:

- booleans,
- numbers (number tower),
  - complex
    - real
      - rational
        - integer
- strings (compound symbol),
- characters,
- symbols (atomic),
- lists,
- pairs,
- procedures,
- continuations,
- vectors ja

- ...ja lisaks implementatsiooni-spetsiifilised tüübid.
- Näiteks *Gambit* 'is on veel:
  - hashtables,
  - structures,
  - u8vectors,
  - uninterned symbols,
  - “foreign type”s,
  - jt

- Teisendused on on eksplitsiitsed:
  - (from->to arg)
  - (number->string 5)

=> "5"

Viidaruum, rekursioon

- skoop ehk viidaruum
  - staatiline ehk leksikaalne (static/lexical scoping).
  - dünaamilist võimalust polegi
  - on ühine nii protseduuridele kui muutujatele
- Klausuurid (*closures*)
- rekursioon, tsüklid ja *tail-call optimization*

```
(define (fact n)
  (define (fact2 n m)
    (if (= n 0)
        m
       (fact2 (- n 1) (* m n))))
  (fact2 n 1))
```

Andmete juhtimine, makrod

- if
  - (if (ńumber? x)
    (print "jee!")
    (print "oli jutt, et number tuleb!??"))
  - (if <test> <then> <else>)
- cond
  - (cond
    (<test1> <then1>)
    (<testn> <thenn>)
    (else <else>))
- makrod
  - (define-syntax def
    (syntax-rules ()
    ((def f (p ...) body)
    (define (f p ...) body))))
    => def = define

let on olgu

- Loob kohaliku viidaruumi oma koodiblokis
  - (let (

(<var1> <exp1>)
(<varn> <expn>))
<body>)

  - ...on sama mis
  - ((lambda

(<var1> ...<varn>)
<body>)
<exp1> <expn>)

  - näiteks
  - (define x 5)

(+ (let ((x 3))
(+ x (* x 10)))
x) => 38

Kontinuatsioonid

- jätkud ehk kontinuatsioon (*continuations*)

```
(call/cc
  (lambda
    (tahan-tagasi!)
    (for-each
      (lambda (x)
        (if (eq? #f x)
            (tahan-tagasi!)
            (print x)))
      '(1 2 3 4 5 6 #f 7 8 9) )))=> 123456
```

Objekt-orienteeritus

- Toimib läbi vastavate pakettide
- Näiteks:
  - Meroon
    - (define-class
      <class-name>
      <superclass-name>
      (<field-descriptions> ...)
      <class-options>)
    - (define-class Point Object (x y))
  - Tiny-CLOS
  - YASOS
  - ClosureTalk
  - jne jne

IO

- Toimub läbi portide (mis on üks algtüüpidest),
- selleks on vastavad protseduurid, näiteks:
  - open-input-file
  - open-output-file
  - read-char
  - write-char
  - eof-object?
  - jne

Linke

- Sussman
  - video, 1986: http://groups.csail.mit.edu/mac/classes/6.001/abelson-sussman-lectures/
  - õpik, "Structure and Interpretation of Computer Programs": http://mitpress.mit.edu/sicp/full-text/book/book.html
  - õpetus, "A Tour of Scheme in Gambit": http://dynamo.iro.umontreal.ca/~gambit/wiki/images/a/a7/A_Tour_of_Scheme_in_Gambit.pdf
- R5RS: http://schemers.org/Documents/Standards/R5RS/
- R6RS: http://www.r6rs.org/
- Gambit: http://dynamo.iro.umontreal.ca/~gambit/wiki/index.php/Main_Page
- PLT Scheme: http://www.plt-scheme.org/