Kirjed

- Kirjeid Scheme standardis pole, kuid saab realiseerida kasutades makrosid
- Allpool kirjeldatud kirjete defineerimise mehhanismi kasutamiseks `(load "drscheme-eopl.ss")`
- Kirjete defineerimine

  (`define-record` *name* (*field*$_1$ ... *field*$_n$))

- Definitsiooni tulemusena luuakse protseduurid:
  - kirje konstruktor `make-`*name*
  - kirje tüübipredikaat *name*`?`
  - väljade selektorid *name*`->`*field*$_i$

Näide: binaarpuud

- Binaarpuude spetsifikatsioon:

  $\langle tree \rangle$ ::= $\langle number \rangle$
  | ($\langle symbol \rangle$ $\langle tree \rangle$ $\langle tree \rangle$)

- Realisatsioon:

```
> (define-record interior (symbol left-tree
                                  right-tree))
> (define tree-1
    (make-interior 'a (make-interior 'b 1 2) 3))
> (interior->symbol tree-1)
a
```

Näide: binaarpuud

Puu lehtede summa leidmine:

```
(define leaf-sum
  (lambda (tree)
    (cond
      ((number? tree) tree)
      ((interior? tree)
            (+ (leaf-sum (interior->left-tree tree))
               (leaf-sum (interior->right-tree tree))))
      (else (error "leaf-sum: Invalid tree" tree)))))
```

Variantkirjed

- Tavaliselt realiseerime iga BNFi variandi eraldi kirjena (isegi siis, kui seal on ainult üks element)

```
(define-record leaf (number))
(define-record interior (symbol left-tree
                                 right-tree))
(define tree-a (make-interior 'a (make-leaf 2)
                                 (make-leaf 3)))
(define tree-b (make-interior 'b (make-leaf -1)
                                 tree-a))
(define tree-c (make-interior 'c tree-b
                                 (make-leaf 1)))
```

Variantkirjed

- Hargnemine variantkirjetel:

(`variant-case` *record-expression*
  (*name*$_1$ *field-list*$_1$ *expression*$_1$)
  ...
  (*name*$_n$ *field-list*$_n$ *expression*$_n$)
  (`else` *expression*$_{n+1}$))

Variantkirjed

Puu lehtede summa leidmine:

```
(define leaf-sum
  (lambda (tree)
    (variant-case tree
      (leaf (number) number)
      (interior (left-tree right-tree)
           (+ (leaf-sum left-tree)
              (leaf-sum right-tree)))
      (else (error "leaf-sum: Invalid tree" tree)))))
```

Variantkirjed

```
(define leaf-sum
 (lambda (tree)
  (let ((*rec* tree))
   (cond ((leaf? *rec*)
          (let ((number (leaf->number *rec*)))
           number))
     ((interior? *rec*)
      (let ((left-tree (interior->left-tree *rec*))
            (right-tree (interior->right-tree *rec*)))
       (+ (leaf-sum  left-tree) (leaf-sum right-tree))))
     (else (error "leaf-sum: Invalid tree" tree))))))
```

Abstraktne süntaks

- Konkreetne süntaks on inimkasutuseks mõeldud süntaks
- Abstraktne süntaks esitab konkreetsüntaksi struktuuri ilma ebaoluliste detailideta
- Konstantidega lambda-arvutus:

$\langle exp \rangle ::= \langle number \rangle$	`lit (datum)`
$\mid \langle varref \rangle$	`varref (var)`
$\mid$ `(lambda (`$\langle var \rangle$`)` $\langle exp \rangle$`)`	`lambda (formal body)`
$\mid$ `(`$\langle exp \rangle$ $\langle exp \rangle$`)`	`app (rator rand)`

Abstraktne süntaks

- Meil on konkreetne süntaks esitatud listi-struktuuridena
- Abstraktse süntaksi esitame variantkirjetena

```
(define-record  lit (datum))
(define-record  varref (var))
(define-record  lambda (formal body))
(define-record  app (rator rand))
```

- Konkreetsest abstraktsesse süntaksisse teisendamist nimetatakse parsimiseks
- Abstraktsest konkreetsesse süntaksisse teisendamist nimetatakse ilutrükiks (pretty printing)

Konstantidega lambda-avaldiste parser

```
(define parse (lambda (datum)
  (cond
    ((number? datum) (make-lit datum))
    ((symbol? datum) (make-varref datum))
    ((pair? datum)
       (if (eq? (car datum) 'lambda)
           (make-lambda (caadr datum)
                        (parse (caddr datum)))
           (make-app (parse (car datum))
                     (parse (cadr datum)))))
    (else (error "parse: Invalid syntax" datum)))))
```

Konstantidega lambda-avaldiste ilutrükk

```
(define unparse
  (lambda (exp)
    (variant-case exp
       (lit (datum) datum)
       (varref (var) var)
       (lambda (formal body)
         (list 'lambda (list formal) (unparse body)))
       (app (rator rand) (list (unparse rator)
                               (unparse rand)))
       (else (error "unparse: Invalid syntax" exp)))))
```

Järgmiseks korraks

- Lugeda läbi EOPL ptk. 3.4 – 3.7
  - ptk. 3.4.4 Kirjete realisatsioon
  - ptk. 3.6 Andmeabstraktsioon, lõplikud funktsioonid
- Järgmine loeng toimub 12. märtsil
- NB! 1. kodutöö lahenduste tähtaeg 14. märts