CPS transformatsioon

- CPS transformatsioon teisendab protseduuri ekvivalentseks protseduuriks, mis on saba vormis
- Kasutab kahte (vastastikku rekursiivset) abiteisendust:
  - $\langle\langle S \rangle\rangle$ teisendab avaldisi, mis on lihtsad; lisab avaldises S olevatele lambda-abstraktsioonidele ühe lisaparameetri (jätku)
  - $[\![E]\!][K]\!]$ teisendab avaldise E ja jätku K saba vormis olevaks avaldiseks

Lihtsate avaldiste teisendamine

- Lambda-abstraktsioonile lisatakse parameeter k, mida kasutatakse keha teisendamisel jätkuna:

$$\langle\langle(\texttt{lambda}\ (x_1\ \ldots\ x_n)\ E)\rangle\rangle \quad \Rightarrow \quad (\texttt{lambda}\ (x_1\ \ldots\ x_n\ k)\ [\![E]\!][k]\!])$$

- Teiste lihtsate avaldiste korral teisendatakse alamavaldised rekursiivselt:

$$\begin{aligned}
\langle\langle x\rangle\rangle &\Rightarrow x \\
\langle\langle(\textit{prim-op}\ S_1\ \ldots\ S_n)\rangle\rangle &\Rightarrow (\textit{prim-op}\ \langle\langle S_1\rangle\rangle\ \ldots\ \langle\langle S_n\rangle\rangle) \\
\langle\langle(\texttt{if}\ S\ S_1\ S_2)\rangle\rangle &\Rightarrow (\texttt{if}\ \langle\langle S\rangle\rangle\ \langle\langle S_1\rangle\rangle\ \langle\langle S_2\rangle\rangle) \\
\langle\langle(\texttt{let}\ ((v_1\ S_1)\ldots(v_n\ S_n))\ S)\rangle\rangle & \\
&\Rightarrow (\texttt{let}\ ((v_1\ \langle\langle S_1\rangle\rangle)\ldots(v_n\ \langle\langle S_n\rangle\rangle))\ \langle\langle S\rangle\rangle)
\end{aligned}$$

Saba vormi teisendamine

- Teisendus $[\![E]\!][\![K]\!]$ väljastab saba vormis avaldise, mis väärtustab avaldise E ja "saadab" selle väärtuse jätkule K
- Reegel (C_{simple}) — kui avaldis E on lihtne, siis "saadame" E väärtuse jätkule K

$$[\![E]\!][\![K]\!] \quad \Rightarrow \quad (K\ \langle\langle E \rangle\rangle)$$

- NB! Kui jätk K on kujul $(\texttt{lambda}\ (v)\ T)$, siis tekib uus reedeks; selle võib kohe ära lihtsustada, kuna muutuja v esineb avaldises T alati täpselt üks kord

Saba vormi teisendamine

- Reegel (C_{app}) — kui avaldis $E = (S_0 \ \ldots \ S_n)$, kus kõik alamavaldised S_i on lihtsad, siis anname jätku K protseduuri S_0 on viimaseks argumendiks

$$[\![(S_0 \ \ldots \ S_n)]\![K]\!] \quad \Rightarrow \quad (\langle\langle S_0 \rangle\rangle \ \ldots \ \langle\langle S_n \rangle\rangle \ K)$$

- NB! Kui S_0 tähistas algprogrammis n argumendilist protseduuri, siis nüüd tähistab $n+1$ argumendilist protseduuri, kuna teisendatud programmis on kõigil protseduuridel lisaparameeteriks jätk

Saba vormi teisendamine

- Reegel (C_{head}) — kui avaldises E on peapositsioonis olev vahetu alamavaldis H mitte-lihtne

$$[\![(\ldots H \ldots)][K]\!] \quad \Rightarrow \quad [\![H][(\texttt{lambda}\ (v)\ [\![(\ldots v \ldots)][K]\!])]\!]$$

- NB! v on uus muutuja
- Näide:

`[[(+ (f x) y)][k]]`	(C_{head})
$\Rightarrow$ `[[(f x)][(lambda (v) [[(+ v y)][k]])]]`	(C_{simple})
$\Rightarrow$ `[[(f x)][(lambda (v) (k (+ v y)))]]`	(C_{app})
$\Rightarrow$ `(f x (lambda (v) (k (+ v y))))`	

Saba vormi teisendamine

- Näide, kui H on mitmekordne aplikatsioon:

$[\![$`(+ (f (g x)) y)`$]\!][$`k`$]\!]$ (C_{head})

$\Rightarrow [\![$`(f (g x))`$]\!][$`(lambda (v)`$[\![$`(+ v y)`$]\!][$`k`$]\!]$`)`$]\!]$ (C_{simple})

$\Rightarrow [\![$`(f (g x))`$]\!][$`(lambda (v) (k (+ v y)))`$]\!]$ (C_{head})

$\Rightarrow [\![$`(g x)`$]\!][$`(lambda (w)`$[\![$`(f w)`$]\!][$`(lambda (v) (k (+ v y)))`$]\!]$`)`$]\!]$ (C_{app})

$\Rightarrow [\![$`(g x)`$]\!][$`(lambda (w) (f w (lambda (v) (k (+ v y)))))`$]\!]$ (C_{app})

$\Rightarrow$ `(g x (lambda (w) (f w (lambda (v) (k (+ v y))))))`

Saba vormi teisendamine

- Näide, kui on mitu mitte-lihtsat peapositsioonis olevat vahetut alamavaldist

$[\![$`(+ (f x) (g y))`$]\!][$`k`$]]$ (C_{head})

$\Rightarrow [\![$`(f x)`$]\!][$`(lambda (v)`$[\![$`(+ v (g y))`$]\!][$`k`$])]\!]$ (C_{app})

$\Rightarrow$ `(f x (lambda (v)` $[\![$`(+ v (g y))`$]\!][$`k`$]))$ (C_{head})

$\Rightarrow$ `(f x (lambda (v)` $[\![$`(g y)`$]\!][$`(lambda (w)`$[\![$`(+ v w)`$]\!][$`k`$])]\!]))$ (C_{app})

$\Rightarrow$ `(f x (lambda (v)` $[\![$`(g y)`$]\!][$`(lambda (w) (+ v w k))`$]\!]))$ (C_{app})

$\Rightarrow$ `(f x (lambda (v) (g y (lambda (w) (+ v w k)))))`

Saba vormi teisendamine

- Avaldis E võib olla ka näiteks if- või let-avaldis

$[\![$`(if (f x) 1 -1)`$]\!][$`k`$]]$ (C_{head})

$\Rightarrow [\![$`(f x)`$]\!][$`(lambda (v)`$[\![$`(if v 1 -1)`$]\!][$`k`$])]\!]$ (C_{app})

$\Rightarrow$ `(f x (lambda (v)` $[\![$`(if v 1 -1)`$]\!][$`k`$]))$ (C_{simple})

$\Rightarrow$ `(f x (lambda (v) (k (if v 1 -1))))`

$[\![$`(let ((z (f x))) (+ z z))`$]\!][$`k`$]]$ (C_{head})

$\Rightarrow [\![$`(f x)`$]\!][$`(lambda (v)`$[\![$`(let ((z v)) (+ z z))`$]\!][$`k`$])]\!]$ (C_{app})

$\Rightarrow$ `(f x (lambda (v)` $[\![$`(let ((z v)) (+ z z))`$]\!][$`k`$]))$ (C_{simple})

$\Rightarrow$ `(f x (lambda (v) (k (let ((z v)) (+ z z)))))`

Saba vormi teisendamine

- Reegel (C_{tail}) — kui avaldises E on kõik peapositsioonis asuvad alamavaldised lihtsad, aga mõni sabapositsioonis olev vahetu alamavaldis on mitte-lihtne

$$[\![(\texttt{if}\ S\ E_1\ E_2)]\!][K]\!] \quad \Rightarrow \quad (\texttt{if}\ \langle\langle S \rangle\rangle\ [\![E_1]\!][K]\!]\ [\![E_2]\!][K]\!])$$

$$[\![(\texttt{let}\ ((v_1\ S_1)\ldots(v_n\ S_n))\ E_0)]\!][K]\!]$$

$$\Rightarrow \quad (\texttt{let}\ ((v_1\ \langle\langle S_1 \rangle\rangle)\ldots(v_n\ \langle\langle S_n \rangle\rangle))\ [\![E_0]\!][K]\!])$$

- Analoogselt teiste (letrec, begin, ...) avaldiste jaoks
- NB! let-avaldise korral satub jätk K muutujate $v_1 \ldots v_n$ mõjupiirkonda. Kui K sisaldab mõnda neist muutujatest vabalt, siis tuleb vastavad muutujad let-avaldises enne teisendamist ümbernimetada

CPS transformatsioon — näited

- Faktoriaal (uuesti)

```
(define fact
  (lambda (n)
    (if (zero? n) 1 (* n (fact (- n 1))))))
```

CPS transformatsioon — näited

- Faktoriaal (uuesti)

```
(define fact
  (lambda (n)
    (if (zero? n) 1 (* n (fact (- n 1))))))


(define fact-cps
  (lambda (n k)
    (if (zero? n)
        (k 1)
        (fact-cps (- n 1)
          (lambda (v) (k (* n v)))))))
```

CPS transformatsioon — näited

```
(define remove (lambda (s los)
   (if (null? los)
       '()
       (if (eq? s (car los))
           (remove s (cdr los))
           (cons (car los) (remove s (cdr los)))))))
```

CPS transformatsioon — näited

```
(define remove (lambda (s los)
   (if (null? los)
       '()
       (if (eq? s (car los))
           (remove s (cdr los))
           (cons (car los) (remove s (cdr los)))))))

(define remove-cps (lambda (s los k)
   (if (null? los)
       (k '())
       (if (eq? s (car los))
           (remove-cps s (cdr los) k)
           (remove-cps s (cdr los)
              (lambda (v) (k (cons (car los) v))))))))
```

Järgmiseks korraks

- Lugeda läbi EOPL ptk. 8.4 – 8.8
  - NB! raamatus on natuke teistsugused reeglid
- 2. kodutöö lahenduste tähtaeg 25. aprill