Struktuursed andmetüübid

Laiendame funktsionaalset keelt PuF andmestruktuuridega:

ennikud:

$$\begin{array}{rcl} e & ::= & \ldots \mid (e_0, \ldots, e_{k-1}) \mid \#j\ e \\ & \mid & \textbf{let}\ (x_0, \ldots, x_{k-1}) = e_1\ \textbf{in}\ e_0 \end{array}$$

listid:

$$\begin{array}{rcl} e & ::= & \ldots \mid [\,] \mid (e_1 : e_2) \\ & \mid & (\textbf{case}\ e_0\ \textbf{of}\ [\,] \to e_1;\ h : t \to e_2) \end{array}$$

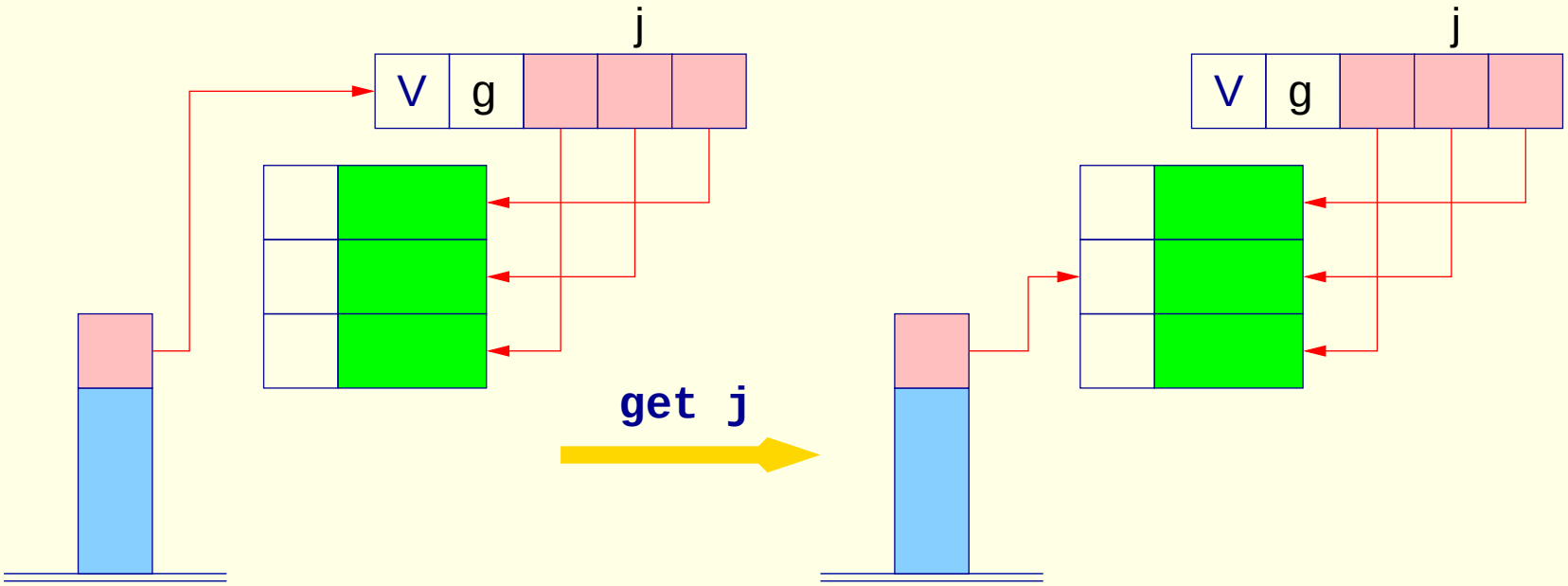
Struktuursed andmetüübid

Ennikute konstrueerimisel paigutatakse viidad komponentidele magasini ja seejärel luuakse vektor.
Konkreetse komponendi kättesaamiseks väärtustatakse ennik vektoriks ja seejärel väljastatakse vektori vastava indeksiga element.

$$
\begin{aligned}
\text{code}_V\ (e_0, \ldots, e_{k-1})\ \rho\ \text{sd} \quad &= \quad \text{code}_C\ e_0\ \rho\ \text{sd} \\
&\qquad \text{code}_C\ e_1\ \rho\ (\text{sd}+1) \\
&\qquad \ldots \\
&\qquad \text{code}_C\ e_{k-1}\ \rho\ (\text{sd}+k-1) \\
&\qquad \texttt{mkvec k} \\
\text{code}_V\ (\#j\ e)\ \rho\ \text{sd} \quad &= \quad \text{code}_V\ e\ \rho\ \text{sd} \\
&\qquad \texttt{get j}
\end{aligned}
$$

CBV korral väärtustatakse komponendid otse code_V abil.

Struktuursed andmetüübid




```
if (S[SP]→tag = V)
    S[SP] = S[SP]→v[j];
else Error ("Not Vector");
```

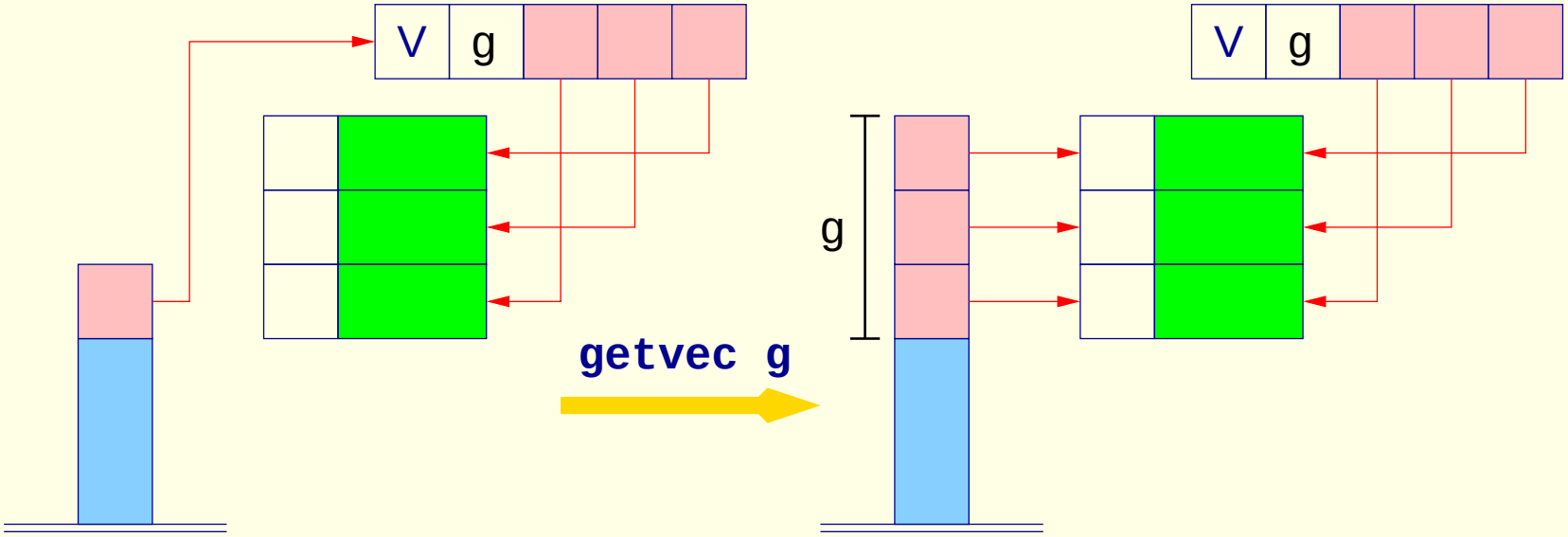
Struktuursed andmetüübid

Kõikide komponentide kättesaamiseks väärtustatakse ennik vektoriks ja seejärel lisatakse magasini viidad kõigile komponentidele.

$$\text{code}_V\ (\textbf{let}\ (y_0,\ldots,y_{k-1}) = e_1\ \textbf{in}\ e_0)\rho\ \text{sd} \quad = \quad \begin{array}{l} \text{code}_V\ e_1\ \rho\ \text{sd} \\ \texttt{getvec k} \\ \text{code}_V\ e_0\ \rho'\ \text{sd} \\ \texttt{slide k} \end{array}$$

kus $\rho' = \rho \oplus \{y_i \mapsto \text{sd} + i \mid i = 0,\ldots,k-1\}$.

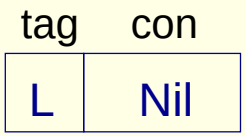
Struktuursed andmetüübid



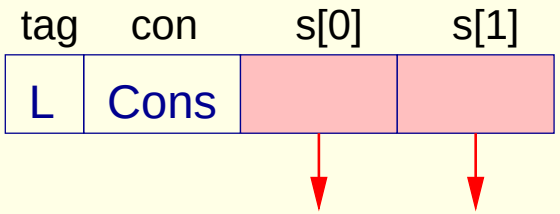

```
if (S[SP]→tag = V) {
    h = S[SP]; SP–;
    for (i=0; i<g; i++) {
        SP++; S[SP] = h→v[i];
    }
} else Error ("Not Vector");
```

Struktuursed andmetüübid

Listi konstruktorite esitamiseks kuhjas on uued objektid:



Empty List



Non−empty List

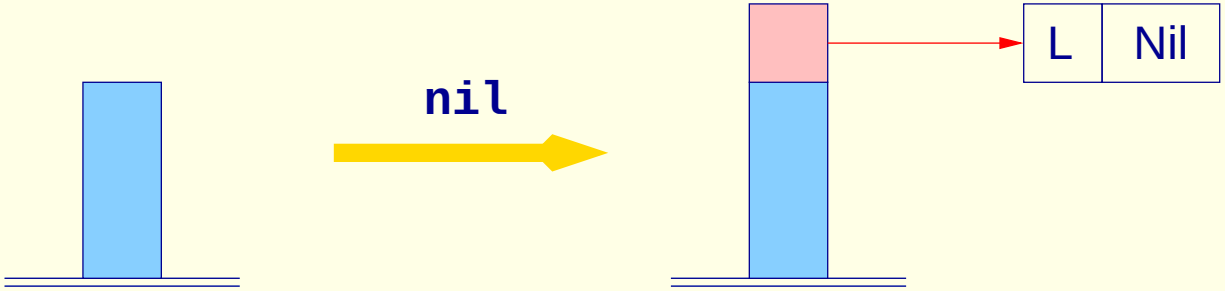
Struktuursed andmetüübid

Listi konstrueerimisel väärtustatakse argumendid (kui neid on; so. ":" korral) ning luuakse kuhjas konstruktorile vastav objekt:

$$
\begin{array}{lcl}
\text{code}_V\ [\,]\ \rho\ \text{sd} & = & \texttt{nil} \\
\text{code}_V\ (e_1 : e_2)\ \rho\ \text{sd} & = & \text{code}_C\ e_1\ \rho\ \text{sd} \\
 & & \text{code}_C\ e_2\ \rho\ (\text{sd}+1) \\
 & & \texttt{cons}
\end{array}
$$

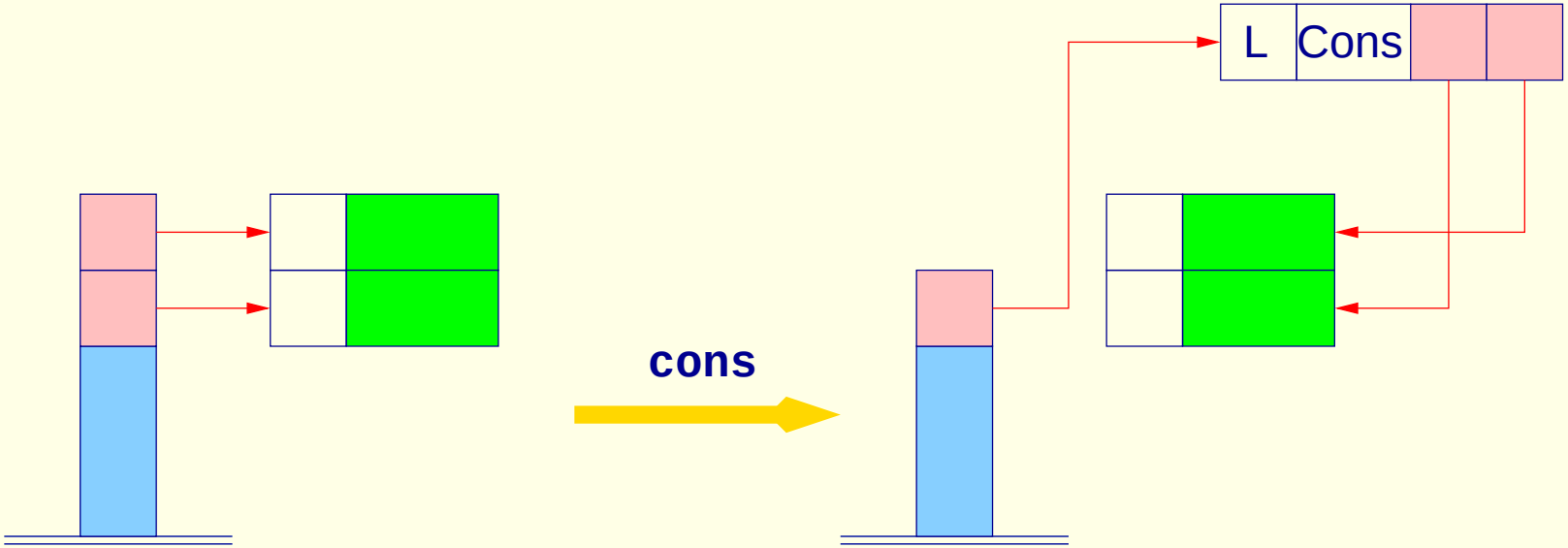
CBV korral väärtustatakse listi pea ja saba otse code_V abil.

Struktuursed andmetüübid



SP++;
S[SP] = new (L,Nil);

Struktuursed andmetüübid



S[SP-1] = new (L,Cons,S[SP-1],S[SP]);
SP–;

Struktuursed andmetüübid

- Listide inspekteerimine toimub *näidiste sobitamise* abil.
- Case-avaldise $e \equiv$ **case** e_0 **of** $[] \rightarrow e_1; h : t \rightarrow e_2$ väärtustamiseks:
  - väärtustatakse avaldis e_0;
  - kui e_0 väärtus on tühi list, siis väärtustatakse avaldis e_1;
  - kui e_0 on mitte-tühi list, siis lisatakse listi peale ja sabale vastavate komponentide viidad magasini (so. seotakse muutujad h ja t) ning väärtustatakse avaldis e_2.

Struktuursed andmetüübid

$$\begin{aligned}
\text{code}_V\ (\textbf{case}\ e_0\ \textbf{of}\ [\,] \rightarrow e_1;\ h : t \rightarrow e_2)\ \rho\ \text{sd} \quad = \quad & \text{code}_V\ e_0\ \rho\ \text{sd} \\
& \texttt{tlist A} \\
& \text{code}_V\ e_1\ \rho\ \text{sd} \\
& \texttt{jump B} \\
\text{A:}\ & \text{code}_V\ e_2\ \rho'\ (\text{sd}+2) \\
& \texttt{slide 2} \\
\text{B:}\ & \ldots
\end{aligned}$$

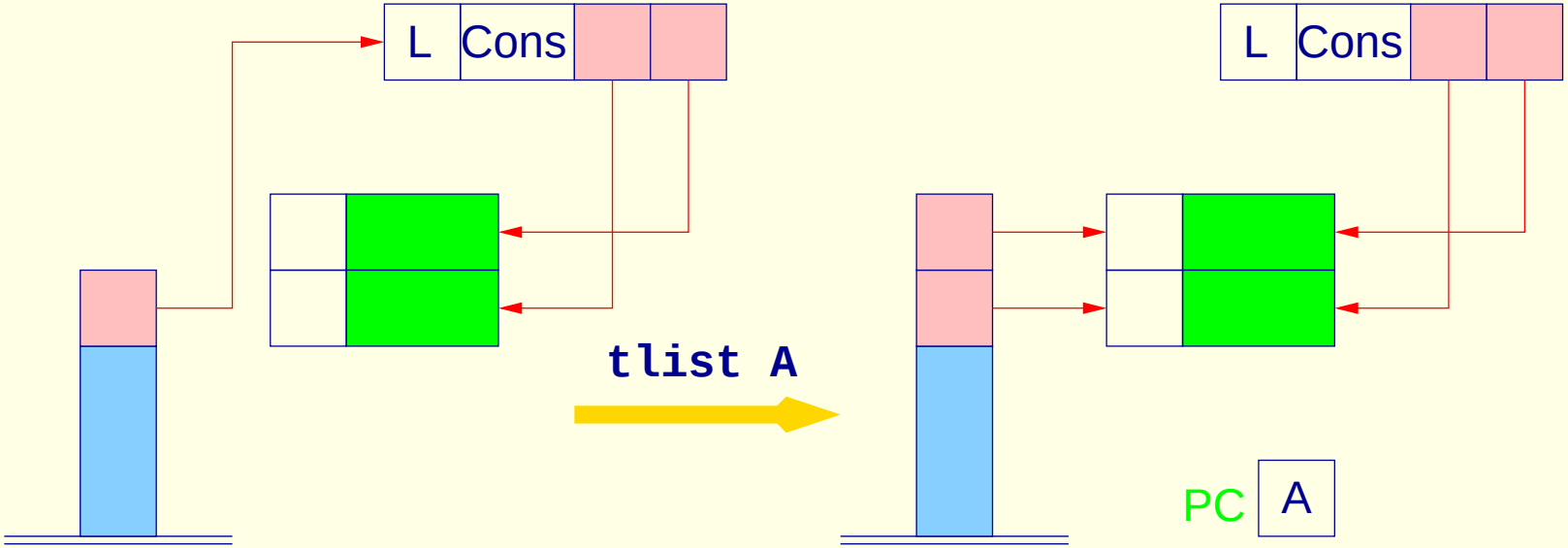
kus $\rho' = \rho \oplus \{h \mapsto (L, \text{sd}+1),\ t \mapsto (L, \text{sd}+2)\}$.

NB! On sama nii CBN kui CBV korral.

Struktuursed andmetüübid


```
if (S[SP]→tag ≠ L)
   Error ("Not List");
if (S[SP]→con = Nil)
   SP–;
```

Struktuursed andmetüübid




```
else {
   S[SP+1] = S[SP]→s[1];
   S[SP] = S[SP]→s[0];
   SP++; PC = A;
}
```

Struktuursed andmetüübid

Näide:

$$app = \textbf{fn}\ l, y \Rightarrow \textbf{case}\ l\ \textbf{of}$$
$$[\,] \rightarrow y$$
$$h : t \rightarrow h : (app\ t\ y)$$

```
0  targ 2
0  pushloc 0
1  eval
1  tlist A
0  pushloc 1
1  eval
1  jump B
```

```
2  A: pushloc 1
3     pushglob 0
4     pushloc 2
5     pushloc 6
6     mkvec 3
4     mkclos C
4     cons
0     slide 2
```

```
3  B: return 2
0  C: mark D
3     pushglob 2
4     pushglob 1
5     pushglob 0
6     eval
6     apply
1  D: update
```

Struktuursed andmetüübid

Kui ennik või list on sulundkontekstis, võib sulundi konstrueerimise asemel luua vastvad objektid otse:

$$
\begin{array}{lcl}
\text{code}_C\ (e_0,\ldots,e_{k-1})\ \rho\ \text{sd} & = & \text{code}_C\ e_0\ \rho\ \text{sd} \\
 & & \text{code}_C\ e_1\ \rho\ (\text{sd}+1) \\
 & & \ldots \\
 & & \text{code}_C\ e_{k-1}\ \rho\ (\text{sd}+k-1) \\
 & & \texttt{mkvec k} \\
\text{code}_C\ [\,]\ \rho\ \text{sd} & = & \texttt{nil} \\
\text{code}_C\ (e_1 : e_2)\ \rho\ \text{sd} & = & \text{code}_C\ e_1\ \rho\ \text{sd} \\
 & & \text{code}_C\ e_2\ \rho\ (\text{sd}+1) \\
 & & \texttt{cons}
\end{array}
$$

Sabarekursioon

- Funktsiooni aplikatsioon on avaldises *e sabapositsioonis*, kui tema väärtus võib olla kogu avaldise väärtuseks.
  - aplikatsioon $r\ t\ (h : y)$ on sabapositsioonis avaldises:

$$\textbf{case}\ x\ \textbf{of}\ [\,]\ \rightarrow\ y;\ h : t\ \rightarrow\ r\ t\ (h : y)$$

  - aplikatsioon $f(x-1)$ ei ole sabapositsioonis avaldises:

$$\textbf{if}\ x \leq 1\ \textbf{then}\ 1\ \textbf{else}\ x * f(x-1)$$

- Funktsioon on *sabarekursiivne*, kui kõik tema (nii otse, kui kaudselt) rekursiivsed väljakutsed on sabapositsioonis.
- Sabapositsioonis oleva funktsiooni aplikatsiooni jaoks ei ole vaja uut freimi luua!

Sabarekursioon

Sabapositsioonis oleva funktsiooni aplikatsiooni $e'\ e_0\ \ldots\ e_{m-1}$ korral genereeritakse kood, mis:

- seob argumendid e_i formaalsete parameetritega ning väärtustab avaldise e' F-objektiks;
- vabastab aktiivse freimi lokaalsed muutujad;
- rakendab funktsiooni argumentidele.

NB! Argumentide ja funktsiooni väärtustamine toimub kehtivas aktiivses freimis.

Sabarekursioon

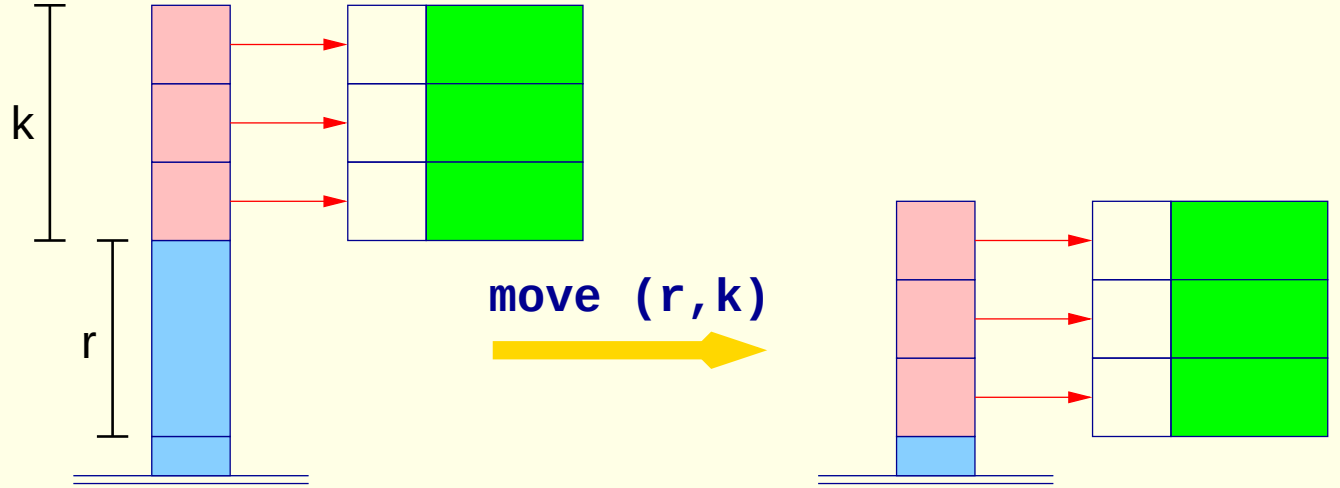
CBN korral genereeritakse kood:

$$
\begin{aligned}
\text{code}_V\ (e'\ e_0\ \ldots\ e_{m-1})\ \rho\ \text{sd} \quad = \quad & \text{code}_C\ e_{m-1}\ \rho\ \text{sd} \\
& \text{code}_C\ e_{m-2}\ \rho\ (\text{sd}+1) \\
& \ldots \\
& \text{code}_C\ e_0\ \rho\ (\text{sd}+m-1) \\
& \text{code}_V\ e'\ \rho\ (\text{sd}+m) \\
& \texttt{move}\ (\text{sd}+k, m+1) \\
& \texttt{apply}
\end{aligned}
$$

kus k on ”välise” funktsiooni parameetrite arv.

CBV korral on argumentide e_i jaoks code_C asemel code_V.

Sabarekursioon




```
SP = SP - k - r;
for (i=1; i≤k; i++)
   S[SP+i] = S[SP+i+r];
SP = SP + k;
```

Sabarekursioon

Näide: olgu $r = \textbf{case}\ x\ \textbf{of}\ [] \rightarrow y;\ h:t \rightarrow r\ t\ (h:y)$, siis funktsiooni r kehale vastab CBN korral kood:

```
0  targ 2          0     jump B          4     pushglob 0
0  pushloc 0                             5     eval
1  eval            2  A: pushloc 1       5     move (4,3)
1  tlist A         3     pushloc 4             apply
0  pushloc 1       4     cons
1  eval            3     pushloc 1       1  B: return 2
```

Kuna vanad organisatoorsed pesad on alles, siis on `return 2` saavutatav ainult otsehüppega tühjale listile vastavast harust.