Jätkud

Rekursiivne vs. iteratiivne käitumine

Rekursiivne faktoriaal

$$factR\,0 = 1$$
$$factR\,n = n * factR\,(n-1)$$

Rekursiivne vs. iteratiivne käitumine

Rekursiivne faktoriaal

$$factR\,0 = 1$$
$$factR\,n = n * factR\,(n-1)$$

NB!

factR iga järgmine rekursiivne väljakutse toimub üha suuremas kontekstis:

$$\begin{aligned}
factR\,4 &\Longrightarrow 4 * factR\,3 \\
&\Longrightarrow 4 * (3 * factR\,2) \\
&\Longrightarrow 4 * (3 * (2 * factR\,1)) \\
&\Longrightarrow 4 * (3 * (2 * (1 * factR\,0))) \\
&\Longrightarrow 4 * (3 * (2 * (1 * 1))) \\
&\Longrightarrow 24
\end{aligned}$$

Rekursiivne vs. iteratiivne käitumine

Sabarekursiivne (akumulaatoriga) faktoriaal

$$
\begin{aligned}
&\mathit{factA} = \mathit{factAcc}\ 1 \\
&\quad \textbf{where}\ \mathit{factAcc}\ a\ 0 = a \\
&\qquad \mathit{factAcc}\ a\ n = \mathit{factAcc}\ (a * n)\ (n-1)
\end{aligned}
$$

NB!

Kõik väljakutsed toimuvad samas kontekstis:

$$
\begin{aligned}
\mathit{factA}\ 4 &\Longrightarrow \mathit{factAcc}\ 1\ 4 \\
&\Longrightarrow \mathit{factAcc}\ 4\ 3 \\
&\Longrightarrow \mathit{factAcc}\ 12\ 2 \\
&\Longrightarrow \mathit{factAcc}\ 24\ 1 \\
&\Longrightarrow \mathit{factAcc}\ 24\ 0 \\
&\Longrightarrow 24
\end{aligned}
$$

Rekursiivne vs. iteratiivne käitumine

Iteratiivne kontroll-käitumine

- Funktsioon on iteratiivse käitumisega kui ta kasutab O(1) mälu
- Funktsioon on iteratiivse kontroll-käitumisega kui ta kasutab O(1) mälu konteksti säilitamiseks
- Funktsioon on rekursiivse (kontroll-)käitumisega, kui ta ei ole iteratiivse (kontroll-)käitumisega

Näide

$$
\begin{aligned}
&\mathit{reverse} = \mathit{revApp}\ [\,] \\
&\qquad \textbf{where}\ \mathit{revApp}\ \mathit{as}\ [\,] \quad\ = \mathit{as} \\
&\qquad\qquad \mathit{revApp}\ \mathit{as}\ (x : \mathit{xs}) = \mathit{revApp}\ (x : \mathit{as})\ \mathit{xs}
\end{aligned}
$$

Rekursiivne vs. iteratiivne käitumine

Küsimus

Millal on funktsioon iteratiivse kontroll-käitumisega?

NB!

Üldjuhul lahendamatu ülesanne!

Näide

$$
\begin{aligned}
\mathit{factQ}\ n = \ & \textbf{if}\ \mathit{strangePredicate}\ n \\
& \textbf{then}\ \mathit{factR}\ n \\
& \textbf{else}\ \ \mathit{factA}\ n
\end{aligned}
$$

Rekursiivne vs. iteratiivne käitumine

Sabapositsioonid/-kutsed/-vormid

- Alamavaldis on sabapositsioonis (ingl. *tail position*), kui tema väärtustamise korral on ta väärtus "koheselt" kogu avaldise väärtuseks

  if $expr_0$ **then** $expr_1$ **else** $expr_2$
  let $x_1 = expr_1$ **in** $expr_0$

- Sabapositsioonis olevat funktsiooniaplikatsiooni nimetatakse sabakutseks (ingl. *tail call*)
- Avaldis on sabavormis (ingl. *tail form*), kui kõik tema mittesabapositsioonis olevad alamavaldised on "lihtsad"

NB!

Sabavormis avaldis on iteratiivse kontroll-käitumisega!

Jätkud

Arvutuskontekstid ja jätkud

- Vaatame $\mathit{factR}\,4$ arvutussammu $4 * (3 * (2 * \mathit{factR}\,1))$
- Väljakutse $\mathit{factR}\,1$ arvutatakse kontekstis $4 * (3 * (2 * \Box))$
- Me saame konteksti ("ühe auguga" avaldist) esitada ühe argumendilise lambda-avaldisena, mida kutsutakse jätkuks (ingl. *continuation*) $\backslash v \rightarrow 4 * (3 * (2 * v))$
- Tähistagu k seda lambda-avaldist, siis kehtib

$$k\,(\mathit{factR}\,1) \Longrightarrow 4 * (3 * (2 * \mathit{factR}\,1))$$

- Analoogselt on kõik factR arvutamise sammud esitatavad kujul $(k\,(\mathit{factR}\,n))$, kus k on vastavat konteksti esitav jätk
- CPS = *Continuation Passing Style*

Jätkud

CPS-kujul faktoriaal

$$factCPS\ 0\ k = k\ 1$$
$$factCPS\ n\ k = factCPS\ (n-1)\ (\backslash v \rightarrow k\ (n * v))$$

Jätkud

CPS-kujul faktoriaal

$$factCPS\ 0\ k = k\ 1$$
$$factCPS\ n\ k = factCPS\ (n-1)\ (\backslash v \rightarrow k\ (n * v))$$

Väide

Iga k ja n korral

$$k\ (factR\ n) \quad = \quad factCPS\ n\ k$$

Jätkud

CPS-kujul faktoriaal

$$factCPS\ 0\ k = k\ 1$$
$$factCPS\ n\ k = factCPS\ (n-1)\ (\backslash v \rightarrow k\ (n * v))$$

Väide

Iga k ja n korral

$$k\ (factR\ n) \quad = \quad factCPS\ n\ k$$

Tõestus ($n = 0$)

$$k\ (factR\ 0) \quad = \quad k\ 1 \quad = \quad factCPS\ 0\ k$$

Jätkud

CPS-kujul faktoriaal

$$factCPS\ 0\ k = k\ 1$$
$$factCPS\ n\ k = factCPS\ (n-1)\ (\backslash v \rightarrow k\ (n * v))$$

Väide

Iga k ja n korral

$$k\ (factR\ n) \quad = \quad factCPS\ n\ k$$

Tõestus ($n > 0$)

$$\begin{aligned} k\ (factR\ n) &= k\ (n * factR\ (n-1)) \\ &= (\backslash v \rightarrow k\ (n * v))\ (factR\ (n-1)) \\ &= factCPS\ (n-1)\ (\backslash v \rightarrow k\ (n * v)) \\ &= factCPS\ n\ k \end{aligned}$$

Jätkud

NB!

factCPS on iteratiivse kontroll-käitumisega

$$
\begin{aligned}
& \mathit{factCPS}\ 4\ k \\
\Longrightarrow\ & \mathit{factCPS}\ 3\ (\backslash v \rightarrow k\ (4 * v)) \\
\Longrightarrow\ & \mathit{factCPS}\ 2\ (\backslash v \rightarrow k\ (4 * (3 * v))) \\
\Longrightarrow\ & \mathit{factCPS}\ 1\ (\backslash v \rightarrow k\ (4 * (3 * (2 * v)))) \\
\Longrightarrow\ & \mathit{factCPS}\ 0\ (\backslash v \rightarrow k\ (4 * (3 * (2 * (1 * v))))) \\
\Longrightarrow\ & k\ (4 * (3 * (2 * (1 * 1)))) \\
\Longrightarrow\ & k\ 24
\end{aligned}
$$

NB!

“Kontekstist sõltumatu” faktoriaali saame, kui anname algjätkuks identsusfunktsiooni

$$\mathit{factC}\ n = \mathit{factCPS}\ n\ \mathit{id}$$

Jätkud

Rekursiivne *length*

$$
\begin{aligned}
&\mathit{length}\,[\,] &&= 0\\
&\mathit{length}\,(x : xs) &&= 1 + \mathit{length}\,xs
\end{aligned}
$$

CPS-kujul *length*

$$
\begin{aligned}
&\mathit{lengthC}\,xs = \mathit{lengthCPS}\,xs\,\mathit{id}\\
&\quad \textbf{where}\ \mathit{lengthCPS}\,[\,] \quad k = k\,0\\
&\qquad\qquad \mathit{lengthCPS}\,(x : xs)\ k = \mathit{lengthCPS}\,xs\,(\backslash v \rightarrow\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad k\,(1 + v))
\end{aligned}
$$

Jätkud

CPS-teisendus

Üldine meetod avaldise teisendamiseks CPS kujule:

- Identifitseerida mittetriviaalsed vahetulemused
- Järjestikustada nende arvutamine
- Tuua sisse jätkud konteksti esitamiseks

Jätkud

CPS-teisendus

Üldine meetod avaldise teisendamiseks CPS kujule:

- Identifitseerida mittetriviaalsed vahetulemused
- Järjestikustada nende arvutamine
- Tuua sisse jätkud konteksti esitamiseks

Näide: S-kombinaator

$$f\ x\ (g\ x)$$

$$\begin{array}{l}\textbf{let } v_1 = f\ x \\ \quad\ v_2 = g\ x \\ \quad\ v_3 = v_1\ v_2 \\ \textbf{in } v_3\end{array}$$

$$\begin{array}{l}\backslash k \to f\ x \quad (\backslash v_1 \to \\ \qquad\quad g\ x \quad (\backslash v_2 \to \\ \qquad\quad v_1\ v_2\ (\backslash v_3 \to \\ \qquad\quad k\ v_3)))\end{array}$$

Jätkud

CPS-teisendus

Üldine meetod avaldise teisendamiseks CPS kujule:

- Identifitseerida mittetriviaalsed vahetulemused
- Järjestikustada nende arvutamine
- Tuua sisse jätkud konteksti esitamiseks

Näide: S-kombinaator

$$f\ x\ (g\ x)$$

$$\begin{aligned}&\textbf{let } v_1 = f\ x\\ &\quad\ \ v_2 = g\ x\\ &\quad\ \ v_3 = v_1\ v_2\\ &\textbf{in } v_3\end{aligned}$$

$$\begin{aligned}&\backslash k \to f\ x \quad (\backslash v_1 \to\\ &\qquad g\ x \quad (\backslash v_2 \to\\ &\qquad v_1\ v_2\ (\backslash v_3 \to\\ &\qquad k\ v_3)))\end{aligned}$$

Jätkud

CPS-teisendus

Üldine meetod avaldise teisendamiseks CPS kujule:

- Identifitseerida mittetriviaalsed vahetulemused
- Järjestikustada nende arvutamine
- Tuua sisse jätkud konteksti esitamiseks

Näide: S-kombinaator

$$f\ x\ (g\ x)$$

$$\begin{aligned}&\textbf{let } v_1 = f\ x\\ &\quad\ \ v_2 = g\ x\\ &\quad\ \ v_3 = v_1\ v_2\\ &\textbf{in } v_3\end{aligned}$$

$$\begin{aligned}\backslash k \to f\ x \quad &(\backslash v_1 \to\\ g\ x \quad &(\backslash v_2 \to\\ v_1\ v_2 \ &(\backslash v_3 \to\\ k\ v_3)))\end{aligned}$$

Jätkud

CPS-teisendus

Üldine meetod avaldise teisendamiseks CPS kujule:

- Identifitseerida mittetriviaalsed vahetulemused
- Järjestikustada nende arvutamine
- Tuua sisse jätkud konteksti esitamiseks

Näide: Fibonacci

$$
\begin{aligned}
&\mathit{fibo}\,0 = 0\\
&\mathit{fibo}\,1 = 1\\
&\mathit{fibo}\,n = \mathit{fibo}\,(n-1) + \mathit{fibo}\,(n-2)
\end{aligned}
$$

Jätkud

CPS-teisendus

Üldine meetod avaldise teisendamiseks CPS kujule:

- Identifitseerida mittetriviaalsed vahetulemused
- Järjestikustada nende arvutamine
- Tuua sisse jätkud konteksti esitamiseks

Näide: Fibonacci

$$
\begin{aligned}
&\mathit{fibo}\,0 = 0\\
&\mathit{fibo}\,1 = 1\\
&\mathit{fibo}\,n = \textbf{let}\ v_1 = \mathit{fibo}\,(n-1)\\
&\qquad\qquad\ \ v_2 = \mathit{fibo}\,(n-2)\\
&\qquad\quad \textbf{in}\ v_1 + v_2
\end{aligned}
$$

Jätkud

CPS-teisendus

Üldine meetod avaldise teisendamiseks CPS kujule:

- Identifitseerida mittetriviaalsed vahetulemused
- Järjestikustada nende arvutamine
- Tuua sisse jätkud konteksti esitamiseks

Näide: Fibonacci

$$
\begin{aligned}
&\mathit{fibo}\ 0\ k = k\ 0\\
&\mathit{fibo}\ 1\ k = k\ 1\\
&\mathit{fibo}\ n\ k = \mathit{fibo}\ (n-1)\ (\backslash v_1 \rightarrow\\
&\qquad\qquad \mathit{fibo}\ (n-2)\ (\backslash v_2 \rightarrow\\
&\qquad\qquad k\ (v_1 + v_2)))
\end{aligned}
$$

Jätkud

CPS-teisendus

Üldine meetod avaldise teisendamiseks CPS kujule:

- Identifitseerida mittetriviaalsed vahetulemused
- Järjestikustada nende arvutamine
- Tuua sisse jätkud konteksti esitamiseks

Näide: *map*

$$\begin{aligned} map\ f\ [\,] &= [\,] \\ map\ f\ (x : xs) &= f\ x : map\ f\ xs \end{aligned}$$

Jätkud

CPS-teisendus

Üldine meetod avaldise teisendamiseks CPS kujule:

- Identifitseerida mittetriviaalsed vahetulemused
- Järjestikustada nende arvutamine
- Tuua sisse jätkud konteksti esitamiseks

Näide: *map*

$$
\begin{aligned}
map\, f\, [\,] &= [\,] \\
map\, f\, (x : xs) &= \textbf{let } v_1 = f\, x \\
&\qquad\quad\; v_2 = map\, f\, xs \\
&\quad\; \textbf{in } v_1 : v_2
\end{aligned}
$$

Jätkud

CPS-teisendus

Üldine meetod avaldise teisendamiseks CPS kujule:

- Identifitseerida mittetriviaalsed vahetulemused
- Järjestikustada nende arvutamine
- Tuua sisse jätkud konteksti esitamiseks

Näide: *map*

$$
\begin{aligned}
&map\ f\ [\,] && k = k\ [\,] \\
&map\ f\ (x : xs) && k = f\ x\ (\backslash v_1 \rightarrow \\
& && \quad map\ f\ xs\ (\backslash v_2 \rightarrow \\
& && \quad k\ (v_1 : v_2)))
\end{aligned}
$$

Jätkud

CPS vahekokkuvõte

CPS kujul programmis on kõik mittetriviaalsed tegevused ning nende tegemise järjekord ilmutatud

- Lihtsustab semantikat ning transleerimist
- Kõrgemat-järku funktsioonide asemel saab jätkude esitamiseks kasutada andmestruktuure (defunktsionaliseerimine)
- Jätke võib ka kasutada ”mitte-standartselt”, võimaldades selliselt modelleerida erinevaid juhtimiskonstruktsioone

Näiteid jätkude rakendustest: erindite töötlus

Leida listielementide korrutis (ver. 1)

$$
\begin{aligned}
prod\,[\,] &= 1 \\
prod\,(x : xs) &= x * prod\,xs
\end{aligned}
$$

Näiteid jätkude rakendustest: erindite töötlus

Leida listielementide korrutis (ver. 1)

$$\begin{aligned} prod\,[\,] &= 1 \\ prod\,(x : xs) &= x * prod\,xs \end{aligned}$$

NB!

Kui argumentlist sisaldab nulli, siis on tulemus alati null!

$$prod\,[1,2,3,0,1,2,3,4,5,6,7] \Longrightarrow 0$$

NB!

Toodud definitsiooni korral läbitakse kogu list ja korrutatakse kõik elemendid!

Näiteid jätkude rakendustest: erindite töötlus

Leida listielementide korrutis (ver. 2)

$$
\begin{aligned}
prod\,[\,] &= 1 \\
prod\,(0 : xs) &= 0 \\
prod\,(x : xs) &= x * prod\,xs
\end{aligned}
$$

Näiteid jätkude rakendustest: erindite töötlus

Leida listielementide korrutis (ver. 2)

$$
\begin{aligned}
prod\,[\,] &= 1 \\
prod\,(0 : xs) &= 0 \\
prod\,(x : xs) &= x * prod\,xs
\end{aligned}
$$

NB!

Läbitakse listi kuni esimese nullini!

NB!

Tehakse nii mitu korrutamist, kui mitu elementi läbiti!

Näiteid jätkude rakendustest: erindite töötlus

Jätkudega versioon

$$
\begin{aligned}
&\mathit{prod}\ \mathit{xs} = \mathit{prodC}\ \mathit{xs}\ \mathit{id} \\
&\quad \textbf{where}\ \mathit{prodC}\ [\,]\ \ \ \ \ \ \ \ k = k\,1 \\
&\qquad\qquad \mathit{prodC}\ (0 : \mathit{xs})\ k = 0 \\
&\qquad\qquad \mathit{prodC}\ (x : \mathit{xs})\ k = \mathit{prodC}\ \mathit{xs}\ (\backslash v \rightarrow k\ (x * v))
\end{aligned}
$$

Näiteid jätkude rakendustest: erindite töötlus

Jätkudega versioon

$$
\begin{aligned}
&prod\ xs = prodC\ xs\ id \\
&\quad \textbf{where}\ prodC\ [\,]\ \ \ \ \ \ k = k\ 1 \\
&\qquad\qquad prodC\ (0 : xs)\ k = 0 \\
&\qquad\qquad prodC\ (x : xs)\ k = prodC\ xs\ (\backslash v \rightarrow k\ (x * v))
\end{aligned}
$$

NB!

List läbitakse ainult kui (esimese) nullini!

NB!

Nulli korral ei tehta ühtegi korrutamist!

Näiteid jätkude rakendustest: mitu resultaati

Leida listi pikkus ja elementide summa

$$
\begin{aligned}
&\mathit{sumLength} = \mathit{sumLen}\ 0\ 0 \\
&\quad \textbf{where}\ \mathit{sumLen}\ s\ l\ [\,] \quad\quad = (s, l) \\
&\qquad\qquad \mathit{sumLen}\ s\ l\ (x : xs) = \mathit{sumLen}\ (s + x)\ (l + 1)\ xs
\end{aligned}
$$

Näiteid jätkude rakendustest: mitu resultaati

Leida listi pikkus ja elementide summa

$$
\begin{aligned}
&\mathit{sumLength} = \mathit{sumLen}\ 0\ 0 \\
&\quad \textbf{where}\ \mathit{sumLen}\ s\ l\ [\,] \quad\quad = (s, l) \\
&\qquad\qquad \mathit{sumLen}\ s\ l\ (x : xs) = \mathit{sumLen}\ (s + x)\ (l + 1)\ xs
\end{aligned}
$$

NB!

Ehitab paari, mille järgmine funktsioon, mis tahab tulemust kasutada, peab kohe “tükeldama”!

$$
\begin{aligned}
&\mathit{average}\ xs = \textbf{let}\ (s, l) = \mathit{sumLength}\ xs \\
&\qquad\qquad\qquad\quad \textbf{in}\ s\,\text{`}\mathit{div}\text{`}\ l
\end{aligned}
$$

Näiteid jätkude rakendustest: mitu resultaati

Jätkudega versioon

$$
\begin{aligned}
&\mathit{sumLengthC}\ \mathit{xs}\ k = \mathit{sumLen}\ 0\ 0\ \mathit{xs} \\
&\quad \textbf{where}\ \mathit{sumLen}\ s\ l\ [\,] \qquad = k\ s\ l \\
&\qquad\qquad \mathit{sumLen}\ s\ l\ (x : \mathit{xs}) = \mathit{sumLen}\ (s + x)\ (l + 1)\ \mathit{xs} \\
&\mathit{averageC}\ \mathit{xs} = \mathit{sumLengthC}\ \mathit{xs}\ (\backslash s\ l \rightarrow s\ `\mathit{div}`\ l)
\end{aligned}
$$

Näiteid jätkude rakendustest: mitu resultaati

Jätkudega versioon

$$
\begin{aligned}
&\mathit{sumLengthC}\ \mathit{xs}\ k = \mathit{sumLen}\ 0\ 0\ \mathit{xs} \\
&\quad \textbf{where}\ \mathit{sumLen}\ s\ l\ [\,] \qquad = k\ s\ l \\
&\qquad\qquad \mathit{sumLen}\ s\ l\ (x : \mathit{xs}) = \mathit{sumLen}\ (s + x)\ (l + 1)\ \mathit{xs} \\
&\mathit{averageC}\ \mathit{xs} = \mathit{sumLengthC}\ \mathit{xs}\ (\backslash s\ l \rightarrow s\ `\mathit{div}`\ l)
\end{aligned}
$$

NB!

Resultaadid antakse otse edasi!

Jätkude esitamine andmestruktuuridena

CPS-kujul faktoriaal

$$
\begin{aligned}
&\mathit{factC}\ n = \mathit{factCPS}\ n\ \mathit{id}\\
&\quad \textbf{where}\ \mathit{factCPS}\ 0\ k = k\ 1\\
&\qquad\quad \mathit{factCPS}\ n\ k = \mathit{factCPS}\ (n-1)\\
&\qquad\qquad\qquad\qquad\qquad\qquad (\backslash v \rightarrow k\ (n * v))
\end{aligned}
$$

Jätkude esitamine andmestruktuuridena

Jätkuesitusest sõltumatu CPS-kujul faktoriaal

$$
\begin{aligned}
&\mathit{factC}\ n = \mathit{factCPS}\ n\ \mathit{mkFinalCont}\\
&\quad \textbf{where}\ \mathit{factCPS}\ 0\ \ k = \mathit{applyCont}\ k\ 1\\
&\qquad\qquad \mathit{factCPS}\ n\ k = \mathit{factCPS}\ (n-1)\\
&\qquad\qquad\qquad\qquad\qquad\qquad (\mathit{mkNewCont}\ k\ n)
\end{aligned}
$$

Jätkude esitamine andmestruktuuridena

Jätkuesitusest sõltumatu CPS-kujul faktoriaal

$$
\begin{aligned}
&\mathit{factC}\ n = \mathit{factCPS}\ n\ \mathit{mkFinalCont}\\
&\quad \textbf{where}\ \mathit{factCPS}\ 0\ k = \mathit{applyCont}\ k\ 1\\
&\qquad\quad \mathit{factCPS}\ n\ k = \mathit{factCPS}\ (n-1)\\
&\qquad\qquad\qquad\qquad\qquad\qquad (\mathit{mkNewCont}\ k\ n)
\end{aligned}
$$

Jätkude esitamine funktsioonidena

$$
\begin{aligned}
\mathit{mkFinalCont} &= \mathit{id}\\
\mathit{mkNewCont}\ k\ n &= \backslash v \rightarrow \mathit{applyCont}\ k\ (n * v)\\
\mathit{applyCont}\ k\ v &= k\ v
\end{aligned}
$$

Jätkude esitamine andmestruktuuridena

Jätkude esitamine andmestruktuuridena

```
data FactCont    = FinalFactCont
                 | NewFactCont Int FactCont

mkFinalCont      = FinalFactCont
mkNewCont k n    = NewFactCont n k
applyCont k v    = case k of
  FinalFactCont     → v
  NewFactCont n k → applyCont k (n * v)
```

Jätkude esitamine andmestruktuuridena

Jätkude esitamine andmestruktuuridena

$$
\begin{aligned}
&\textbf{data}\ \mathit{FactCont} \quad = \mathit{FinalFactCont} \\
&\qquad\qquad\quad\ \mid \mathit{NewFactCont}\ \mathit{Int}\ \mathit{FactCont} \\
&\mathit{mkFinalCont} \quad = \mathit{FinalFactCont} \\
&\mathit{mkNewCont}\ k\ n = \mathit{NewFactCont}\ n\ k \\
&\mathit{applyCont}\ k\ v \quad = \textbf{case}\ k\ \textbf{of} \\
&\quad \mathit{FinalFactCont} \quad \rightarrow v \\
&\quad \mathit{NewFactCont}\ n\ k \rightarrow \mathit{applyCont}\ k\ (n * v)
\end{aligned}
$$

NB!

Tüüp *FactCont* on isomorfne täisarvu listidega!

Jätkude esitamine andmestruktuuridena

Jätkude esitamine andmestruktuuridena

$$
\begin{aligned}
&\mathit{mkFinalCont} &&= [\,] \\
&\mathit{mkNewCont}\ k\ n &&= n : k \\
&\mathit{applyCont}\ k\ v &&= \mathit{foldl}\ (\backslash v\ n \rightarrow n * v)\ v\ k
\end{aligned}
$$

Jätkude esitamine andmestruktuuridena

Jätkude esitamine andmestruktuuridena

$$
\begin{aligned}
&\mathit{mkFinalCont} &&= [\,] \\
&\mathit{mkNewCont}\ k\ n &&= n : k \\
&\mathit{applyCont}\ k\ v &&= \mathit{foldl}\ (\backslash v\ n \rightarrow n * v)\ v\ k
\end{aligned}
$$

NB!

Asendame need definitsioonid tagasi *factC* definitsiooni ...

CPS-kujul faktoriaal — jätkud listidena

$$
\begin{aligned}
&\mathit{factC}\ n = \mathit{factCPS}\ n\ [\,] \\
&\quad \textbf{where}\ \mathit{factCPS}\ 0\ k = \mathit{foldl}\ (\backslash v\ n \rightarrow n * v)\ 1\ k \\
&\qquad\qquad \mathit{factCPS}\ n\ k = \mathit{factCPS}\ (n-1)\ (n : k)
\end{aligned}
$$