Engineering CPS transformations

Matthias Puech

Complogic seminar, McGill University
August 8, 2014

Purpose of this talk

- Introduce you to CPS transformations
- Give you my understanding of classic results
  (reinvent them from a different angle)
- Gather comments about an upcoming draft

Purpose of this talk

- Introduce you to CPS transformations
- Give you my understanding of classic results
  (reinvent them from a different angle)
- Gather comments about an upcoming draft

The medium

Build incrementally an optimized CPS
"one-pass, β-normal, properly tail-recursive"

The message

Tools to engineer transformations based on:

- tight (typed) syntax
- optimization analysis

Continuation-passing styles

A CPS transformation is

- a semantic artifact
  ($\simeq$ operational/denotational/process/. . . semantics)
- an intermediate language in compilers
  (complex language $\rightarrow$ simpler language)
- a proof transformation
  (classical $\rightarrow$ intuitionistic)
- a programming technique
- . . .

Continuation-passing styles

A CPS transformation is

- a semantic artifact
  ($\simeq$ operational/denotational/process/…semantics)
- an intermediate language in compilers
  (complex language $\rightarrow$ simpler language)
- a proof transformation
  (classical $\rightarrow$ intuitionistic)
- a programming technique
- …

Many variants, long, long history

- here: *call-by-value*
  (exercise: *call-by-name*)

Example: CPS for compiler construction

Theoretical Computer Science 1 (1975) 125–159. © North-Holland Publishing Company

CALL-BY-NAME, CALL-BY-VALUE AND THE λ-CALCULUS

G. D. PLOTKIN

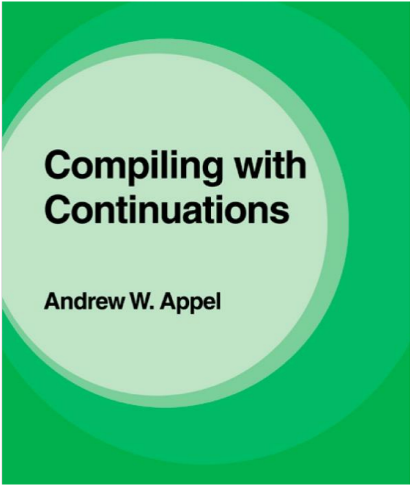
Department of Machine Intelligence, School of Artificial Intelligence, University of Edinburgh, Edinburgh, United Kingdom

Communicated by R. Milner
Received 1 August 1974

Abstract. This paper examines the old question of the relationship between ISWIM and the λ-calculus, using the distinction between call-by-value and call-by-name. It is held that the relationship should be mediated by a standardisation theorem. Since this leads to difficulties, a new λ-calculus is introduced whose standardisation theorem gives a good correspondence with ISWIM as given by the SECD machine, but without the *letrec* feature. Next a call-by-name variant of ISWIM is introduced which is in an analogous correspondence with the usual λ-calculus. The relation between call-by-value and call-by-name is then studied by giving simulations of each language by the other and interpretations of each calculus in the other. These are obtained as another application of the continuation technique. Some emphasis is placed throughout on the notion of operational equality (or contextual equality). If terms can be proved equal in a calculus they are operationally equal in the corresponding language. Unfortunately, operational equality is not preserved by either of the simulations.

1974

Example: CPS for compiler construction



1992

Example: CPS for compiler construction

The Essence of Compiling with Continuations

Cormac Flanagan* Amr Sabry* Bruce F. Duba Matthias Felleisen*

Department of Computer Science
Rice University
Houston, TX 77251-1892

Abstract

In order to simplify the compilation process, many compilers for higher-order languages use the continuation-passing style (CPS) transformation in a first phase to generate an intermediate representation of the source program. The salient aspect of this intermediate form is that all procedures take an argument that represents the rest of the computation (the "continuation"). Since the naïve CPS transformation considerably increases the size of programs, CPS compilers perform reductions to produce a more compact intermediate representation. Although often implemented as a part of the CPS transformation, this step is conceptually a second phase. Finally, code generators for typical CPS compilers treat continuations specially in order to optimize the interpretation of continuation parameters.

A thorough analysis of the abstract machine for CPS terms shows that the actions of the code generator *invert* the naïve CPS translation step. Put differently, the combined effect of the three phases is equivalent

the β-value rule is an operational semantics for the source language, that the conventional *full* λ-calculus is a semantics for the intermediate language, and, most importantly, that the λ-calculus proves more equations between CPS terms than the λ_v-calculus does between corresponding terms of the source language. Translated into practice, a compiler can perform more transformations on the intermediate language than on the source language [2:4–5]. Second, the language of CPS terms is basically a stylized assembly language, for which it is easy to generate actual assembly programs for different machines [2, 13, 20]. In short, the CPS transformation provides an organizational principle that simplifies the construction of compilers.

To gain a better understanding of the role that the CPS transformation plays in the compilation process, we recently studied the precise connection between the λ_v-calculus for source terms and the λ-calculus for CPS terms. The result of this research [17] was an extended λ_v-calculus that precisely corresponds to the λ-calculus of the intermediate CPS language and that is still

1993

Example: CPS for compiler construction

Compiling with Continuations, Continued

Andrew Kennedy
Microsoft Research Cambridge
akenn@microsoft.com

Abstract

We present a series of CPS-based intermediate languages suitable for functional language compilation, arguing that they have practical benefits over direct-style languages based on *A*-normal form (ANF) or monads. Inlining of functions demonstrates the benefits most clearly: in ANF-based languages, inlining involves a renormalization step that rearranges let expressions and possibly introduces a new 'join point' function, and in monadic languages, commuting conversions must be applied; in contrast, inlining in our CPS language is a simple substitution of variables for variables.

We present a contification transformation implemented by simple rewrites on the intermediate language. Exceptions are modelled using so-called 'double-barrelled' CPS. Subtyping on exception constructors then gives a very straightforward effect analysis for exceptions. We also show how a graph-based representation of CPS terms can be implemented extremely efficiently, with linear-time term simplification.

Categories and Subject Descriptors D.3.4 [*Programming Languages*]: Processors – Compilers

so monads were a natural choice for separating computations from values in both terms and types. But, given the history of CPS, probably there was also a feeling that "CPS is for call/cc", something that is not a feature of Standard ML.

Recently, the author has re-implemented all stages of the SML.NET compiler pipeline to use a CPS-based intermediate language. Such a change was not undertaken lightly, amounting to roughly 25,000 lines of replaced or new code. There are many benefits: the language is smaller and more uniform, simplification of terms is more straightforward and extremely efficient, and advanced optimizations such as contification are more easily expressed. We use CPS only because it is a *good place to do optimization*; we are not interested in first-class control in the source language (call/cc), or as a means of implementing other features such as concurrency. Indeed, as SML.NET targets .NET IL, a call-stack-based intermediate language with support for structured exception handling, the compilation process can be summarized as "transform direct style (SML) into CPS; optimize CPS; transform CPS back to direct style (.NET IL)".

2007

Outline

1. Fischer & Plotkin's original CPS transformation
2. *One-pass CPS* (through Control-Flow Analysis)
3. The syntax of CPS terms (through syntax aggregation)
4. Proper transformation of *β-redexes*
5. Proper transformation of *tail calls*

Fischer & Plotkin's original transformation

$$M ::= \lambda x.\, M \mid M\; M \mid x \mid \mathbf{let}\, x = M\, \mathbf{in}\, M \qquad \in \mathit{Exp}$$

Fischer & Plotkin's original transformation

$$M ::= \lambda x.\, M \mid M\; M \mid x \mid \mathbf{let}\, x = M\, \mathbf{in}\, M \qquad\qquad \in \mathit{Exp}$$

$$\begin{aligned}
[\![\cdot]\!] \;&:\; M \to M \\
[\![x]\!] \;&= \lambda k.\, k\; x \\
[\![\lambda x.\, M]\!] &= \lambda k.\, k\; (\lambda x.\, [\![M]\!]) \\
[\![M\; N]\!] &= \lambda k.\, [\![M]\!]\; (\lambda m.\, [\![N]\!]\; (\lambda n.\, m\; n\; k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\; (\lambda x.\, [\![N]\!]\; k)
\end{aligned}$$

Fischer & Plotkin's original transformation

$$M ::= \lambda x. M \mid M\ M \mid x \mid \mathbf{let}\, x = M\, \mathbf{in}\, M \qquad \in Exp$$

$$\llbracket \cdot \rrbracket \ : \ M \to M$$

$$\begin{aligned}
\llbracket x \rrbracket &= \lambda k. k\ x \\
\llbracket \lambda x. M \rrbracket &= \lambda k. k\ (\lambda x. \llbracket M \rrbracket) \\
\llbracket M\ N \rrbracket &= \lambda k. \llbracket M \rrbracket\ (\lambda m. \llbracket N \rrbracket\ (\lambda n. m\ n\ k)) \\
\llbracket \mathbf{let}\, x = M\, \mathbf{in}\, N \rrbracket &= \lambda k. \llbracket M \rrbracket\ (\lambda x. \llbracket N \rrbracket\ k)
\end{aligned}$$

Properties

Simulation $\llbracket eval_v(M) \rrbracket \simeq eval_v(\llbracket M \rrbracket\ (\lambda x. x))$

Indifference $eval_v(\llbracket M \rrbracket (\lambda x. x)) \simeq eval_n(\llbracket M \rrbracket (\lambda x. x))$

Problem *"administrative redexes"*

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x. M]\!] &= \lambda k.\, k\, (\lambda x.\, [\![M]\!]) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda m.\, [\![N]\!]\ (\lambda n. m\, n\, k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda x.\, [\![N]\!]\ k)
\end{aligned}
$$

Examples

- $[\![\lambda x. x]\!] = \lambda k.\, k\, (\lambda x k.\, k\, x)$

Problem *"administrative redexes"*

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x.M]\!] &= \lambda k.\, k\, (\lambda x.\, [\![M]\!]) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda m.\, [\![N]\!]\ (\lambda n.\, m\ n\ k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda x.\, [\![N]\!]\ k)
\end{aligned}
$$

Examples

- $[\![\lambda x.x]\!] = \lambda k.\, k\, (\lambda xk.\, k\, x)$
- $[\![(\lambda x.x)(\lambda x.x)]\!] =$
  $\lambda k.\, (\lambda k.\, k\, (\lambda xk.\, k\, x))\ (\lambda m.\, (\lambda k.\, k\, (\lambda xk.\, k\, x))\ (\lambda n.\, m\ n\ k))$

Problem *"administrative redexes"*

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x.\, M]\!] &= \lambda k.\, k\, (\lambda x.\, [\![M]\!]) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda m.\, [\![N]\!]\ (\lambda n.\, m\ n\ k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda x.\, [\![N]\!]\ k)
\end{aligned}
$$

Examples

- $[\![\lambda x. x]\!] = \lambda k.\, k\, (\lambda x k.\, k x)$
- $[\![(\lambda x. x)(\lambda x. x)]\!] =$
  $\lambda k. (\lambda k.\, k\, (\lambda x k.\, k x))\ (\lambda m. (\lambda k.\, k\, (\lambda x k.\, k x))\ (\lambda n.\, m\ n\ k))$

Proposition

Translate, then reduce administrative redexes (two passes).

Problem *“administrative redexes”*

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x.\, M]\!] &= \lambda k.\, k\, (\lambda x.\, [\![M]\!]) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda m.\, [\![N]\!]\ (\lambda n.\, m\, n\, k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda x.\, [\![N]\!]\ k)
\end{aligned}
$$

Examples

- $[\![\lambda x. x]\!] = \lambda k.\, k\, (\lambda x k.\, k x)$
- $[\![(\lambda x. x)(\lambda x. x)]\!] =$
  $\lambda k.\, (\lambda k.\, k\, (\lambda x k.\, k x))\ (\lambda m.\, (\lambda k.\, k\, (\lambda x k.\, k x))\ (\lambda n.\, m\, n\, k))$

Proposition

Translate, then reduce administrative redexes (two passes).
But how to distinguish administrative/source redexes?

Analysis Control flow in the CPS

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x. M]\!] &= \lambda k.\, k\, (\lambda x.\, [\![M]\!]) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda m.\, [\![N]\!]\, (\lambda n. m\, n\, k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda x.\, [\![N]\!]\, k)
\end{aligned}
$$

Analysis Control flow in the CPS

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x.M]\!] &= \lambda k.\, k\, (\lambda x.\, [\![M]\!]) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda m.\, [\![N]\!]\, (\lambda n.\, m\, n\, k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda x.\, [\![N]\!]\, k)
\end{aligned}
$$

1. where can the $\lambda k.$ occur in the residual term?

Analysis Control flow in the CPS

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x.M]\!] &= \lambda k.\, k\, (\lambda x.\, [\![M]\!]) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda m.\, [\![N]\!]\, (\lambda n.\, m\, n\, k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda x.\, [\![N]\!]\, k)
\end{aligned}
$$

1. where can the $\lambda k.$ occur in the residual term?

Analysis Control flow in the CPS

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x. M]\!] &= \lambda k.\, k\, (\lambda x k.\, [\![M]\!]\ k) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda m.\, [\![N]\!]\ (\lambda n. m\, n\, k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda x.\, [\![N]\!]\ k)
\end{aligned}
$$

1. where can the $\lambda k.$ occur in the residual term?

Analysis Control flow in the CPS

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x.M]\!] &= \lambda k.\, k\, (\lambda xk.\, [\![M]\!]\ k) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda m.\, [\![N]\!]\ (\lambda n.\, m\, n\, k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\ (\lambda x.\, [\![N]\!]\ k)
\end{aligned}
$$

1. where can the $\lambda k.$ occur in the residual term?
2. which terms can be denoted by the k?

Analysis Control flow in the CPS

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x.M]\!] &= \lambda k.\, k\, (\lambda x k.\, [\![M]\!]\, (\lambda m.\, k\, m)) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda m.\, [\![N]\!]\, (\lambda n.\, m\, n\, k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda x.\, [\![N]\!]\, (\lambda n.\, k\, n))
\end{aligned}
$$

1. where can the $\lambda k.$ occur in the residual term?
2. which terms can be denoted by the k?

Analysis Control flow in the CPS

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x.\, M]\!] &= \lambda k.\, k\, (\lambda x k.\, [\![M]\!]\, (\lambda m.\, k\, m)) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda m.\, [\![N]\!]\, (\lambda n.\, m\, n\, k)) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda x.\, [\![N]\!]\, (\lambda n.\, k\, n))
\end{aligned}
$$

1. where can the $\lambda k.$ occur in the residual term?
2. which terms can be denoted by the k?
3. where do these k occur?

Analysis Control flow in the CPS

$$
\begin{aligned}
[\![x]\!] &= \lambda k.\, k\, x \\
[\![\lambda x.M]\!] &= \lambda k.\, k\, (\lambda x k.\, [\![M]\!]\, (\lambda m.\, k\, m)) \\
[\![M\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda m.\, [\![N]\!]\, (\lambda n.\, m\, n\, (\lambda v.\, k\, v))) \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda k.\, [\![M]\!]\, (\lambda x.\, [\![N]\!]\, (\lambda n.\, k\, n))
\end{aligned}
$$

1. where can the $\lambda k.$ occur in the residual term?
2. which terms can be denoted by the k?
3. where do these k occur?

Analysis Control flow in the CPS

$$
\begin{aligned}
[\![x]\!] &= \lambda \underline{k}.\, \underline{k}[x] \\
[\![\lambda x. M]\!] &= \lambda \underline{k}.\, \underline{k}[\lambda x k.\, [\![M]\!]\, [\lambda \underline{m}.\, k\ \underline{m}]] \\
[\![M\ N]\!] &= \lambda \underline{k}.\, [\![M]\!]\, [\lambda \underline{m}.\, [\![N]\!]\, [\lambda \underline{n}.\, \underline{m}\ \underline{n}\ (\lambda v.\, \underline{k}[v])]] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda \underline{k}.\, [\![M]\!]\, [\lambda \underline{x}.\, [\![N]\!]\ [\lambda \underline{n}.\, \underline{k}[\underline{n}]]]
\end{aligned}
$$

1. where can the $\lambda k.$ occur in the residual term?
2. which terms can be denoted by the k?
3. where do these k occur?
4. what are the static abs. $\lambda \underline{x}. T$ and app. $T[U]$?

Analysis Control flow in the CPS

$$
\begin{aligned}
[\![x]\!] &= \lambda \underline{k}.\, \underline{k}[x] \\
[\![\lambda x.M]\!] &= \lambda \underline{k}.\, \underline{k}[\lambda x k.\, [\![M]\!]\,[\lambda \underline{m}.\, k\ \underline{m}]] \\
[\![M\ N]\!] &= \lambda \underline{k}.\, [\![M]\!]\,[\lambda \underline{m}.\, [\![N]\!]\,[\lambda \underline{n}.\, \underline{m}\ \underline{n}\ (\lambda v.\, \underline{k}[v])]] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!] &= \lambda \underline{k}.\, [\![M]\!]\,[\lambda \underline{m}.\, \mathbf{let}\, x = \underline{m}\, \mathbf{in}\, [\![N]\!]\ [\lambda \underline{n}.\, \underline{k}[\underline{n}]]]
\end{aligned}
$$

1. where can the $\lambda k.$ occur in the residual term?
2. which terms can be denoted by the k?
3. where do these k occur?
4. what are the static abs. $\lambda \underline{x}.T$ and app. $T[U]$?
5. are there variable mismatches?

Result The *one-pass* CPS transform

(Danvy & Filinski, *Representing Control*, 1991)

$$
\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M.k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\ N\ (\lambda v.\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\mathbf{let}\,x = M\ \mathbf{in}\ [\![N]\!]\,[\lambda N.\kappa[N]]]
\end{aligned}
$$

Result The *one-pass* CPS transform

(Danvy & Filinski, *Representing Control*, 1991)

$$[\![\cdot]\!]\ \cdot\ :\ M \to (M \to M) \to M$$

$$
\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!][\lambda M.k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\ N\ (\lambda v.\kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!][\lambda N.\kappa[N]]]
\end{aligned}
$$

Result The *one-pass* CPS transform

(Danvy & Filinski, *Representing Control*, 1991)

$$[\![\cdot]\!]\ \cdot\ :\ M \to (M \to M) \to M$$

$$\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M.\,k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.\,M\ N\ (\lambda v.\,\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!]\,[\lambda N.\,\kappa[N]]]
\end{aligned}$$

$$[\![\cdot]\!]\ :\ M \to M$$

$$[\![M]\!] = [\![M]\!]\,[?]$$

Result The *one-pass* CPS transform

(Danvy & Filinski, *Representing Control*, 1991)

$$[\![\cdot]\!]\ \cdot\ :\ M \to (M \to M) \to M$$

$$\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x. M]\!]\ \kappa &= \kappa[\lambda x k.\ [\![M]\!]\,[\lambda M. k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N. M\ N\ (\lambda v. \kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M. \mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!]\,[\lambda N. \kappa[N]]]
\end{aligned}$$

$$[\![\cdot]\!]\ :\ M \to M$$

$$[\![M]\!] = \lambda k.\ [\![M]\!]\,[k]$$

Result The *one-pass* CPS transform

(Danvy & Filinski, *Representing Control*, 1991)

$$[\![\cdot]\!]\ \cdot\ :\ M \to (M \to M) \to M$$

$$\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x. M]\!]\ \kappa &= \kappa[\lambda x k.\ [\![M]\!]\,[\lambda M. k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N. M\ N\ (\lambda v. \kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M. \mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!]\,[\lambda N. \kappa[N]]]
\end{aligned}$$

$$[\![\cdot]\!]\ :\ M \to M$$

$$[\![M]\!] = \lambda k.\ [\![M]\!]\,[\lambda M. k\ M]$$

Result The *one-pass* CPS transform

(Danvy & Filinski, *Representing Control*, 1991)

$$[\![\cdot]\!]\ \cdot\ :\ M \to (M \to M) \to M$$

$$\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!][\lambda M.k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\ N\ (\lambda v.\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!][\lambda N.\kappa[N]]]
\end{aligned}$$

$$[\![\cdot]\!]\ :\ M \to M$$

$$[\![M]\!] = \lambda k.\ [\![M]\!][\lambda M.k\ M]$$

Examples

- $[\![\lambda x.x]\!] = \lambda k.k\ (\lambda xk.k\ x)$

Result The *one-pass* CPS transform

(Danvy & Filinski, *Representing Control*, 1991)

$$[\![\cdot]\!]\ \cdot\ :\ M \to (M \to M) \to M$$

$$\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\,[\![M]\!]\,[\lambda M.\,k\,M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,[\![N]\!]\ (\lambda N.\,M\,N\,(\lambda v.\,\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,\mathbf{let}\,x = M\,\mathbf{in}\,[\![N]\!]\,[\lambda N.\,\kappa[N]]]
\end{aligned}$$

$$[\![\cdot]\!]\ :\ M \to M$$

$$[\![M]\!] = \lambda k.\,[\![M]\!]\,[\lambda M.\,k\,M]$$

Examples

- $[\![\lambda x.x]\!] = \lambda k.\,k\,(\lambda xk.\,k\,x)$
- $[\![(\lambda x.x)\ (\lambda x.x)]\!] = \lambda k.\,(\lambda xk.\,k\,x)(\lambda xk.\,k\,x)(\lambda v.\,k\,v)$

Result The *one-pass* CPS transform

(Danvy & Filinski, *Representing Control*, 1991)

$$[\![\cdot]\!]\ \cdot\ :\ M \to (M \to M) \to M$$

$$\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\,[\![M]\!][\lambda M.k\,M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,[\![N]\!]\ (\lambda N.M\,N\,(\lambda v.\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!][\lambda N.\kappa[N]]]
\end{aligned}$$

$$[\![\cdot]\!]\ :\ M \to M$$

$$[\![M]\!] = \lambda k.\,[\![M]\!][\lambda M.k\,M]$$

Examples

- $[\![\lambda x.x]\!] = \lambda k.k\ (\lambda xk.k\ x)$
- $[\![(\lambda x.x)\ (\lambda x.x)]\!] = \lambda k.(\lambda xk.k\ x)(\lambda xk.k\ x)(\lambda v.k\ v)$
- $[\![\lambda fx.f\ x]\!] = \lambda k.k\ (\lambda fk.k\ \lambda xk.k\ (fx(\lambda v.k\ v)))$

Question What is the structure of CPS terms?

$$
\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!][\lambda M.k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\ N\ (\lambda v.\kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!][\lambda N.\kappa[N]]]
\end{aligned}
$$

$$[\![M]\!] = \lambda k.\ [\![M]\!]\ [\lambda M.k\ M]$$

Quiz

Is there M s.t. $[\![M]\!] = \lambda k.k\ (\lambda xk.x)$?

Question What is the structure of CPS terms?

$$
\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x. M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M. k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N. M\ N\ (\lambda v. \kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M. \mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!]\,[\lambda N. \kappa[N]]]
\end{aligned}
$$

$$
[\![M]\!] = \lambda k.\ [\![M]\!]\ [\lambda M. k\ M]
$$

Quiz

Is there M s.t. $[\![M]\!] = \lambda k. k\ (\lambda xk. x)$?

What is the image of the one-pass CPS transform?

Question What is the structure of CPS terms?

$$
\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!][\lambda M.k\,M]] \\
[\![M\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\,N\,(\lambda v.\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\mathbf{let}\,x = M\,\mathbf{in}\,[\![N]\!][\lambda N.\kappa[N]]]
\end{aligned}
$$

$$[\![M]\!] = \lambda k.\ [\![M]\!]\ [\lambda M.k\,M]$$

Quiz

Is there M s.t. $[\![M]\!] = \lambda k.k\,(\lambda xk.x)$?
What is the image of the one-pass CPS transform?

Motivation

A precise syntax for CPS terms?

Analysis Output syntax of the one-pass CPS

$$
\begin{aligned}
[\![\cdot]\!]\ \cdot\ &:\ M \to (M \to M) \to M \\
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M.\,k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\ N\ (\lambda v.\,\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!]\,[\lambda N.\,\kappa[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![\cdot]\!]\ &:\ M \to M \\
[\![M]\!] &= \lambda k.\ [\![M]\!]\,[\lambda M.\,k\ M]
\end{aligned}
$$

Analysis Output syntax of the one-pass CPS

$$
\begin{aligned}
[\![\cdot]\!]\ \cdot\ &:\ M \to (T \to S) \to U \\
[\![x]\!]\ \kappa &= \kappa\,[x] \\
[\![\lambda x.\,M]\!]\ \kappa &= \kappa\,[\lambda x k.\ [\![M]\!]\,[\lambda M.\,k\,M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.\,M\,N\,(\lambda v.\,\kappa\,[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!]\,[\lambda N.\,\kappa\,[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![\cdot]\!]\ &:\ M \to P \\
[\![M]\!] &= \lambda k.\ [\![M]\!]\,[\lambda M.\,k\,M]
\end{aligned}
$$

$S ::=$

$T ::=$

$P ::=$

Analysis Output syntax of the one-pass CPS

$$
\begin{aligned}
[\![\cdot]\!]\ \cdot &: M \to (T \to S) \to U \\
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x. M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M. k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N. M\ N\ (\lambda v. \kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M. \mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!]\,[\lambda N. \kappa[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![\cdot]\!] &: M \to P \\
[\![M]\!] &= \lambda k.\ [\![M]\!]\,[\lambda M. k\ M]
\end{aligned}
$$

$S ::=$

$T ::=$

$P ::=$

Analysis Output syntax of the one-pass CPS

$$\begin{aligned}
[\![\cdot]\!]\ \cdot\ &:\ M \to (T \to S) \to S \\
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\,[\![M]\!]\,[\lambda M.\,k\,M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,[\![N]\!]\ (\lambda N.\,M\,N\,(\lambda v.\,\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!]\,[\lambda N.\,\kappa[N]]]
\end{aligned}$$

$$\begin{aligned}
[\![\cdot]\!]\ &:\ M \to P \\
[\![M]\!] &= \lambda k.\,[\![M]\!]\,[\lambda M.\,k\,M]
\end{aligned}$$

$S ::=$

$T ::=$

$P ::=$

Analysis Output syntax of the one-pass CPS

$$
\begin{aligned}
[\![\cdot]\!]\ \cdot\ &:\ M \to (T \to S) \to S \\
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M.k\,M]] \\
[\![M\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\,N\,(\lambda v.\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!]\,[\lambda N.\kappa[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![\cdot]\!]\ &:\ M \to P \\
[\![M]\!] &= \lambda k.\ [\![M]\!]\,[\lambda M.k\,M]
\end{aligned}
$$

$S ::=$

$T ::=$

$P ::=$

Analysis Output syntax of the one-pass CPS

$$
\begin{aligned}
[\![\cdot]\!]\ \cdot\ &:\ M \to (T \to S) \to S \\
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x. M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!][\lambda M. k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N. M\ N\ (\lambda v. \kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M. \mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!][\lambda N. \kappa[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![\cdot]\!]\ &:\ M \to P \\
[\![M]\!] &= \lambda k.\ [\![M]\!][\lambda M. k\ M]
\end{aligned}
$$

$S ::=$

$T ::= \lambda xk. S \mid x \mid v$

$P ::=$

Analysis Output syntax of the one-pass CPS

$$
\begin{aligned}
[\![\cdot]\!]\ \cdot\ &:\ M \to (T \to S) \to S \\
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M.\,k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.\,M\ N\ (\lambda v.\,\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!]\,[\lambda N.\,\kappa[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![\cdot]\!]\ &:\ M \to P \\
[\![M]\!] &= \lambda k.\ [\![M]\!]\,[\lambda M.\,k\ M]
\end{aligned}
$$

$S ::=$

$T ::= \lambda xk.S \mid x \mid v$

$P ::=$

Analysis Output syntax of the one-pass CPS

$$
\begin{aligned}
[\![\cdot]\!]\ \cdot &: M \to (T \to S) \to S \\
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M.\,k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.\,M\ N\ (\lambda v.\,\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!]\,[\lambda N.\,\kappa[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![\cdot]\!] &: M \to P \\
[\![M]\!] &= \lambda k.\ [\![M]\!]\,[\lambda M.\,k\ M]
\end{aligned}
$$

$$S ::= k\ T \mid T\ T\ (\lambda v.\,S) \mid \mathbf{let}\,x = T\,\mathbf{in}\,S$$

$$T ::= \lambda xk.\,S \mid x \mid v$$

$$P ::=$$

Analysis Output syntax of the one-pass CPS

$$
\begin{aligned}
[\![\cdot]\!]\ \cdot\ &:\ M \to (T \to S) \to S \\
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M.\,k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.\,M\ N\ (\lambda v.\,\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!]\,[\lambda N.\,\kappa[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![\cdot]\!]\ &:\ M \to P \\
[\![M]\!] &= \lambda k.\ [\![M]\!]\,[\lambda M.\,k\ M]
\end{aligned}
$$

$$
\begin{aligned}
S &::= k\ T \mid T\ T\ (\lambda v.\,S) \mid \mathbf{let}\,x = T\,\mathbf{in}\,S \\
T &::= \lambda xk.\,S \mid x \mid v \\
P &::=
\end{aligned}
$$

Analysis Output syntax of the one-pass CPS

$$
\begin{aligned}
[\![\cdot]\!] \cdot &: M \to (T \to S) \to S \\
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!][\lambda M.\, k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.\, M\ N\ (\lambda v.\, \kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\, \mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!][\lambda N.\, \kappa[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![\cdot]\!] &: M \to P \\
[\![M]\!] &= \lambda k.\ [\![M]\!][\lambda M.\, k\ M]
\end{aligned}
$$

$$
\begin{aligned}
S &::= k\ T \mid T\ T\ (\lambda v.\, S) \mid \mathbf{let}\, x = T\, \mathbf{in}\, S \\
T &::= \lambda xk.\, S \mid x \mid v \\
P &::= \lambda k.\, S
\end{aligned}
$$

Analysis Output syntax of the one-pass CPS

$$
\begin{aligned}
[\![\cdot]\!]\ \cdot\ &:\ M \to (T \to S) \to S \\
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x. M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M.\, k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.\, M\ N\ (\lambda v.\, \kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\, \mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!]\,[\lambda N.\, \kappa[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![\cdot]\!]\ &:\ M \to P \\
[\![M]\!] &= \lambda k.\ [\![M]\!]\,[\lambda M.\, k\ M]
\end{aligned}
$$

$S ::= k\ T \mid T\ T\ (\lambda v.\, S) \mid \mathbf{let}\, x = T\, \mathbf{in}\, S$	Serious terms
$T ::= \lambda xk.\, S \mid x \mid v$	Trival terms
$P ::= \lambda k.\, S$	Programs

Result The syntax of CPS terms

$S ::= k\ T \mid T\ T\ (\lambda v. S) \mid \mathbf{let}\, x = T\, \mathbf{in}\, S$	Serious terms
$T ::= \lambda xk. S \mid x \mid v$	Trival terms
$P ::= \lambda k. S$	Programs

Result The syntax of CPS terms

$$S ::= k\ T \mid T\ T\ (\lambda v.\, S) \mid \mathbf{let}\, x = T\, \mathbf{in}\, S \qquad \text{Serious terms}$$

$$T ::= \lambda xk.\, S \mid x \mid v \qquad \text{Trival terms}$$

$$P ::= \lambda k.\, S \qquad \text{Programs}$$

Notes

- distinguished x (source), v (value), k (continuation) var.
- $(\lambda v.\, S)$ is a *continuation*
- programs await the *initial* continuation

Result The syntax of CPS terms

$$S ::= \textbf{ret}_k\ T \mid \textbf{bind}\ v = T\ T\ \textbf{in}\ S \mid \textbf{let}\, x = T\ \textbf{in}\, S \qquad \text{Serious terms}$$
$$T ::= \lambda xk.\, S \mid x \mid v \qquad \text{Trival terms}$$
$$P ::= \lambda k.\, S \qquad \text{Programs}$$

Notes

- distinguished x (source), v (value), k (continuation) var.
- $(\lambda v.\, S)$ is a *continuation*
- programs await the *initial* continuation
- the continuation monad

Result The syntax of CPS terms

$$S ::= \mathbf{ret}_k\ T \mid \mathbf{bind}\ v = T\ T\ \mathbf{in}\ S \mid \mathbf{let}\, x = T\ \mathbf{in}\, S \qquad \text{Serious terms}$$

$$T ::= \lambda xk.\, S \mid x \mid v \qquad \text{Trival terms}$$

$$P ::= \lambda k.\, S \qquad \text{Programs}$$

Notes

- distinguished x (source), v (value), k (continuation) var.
- $(\lambda v.\, S)$ is a *continuation*
- programs await the *initial* continuation
- the continuation monad

To go further

With typed input/output syntax, we can *deduce* typing of CPS terms (see draft).

Problem β-redexes or **let**s?

$$
\begin{aligned}
&[\![(\lambda xy.\,x)\; a\; b]\!] = \\
&\qquad \lambda k.\,(\lambda xk.\,k\;(\lambda yk.\,k\;x))\;a\;(\lambda v.\,v\;b\;(\lambda w.\,k\;w))
\end{aligned}
$$

Problem β-redexes or **let**s?

$$[\![(\mathbf{let}\,x = a\,\mathbf{in}\,\lambda y.\,x)\;b]\!] = \\ \lambda k.\,\mathbf{let}\,x = a\,\mathbf{in}\,(\lambda y.\,x)\;b\;(\lambda v.\,k v)$$

Problem β-redexes or **let**s?

$$[\![\mathbf{let}\, x = a\, \mathbf{in}\, (\lambda y.x)\; b]\!] = \lambda k.\, \mathbf{let}\, x = a\, \mathbf{in}\, (\lambda y.x)\; b\; (\lambda v.\, kv)$$

Problem β-redexes or **lets**?

$$\llbracket \mathbf{let}\, x = a \,\mathbf{in}\, \mathbf{let}\, y = b \,\mathbf{in}\, x \rrbracket = \lambda k.\, \mathbf{let}\, x = a \,\mathbf{in}\, \mathbf{let}\, y = b \,\mathbf{in}\, k\ x$$

Problem β-redexes or **lets**?

$$[\![\textbf{let}\, x = a \,\textbf{in}\, \textbf{let}\, y = b \,\textbf{in}\, x]\!] = \lambda k.\, \textbf{let}\, x = a \,\textbf{in}\, \textbf{let}\, y = b \,\textbf{in}\, k\ x$$

Remarks

- two representations for redexes in CPS terms (β redexes and **let**)
- **let** gives more compact CPS terms
- no more apparent β-redexes with **let**
- ... if considering *nested* β-redexes

Problem β-redexes or **lets**?

$$[\![\mathbf{let}\, x = a \,\mathbf{in}\, \mathbf{let}\, y = b \,\mathbf{in}\, x]\!] = \lambda k.\, \mathbf{let}\, x = a \,\mathbf{in}\, \mathbf{let}\, y = b \,\mathbf{in}\, k\ x$$

Remarks

- two representations for redexes in CPS terms (β redexes and **let**)
- **let** gives more compact CPS terms
- no more apparent β-redexes with **let**
- ... if considering *nested* β-redexes

Motivation

More compact CPS terms (Sabry & Felleisen, 1993) (Danvy 2004)
let reordering optimizations

Problem β-redexes or **lets**?

$$[\![\mathbf{let}\, x = a\, \mathbf{in}\, \mathbf{let}\, y = b\, \mathbf{in}\, x]\!] = \lambda k.\, \mathbf{let}\, x = a\, \mathbf{in}\, \mathbf{let}\, y = b\, \mathbf{in}\, k\, x$$

Remarks

- two representations for redexes in CPS terms (β redexes and **let**)
- **let** gives more compact CPS terms
- no more apparent β-redexes with **let**
- . . . if considering *nested* β-redexes

Motivation

More compact CPS terms (Sabry & Felleisen, 1993) (Danvy 2004)
let reordering optimizations

Proposition

Nested redexes → **let**s, then CPS-transformation (2-pass)?

Analysis The syntax of β-normal CPS terms

$$S ::= k\ T \mid T\ T\ (\lambda v.\,S) \mid \mathbf{let}\, x = T\, \mathbf{in}\, S \qquad \text{Serious terms}$$

$$T ::= \lambda x k.\,S \mid x \mid v \qquad \text{Trival terms}$$

$$P ::= \lambda k.\,S \qquad \text{Programs}$$

Analysis The syntax of β-normal CPS terms

$$S ::= k\ T \mid T\ T\ (\lambda v.\, S) \mid \textbf{let}\, x = T\, \textbf{in}\, S \qquad \text{Serious terms}$$

$$T ::= \lambda x k.\, S \mid x \mid v \qquad \text{Trival terms}$$

$$P ::= \lambda k.\, S \qquad \text{Programs}$$

Analysis The syntax of β-normal CPS terms

$$S ::= k\ T \mid I\ T\ (\lambda v.\, S) \mid \mathbf{let}\, x = T\, \mathbf{in}\, S$$ Serious terms

$$T ::= \lambda x k.\, S \mid I$$ Trival terms

$$I ::= x \mid v$$ Identifiers

$$P ::= \lambda k.\, S$$ Programs

Analysis The syntax of β-normal CPS terms

$S ::= k\ T \mid I\ T\ (\lambda v.\,S) \mid \textbf{let}\, x = T\, \textbf{in}\, S$	Serious terms
$T ::= \lambda xk.\,S \mid I$	Trival terms
$I ::= x \mid v$	Identifiers
$P ::= \lambda k.\,S$	Programs

Remarks

- identifiers = "atomic terms"
- CPS is now context-sensitive

Result CPS transformation of β-redexes (Danvy, 2004)

$$\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x. M]\!]\ \kappa &= \kappa[\lambda x k.\ [\![M]\!][\lambda M. k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M. [\![N]\!]\ (\lambda N. M\ N\ (\lambda v. \kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M. \mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!][\lambda N. \kappa[N]]]
\end{aligned}$$

Result CPS transformation of β-redexes (Danvy, 2004)

$$[\![x]\!]_l\ \kappa = \kappa[\psi_l(x)]$$
$$[\![\lambda x.M]\!]_0\ \kappa = \kappa[\lambda xk.\ [\![M]\!]_0[\lambda T.k\ T]]$$

$$[\![M\ N]\!]_l\ \kappa = [\![M]\!]_{S(l)}[\lambda T.\ [\![N]\!]_l[\lambda U.T[U][\lambda V.\kappa[V]]]]$$
$$[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]_l\ \kappa = \kappa[\lambda T\kappa.\mathbf{let}\, x = T\, \mathbf{in}\ [\![M]\!]_l[\lambda M.\kappa[M]]]$$

$$\psi_0(I) = i$$

Result CPS transformation of β-redexes (Danvy, 2004)

$$
\begin{aligned}
[\![x]\!]_l\ \kappa &= \kappa[\psi_l(x)] \\
[\![\lambda x.M]\!]_0\ \kappa &= \kappa[\lambda xk.\,[\![M]\!]_0[\lambda T.k\ T]] \\
[\![\lambda x.M]\!]_{S(l)}\ \kappa &= \kappa[\lambda T\kappa.\,\mathbf{let}\,x = T\,\mathbf{in}\,[\![M]\!]_l[\lambda M.\kappa[M]]] \\
[\![M\ N]\!]_l\ \kappa &= [\![M]\!]_{S(l)}[\lambda T.\,[\![N]\!]_l[\lambda U.T[U][\lambda V.\kappa[V]]]] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]_l\ \kappa &= \kappa[\lambda T\kappa.\,\mathbf{let}\,x = T\,\mathbf{in}\,[\![M]\!]_l[\lambda M.\kappa[M]]]
\end{aligned}
$$

$$\psi_0(I) = i$$

Result CPS transformation of β-redexes (Danvy, 2004)

$$
\begin{aligned}
[\![x]\!]_l\ \kappa &= \kappa[\psi_l(x)] \\
[\![\lambda x.M]\!]_0\ \kappa &= \kappa[\lambda xk.\,[\![M]\!]_0[\lambda T.k\ T]] \\
[\![\lambda x.M]\!]_{S(l)}\ \kappa &= \kappa[\lambda T\kappa.\,\mathbf{let}\,x = T\,\mathbf{in}\,[\![M]\!]_l[\lambda M.\kappa[M]]] \\
[\![M\ N]\!]_l\ \kappa &= [\![M]\!]_{S(l)}[\lambda T.\,[\![N]\!]_l[\lambda U.T[U][\lambda V.\kappa[V]]]] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]_l\ \kappa &= \kappa[\lambda T\kappa.\,\mathbf{let}\,x = T\,\mathbf{in}\,[\![M]\!]_l[\lambda M.\kappa[M]]]
\end{aligned}
$$

$$
\begin{aligned}
\psi_0(I) &= i \\
\psi_{S(l)} &= \lambda T\kappa.IT(\lambda v.\kappa[\psi_l(v)])
\end{aligned}
$$

Result CPS transformation of β-redexes (Danvy, 2004)

$$[\![\cdot]\!]_{\cdot}\ \cdot : \forall l : \mathbb{N}, M \to (\tau_l \to S) \to S$$

$$\begin{aligned}
[\![x]\!]_l\ \kappa &= \kappa[\psi_l(x)] \\
[\![\lambda x.M]\!]_0\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]_0[\lambda T.k\ T]] \\
[\![\lambda x.M]\!]_{S(l)}\ \kappa &= \kappa[\lambda T\kappa.\,\mathbf{let}\,x = T\,\mathbf{in}\ [\![M]\!]_l[\lambda M.\kappa[M]]] \\
[\![M\ N]\!]_l\ \kappa &= [\![M]\!]_{S(l)}[\lambda T.\ [\![N]\!]_l[\lambda U.T[U][\lambda V.\kappa[V]]]] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]_l\ \kappa &= \kappa[\lambda T\kappa.\,\mathbf{let}\,x = T\,\mathbf{in}\ [\![M]\!]_l[\lambda M.\kappa[M]]]
\end{aligned}$$

$$\psi_{\cdot}(\cdot) : \forall l : \mathbb{N}, I \to \tau_l$$

$$\begin{aligned}
\psi_0(I) &= i \\
\psi_{S(l)} &= \lambda T\kappa.IT(\lambda v.\kappa[\psi_l(v)])
\end{aligned}$$

Result CPS transformation of β-redexes (Danvy, 2004)

$$\tau_0 = T$$
$$\tau_{S(l)} = T \to (\tau_l \to S) \to S$$
$$[\![\cdot]\!]_\cdot \cdot : \forall l : \mathbb{N}, M \to (\tau_l \to S) \to S$$

$$[\![x]\!]_l\ \kappa = \kappa[\psi_l(x)]$$
$$[\![\lambda x.M]\!]_0\ \kappa = \kappa[\lambda xk.\ [\![M]\!]_0[\lambda T.k\ T]]$$
$$[\![\lambda x.M]\!]_{S(l)}\ \kappa = \kappa[\lambda T\kappa.\,\mathbf{let}\,x = T\,\mathbf{in}\ [\![M]\!]_l[\lambda M.\kappa[M]]]$$
$$[\![M\ N]\!]_l\ \kappa = [\![M]\!]_{S(l)}[\lambda T.\ [\![N]\!]_l[\lambda U.T[U][\lambda V.\kappa[V]]]]$$
$$[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]_l\ \kappa = \kappa[\lambda T\kappa.\,\mathbf{let}\,x = T\,\mathbf{in}\ [\![M]\!]_l[\lambda M.\kappa[M]]]$$

$$\psi_\cdot(\cdot) : \forall l : \mathbb{N}, I \to \tau_l$$
$$\psi_0(I) = i$$
$$\psi_{S(l)} = \lambda T\kappa.IT(\lambda v.\kappa[\psi_l(v)])$$

Problem η-redexes for tail calls

$$\llbracket \lambda x.f\,x\,(g\,x) \rrbracket = \\ \lambda k.\,k\,(\lambda xk.\,g\,x\,(\lambda v.f\,v\,(\lambda w.\,k\,w)))$$

Problem η-redexes for tail calls

$$
\begin{aligned}
&[\![\lambda x. f\ x\ (g\ x)]\!] = \\
&\qquad \lambda k.\, k\ (\lambda x k.\, g\ x\ (\lambda v. f\ v\ (\lambda w.\, k\ w)))
\end{aligned}
$$

Problem η-redexes for tail calls

$$[\![\lambda x.f\,x\,(g\,x)]\!] =$$

$$\lambda k.\,k\,(\lambda xk.\,g\,x\,(\lambda v.f\,v\,k))$$

Problem η-redexes for tail calls

$$[\![\lambda x. f\ x\ (g\ x)]\!] = \lambda k. k\ (\lambda x k. g\ x\ (\lambda v. f\ v\ k))$$

Remark

- tail calls generate "η-redex"
- induces more (administrative?) substitutions

Motivation

Support for tail calls in later passes

Proposition

CPS-transformation, then η-reduction?

Analysis The syntax of tail-recursive CPS terms

$P ::= \lambda k.\, S$	Programs
$S ::= k\ T \mid I\ T\ (\lambda v.\, S) \mid \mathbf{let}\, x = T\, \mathbf{in}\, S$	Serious terms
$T ::= \lambda xk.\, S \mid I$	Trival terms
$I ::= x \mid v$	Identifiers

Analysis The syntax of tail-recursive CPS terms

$P ::= \lambda k.\, S$	Programs
$S ::= k\ T \mid I\ T\ (\lambda v.\, S) \mid \mathbf{let}\, x = T\, \mathbf{in}\, S$	Serious terms
$T ::= \lambda xk.\, S \mid I$	Trival terms
$I ::= x \mid v$	Identifiers

- continuations can be k

Analysis The syntax of tail-recursive CPS terms

$P ::= \lambda k.\, S$	Programs
$S ::= k\ T \mid I\ T\ C \mid \mathbf{let}\, x = T\, \mathbf{in}\, S$	Serious terms
$C ::= \lambda v.\, S \mid k$	Continuations
$T ::= \lambda xk.\, S \mid I$	Trival terms
$I ::= x \mid v$	Identifiers

- continuations can be k

Analysis The syntax of tail-recursive CPS terms

$P ::= \lambda k.\, S$	Programs
$S ::= k\ T \mid I\ T\ C \mid \mathbf{let}\, x = T\, \mathbf{in}\, S$	Serious terms
$C ::= \lambda v.\, S \mid k$	Continuations
$T ::= \lambda xk.\, S \mid I$	Trival terms
$I ::= x \mid v$	Identifiers

- continuations can be k
- continuations cannot be $(\lambda v.\, k\ v)$

Analysis The syntax of tail-recursive CPS terms

$$
\begin{array}{rcll}
P & ::= & \lambda k.\, S & \text{Programs} \\
S & ::= & k\ T \mid U & \text{Serious terms} \\
U & ::= & I\ T\ C \mid \mathbf{let}\, x = T\, \mathbf{in}\, S & \text{Computations} \\
C & ::= & \lambda v.\, U \mid k & \text{Continuations} \\
T & ::= & \lambda xk.\, S \mid I & \text{Trival terms} \\
I & ::= & x \mid v & \text{Identifiers}
\end{array}
$$

- continuations can be k
- continuations cannot be $(\lambda v.\, k\ v)$

Analysis The syntax of tail-recursive CPS terms

$P ::= \lambda k.\, S$	Programs
$S ::= k\ T \mid U$	Serious terms
$U ::= I\ T\ C \mid \textbf{let}\, x = T\, \textbf{in}\, S$	Computations
$C ::= \lambda v.\, U \mid k$	Continuations
$T ::= \lambda xk.\, S \mid I$	Trival terms
$I ::= x \mid v$	Identifiers

- continuations can be k
- continuations cannot be $(\lambda v.\, k\ v)$

Result Tail-recursive, β-normal CPS transformation

$$
\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]\,[\lambda M.\,k\ M]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\ N\ (\lambda v.\kappa[v]))] \\
[\![\mathbf{let}\,x = M\,\mathbf{in}\,N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\,\mathbf{let}\,x = M\,\mathbf{in}\ [\![N]\!]\,[\lambda N.\kappa[N]]]
\end{aligned}
$$

Result Tail-recursive, β-normal CPS transformation

$$
\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x. M]\!]\ \kappa &= \kappa[\lambda x k.\ [\![M]\!]'[k]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N. M\ N\ (\lambda v. \kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M. \mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!]\, [\lambda N. \kappa[N]]]
\end{aligned}
$$

Result Tail-recursive, β-normal CPS transformation

$$
\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]'[k]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\ N\ (\lambda v.\kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!][\lambda N.\kappa[N]]]
\end{aligned}
$$

$$
\begin{aligned}
[\![x]\!]'\ k &= k\ x \\
[\![\lambda x.M]\!]'\ k &= k\ (\lambda xk.\ [\![M]\!]'[k]) \\
[\![M\ N]\!]'\ k &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\ N\ k)] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]'\ k &= [\![M]\!]\ [\lambda M.\mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!][\lambda N.\kappa[N]]]
\end{aligned}
$$

Result Tail-recursive, β-normal CPS transformation

$$[\![\cdot]\!]\ \cdot : M \to (T \to U) \to U$$

$$
\begin{aligned}
[\![x]\!]\ \kappa &= \kappa[x] \\
[\![\lambda x.M]\!]\ \kappa &= \kappa[\lambda xk.\ [\![M]\!]'[k]] \\
[\![M\ N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\ N\ (\lambda v.\kappa[v]))] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]\ \kappa &= [\![M]\!]\ [\lambda M.\mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!][\lambda N.\kappa[N]]]
\end{aligned}
$$

$$[\![\cdot]\!]\ \cdot : M \to k \to S$$

$$
\begin{aligned}
[\![x]\!]'\ k &= k\ x \\
[\![\lambda x.M]\!]'\ k &= k\ (\lambda xk.\ [\![M]\!]'[k]) \\
[\![M\ N]\!]'\ k &= [\![M]\!]\ [\lambda M.\ [\![N]\!]\ (\lambda N.M\ N\ k)] \\
[\![\mathbf{let}\, x = M\, \mathbf{in}\, N]\!]'\ k &= [\![M]\!]\ [\lambda M.\mathbf{let}\, x = M\, \mathbf{in}\ [\![N]\!][\lambda N.\kappa[N]]]
\end{aligned}
$$

Conclusion

In the draft

- the first CPS
  - one-pass
  - tail-recursive
  - β-normal
  - in a dedicated syntax
- all the code in OCaml
- simply typed input/output syntax (GADT)
  (type-preserving transformations)

Conclusion

In the draft

- the first CPS
  - one-pass
  - tail-recursive
  - β-normal
  - in a dedicated syntax
- all the code in OCaml
- simply typed input/output syntax (GADT)
  (type-preserving transformations)

"Type-directed transformation optimization"

pathological example $\rightsquigarrow$ optimization
$\rightsquigarrow$ syntax/typing modification
$\rightsquigarrow$ algorithm modification