Monads And Continuation-Passing Style

Finn Lawler

November 9, 2008
Outline

Overview
  Review of Monads
  Pure Languages and State

Continuations and Contexts
  Evaluation Contexts
  Manipulating Contexts
  Typing CPS Terms

Relating Monads to CPS
  The Translations
  Properties of the Translations

Issues
Overview
  Review of Monads
  Pure Languages and State

Continuations and Contexts
  Evaluation Contexts
  Manipulating Contexts
  Typing CPS Terms

Relating Monads to CPS
  The Translations
  Properties of the Translations

Issues
Monads on semantic categories (Moggi)

- ‘Pure’ language $\Lambda$ with semantics in category $\mathcal{C}$
- Monad $\langle T : \mathcal{C} \rightarrow \mathcal{C}, \eta, \mu \rangle$ gives ‘notion of computation’, i.e. state, on $\mathcal{C}$
- $\Lambda$ extended with $T$ has semantics in $\mathcal{C}_T$ (Kleisli category of $T$):

\[
\begin{align*}
A_T &= A \\
(f : A \rightarrow B)_T &= f_T : A \rightarrow TB \\
(A \xrightarrow{f} B \xrightarrow{g} C)_T &= A \xrightarrow{f_T} TB \xrightarrow{Tg_T} T^2C \xrightarrow{\mu_C} TC
\end{align*}
\]
Moggi’s metalanguage $\Lambda_{ml}$

**Pure constructs**

\[\Gamma, x : A \vdash x : A\]

\[\Gamma, x : A \vdash M : TA \quad \Gamma \vdash \lambda x. M : A \to TB\]

\[\Gamma \vdash M : A \to TB \quad \Gamma \vdash N : A \quad \Gamma \vdash MN : TB\]

**Monadic constructs**

\[\Gamma \vdash M : A \quad \Gamma \vdash N : TA \quad \Gamma, x : A \vdash M : TB\]

\[\Gamma \vdash N \Rightarrow x. M : TB\]

**Reduction**

\[N \Rightarrow x. M \rightarrow_{\text{unit}} M[x := N]\]

\[(M_1 \Rightarrow x_1. M_2) \Rightarrow x_2. M_3 \rightarrow_{\text{assoc}} M_1 \Rightarrow x_1.(M_2 \Rightarrow x_2. M_3)\]
What is a pure language?

- Pure languages are nice. Why?
- Reductions are *local*: if
  
  \[ N \rightarrow^\beta N' \]

  then

  \[ M[N] \rightarrow^\beta M[N'] \]

- We can reason about a program’s behaviour by induction on its structure.
- So pure languages can’t handle global state.
A stateful language can be simulated in a pure language using a CPS transform.

Stateful terms now ‘return’ by executing a tail call to a (representation of a) superterm.

We want a transformation $T_\alpha$, parametric in a return address $\alpha$, so that:

$$\alpha[M_1[x := M_2]] \mapsto T_\beta(M_2)[\beta := T_\alpha(M_1)]$$

$T_\beta(M_2)$ should pass its result, along with any new state, to $\beta$. 
Overview
Review of Monads
Pure Languages and State

Continuations and Contexts
Evaluation Contexts
Manipulating Contexts
Typing CPS Terms

Relating Monads to CPS
The Translations
Properties of the Translations

Issues
Reduction in Evaluation Contexts

- Reductions take place inside contexts:
  \[
  \alpha[[M \ N]] \rightarrow_{\beta} \alpha[[[M] \ N]] \\
  \alpha[[\lambda x. M] \ N)] \rightarrow_{\beta} \alpha[M[x := N]]
  \]

- We recurse down the left subtrees of a term until we find a redex.
- This builds up a stack of terms with a ‘hole’ for a \(\lambda\)-abstraction.
- A redex arises when an abstraction is plugged into the hole of a stack.
Krivine’s Abstract Machine

- A state of the K-machine is a process

\[ \langle M | K \rangle \]

where \( M \) is a term and \( K \) a context (stack).

- Reduction rules:

\[
\begin{align*}
\langle MN | K \rangle & \rightarrow \langle M | N \cdot K \rangle \\
\langle \lambda x. M | N \cdot K \rangle & \rightarrow \langle M[x := N] | K \rangle
\end{align*}
\]

- The first is a *compilation* step.
- Compilation gives

\[
\alpha[(\lambda x. M) N_1 N_2 \ldots N_n] \quad \mapsto \quad \langle \lambda x. M | N_1 \cdot N_2 \cdot \ldots \cdot N_n \cdot \alpha \rangle
\]
A computation is a process waiting for a continuation or output channel.

Given a process

\[ \langle M|K \rangle \]

with a free continuation variable \( \alpha \),

\[ \kappa \alpha.\langle M|K \rangle \]

is a computation, and

\[ \langle \kappa \alpha.P|J \rangle \quad \rightarrow \quad P[\alpha := J] \]
A continuation is a process waiting for an input value.

Given a process

\[ \langle M | K \rangle \]

with a free variable \( x \),

\[ \bar{\kappa}x.\langle M | K \rangle \]

is a continuation, and

\[ \langle M | \bar{\kappa}x.P \rangle \rightarrow P[x := M] \]
Failure Of Confluence?

Consider

\[ P_1[\alpha := \bar{k}x.P_2] \leftarrow \langle \kappa\alpha.P_1|\bar{k}x.P_2 \rangle \rightarrow P_2[x := \kappa\alpha.P_1] \]

Does reduction fail to be confluent? In general, yes.
If \( P_1 \) is linear in \( \alpha \), then no.

Distinction:
- CPS as representing state/side-effects.
- CPS as representing non-standard control flow.

So (general) continuations are not side-effects.
Example: Some CPS transforms

- **Call-by-name:**

\[
\begin{align*}
\overline{x}_\alpha &= \langle x | \alpha \rangle \\
\overline{\lambda x. M}_\alpha &= \langle \lambda x. \kappa \beta. \overline{M}_\beta | \alpha \rangle \\
\overline{M N}_\alpha &= \langle \kappa \beta. \overline{M}_\beta | \overline{\kappa m. \langle \kappa \gamma. \overline{N}_\gamma | \overline{\kappa n (m | n \cdot \alpha) \rangle \rangle \rangle} \rangle
\end{align*}
\]

- **Call-by-value:**

\[
\begin{align*}
\overline{x}_\alpha &= \langle x | \alpha \rangle \\
\overline{\lambda x. M}_\alpha &= \langle \lambda x. \kappa \beta. \overline{M}_\beta | \alpha \rangle \\
\overline{M N}_\alpha &= \langle \kappa \beta. \overline{M}_\beta | \overline{\kappa m. \langle \kappa \gamma. \overline{N}_\gamma | \overline{\kappa n (m | n \cdot \alpha) \rangle \rangle \rangle} \rangle
\end{align*}
\]
Typing CPS Terms (1)

Computations

\[
\Gamma, x : A \vdash_{\text{cmp}} M : B \\
\Gamma \vdash_{\text{cmp}} \lambda x. M : A \to B
\]

\[
\Gamma; \alpha : A \vdash_{\text{prc}} C \\
\Gamma \vdash_{\text{cmp}} \kappa \alpha. C : A
\]

Continuations

\[
\Gamma; \alpha : A \vdash_{\text{cnt}} \alpha : A
\]

\[
\Gamma, x : A \vdash_{\text{prc}} C \\
\Gamma \vdash_{\text{cnt}} \bar{k} x. C : A
\]

\[
\Gamma \vdash_{\text{cnt}} K : B \\
\Gamma \vdash_{\text{cmp}} N : A
\]

\[
\Gamma \vdash_{\text{cnt}} N \cdot K : A \to B
\]
Typing CPS Terms (2)

Values

\[ \frac{}{\Gamma, x : A \vdash_{val} x : A} \]

\[ \frac{}{\Gamma \vdash_{val} M : A} \]

Processes

\[ \frac{\Gamma \vdash_{cmp} M : A \quad \Gamma \vdash_{cnt} K : A}{\Gamma \vdash_{prc} \langle M | K \rangle} \]
Overview
  Review of Monads
  Pure Languages and State

Continuations and Contexts
  Evaluation Contexts
  Manipulating Contexts
  Typing CPS Terms

Relating Monads to CPS
  The Translations
  Properties of the Translations

Issues
Translating $\Lambda_{ml}$ to CPS

\[
\begin{align*}
C_\alpha(x) &= \langle x | \alpha \rangle \\
C_\alpha(\lambda x. M) &= \langle \lambda x. \kappa \beta. C_\beta(M) | \alpha \rangle \\
C_\alpha(MN) &= \langle \kappa \beta. C_\beta(M) | \kappa \gamma. C_\gamma(N) \cdot \alpha \rangle \\
C_\alpha(M) &= \langle \kappa \beta. C_\beta(M) | \alpha \rangle \\
C_\alpha(M \Rightarrow x. N) &= \langle \kappa \beta. C_\beta(M) | \overline{\kappa} x. C_\alpha(N) \rangle
\end{align*}
\]
Translating (linear) CPS to Λ_{ml}

\[
C_{cmp}^{-1}(\kappa\alpha.P) = C_{prc}^{-1}(P)
\]
\[
C_{cmp}^{-1}(V) = V
\]
\[
C_{prc}^{-1}(\langle M|K\rangle) = C_{cnt}^{-1}(K)[C_{cmp}^{-1}(M)]
\]
\[
C_{cnt}^{-1}(\alpha) = [\cdot]
\]
\[
C_{cnt}^{-1}(\overline{\kappa}\times.P) = [\cdot] \Rightarrow \times.C_{prc}^{-1}(P)
\]
Properties

- $C$ and $C^{-1}$ are equational correspondences.
- Existing CPS transforms can be factored through $\Lambda_{ml}$.
- A generic transform $C$ then compiles $\Lambda_{ml}$ to CPS in a sensible way.
Overview
Review of Monads
Pure Languages and State

Continuations and Contexts
Evaluation Contexts
Manipulating Contexts
Typing CPS Terms

Relating Monads to CPS
The Translations
Properties of the Translations

Issues
Issues

Non-linear CPS terms
What is the categorical interpretation of non-linear CPS? How do such terms relate to monads?

State or algebra?
What is the relationship between monads as CPS transforms and monads as algebraic structure?