Continuation-Passing Style Transforms

Type Theory and Coq

Vincent Koppen

15-05-2018

Paper

Title: Polymorphic Type Assignment and CPS Conversion
Authors: Robert Harpery and Mark Lillibridge
Year: 1993
Journal: LISP AND SYMBOLIC COMPUTATION: An International Journal

Overview

1. Untyped Terms
   - call-by-value
   - call-by-value (modified)
   - call-by-name
2. Simple Type Assignment

Overview

Untyped Terms

Terms

$$e ::= x \mid \lambda x.e \mid e_1 e_2 \mid \textbf{callcc} \mid \textbf{throw} \mid \textbf{let}\ x\ \textbf{be}\ e_1\ \textbf{in}\ e_2$$

- **callcc, throw** introduced last week
- **let** x **be** e_1 **in** e_2: needed for the polymorphic type assignment

Values (call-by-value)

$$v ::= x \mid \lambda x.e \mid \textbf{callcc} \mid \textbf{throw}$$

Untyped Terms

Terms

$$e ::= x \mid \lambda x.e \mid e_1 e_2 \mid \textbf{callcc} \mid \textbf{throw} \mid \textbf{let}\ x\ \textbf{be}\ e_1\ \textbf{in}\ e_2$$

- **callcc, throw** introduced last week
- **let** x **be** e_1 **in** e_2: needed for the polymorphic type assignment

Values (call-by-value)

$$v ::= x \mid \lambda x.e \mid \textbf{callcc} \mid \textbf{throw}$$

Untyped Terms

Terms

$$e ::= x \mid \lambda x.e \mid e_1 e_2 \mid \ \textbf{callcc} \ \mid \ \textbf{throw} \ \mid \ \textbf{let}\ x\ \textbf{be}\ e_1\ \textbf{in}\ e_2$$

- **callcc, throw** introduced last week
- **let** x **be** e_1 **in** e_2: needed for the polymorphic type assignment

Values (call-by-value)

$$v ::= x \mid \lambda x.e \mid \ \textbf{callcc} \ \mid \ \textbf{throw}$$

Untyped Terms

Terms

$$e ::= x \mid \lambda x.e \mid e_1 e_2 \mid \textbf{callcc} \mid \textbf{throw} \mid \textbf{let } x \textbf{ be } e_1 \textbf{ in } e_2$$

- **callcc, throw** introduced last week
- **let** x **be** e_1 **in** e_2: needed for the polymorphic type assignment

Values (call-by-value)

$$v ::= x \mid \lambda x.e \mid \textbf{callcc} \mid \textbf{throw}$$

CPS Transform (call-by-value)

Definition

$$
\begin{aligned}
|v|_{cbv} &= \lambda k.k||v||_{cbv} \\
|e_1 e_2|_{cbv} &= \lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
|\ \textbf{let}\ x\ \textbf{be}\ e_1\ \textbf{in}\ e_2|_{cbv} &= \lambda k.|e_1|_{cbv}(\lambda x.|e_2|_{cbv} k)
\end{aligned}
$$

$$
\begin{aligned}
||x||_{cbv} &= x \\
||\lambda x.e||_{cbv} &= \lambda x.|e|_{cbv} \\
||\ \textbf{callcc}\ ||_{cbv} &= \lambda f.\lambda k.fkk \\
||\ \textbf{throw}\ ||_{cbv} &= \lambda c.\lambda k.k(\lambda x.\lambda l.cx)
\end{aligned}
$$

Note:

- $|\_|_{cbv}$ for terms, $||\_||_{cbv}$ for values.
- **callcc** gets a function f which has k both as input and as continuation
- **throw** discard continuation l and continue with k

CPS Transform (call-by-value)

Definition

$$
\begin{aligned}
|v|_{cbv} &= \lambda k.k||v||_{cbv} \\
|e_1 e_2|_{cbv} &= \lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
|\ \textbf{let}\ x\ \textbf{be}\ e_1\ \textbf{in}\ e_2|_{cbv} &= \lambda k.|e_1|_{cbv}(\lambda x.|e_2|_{cbv} k)
\end{aligned}
$$

$$
\begin{aligned}
||x||_{cbv} &= x \\
||\lambda x.e||_{cbv} &= \lambda x.|e|_{cbv} \\
||\ \textbf{callcc}\ ||_{cbv} &= \lambda f.\lambda k.fkk \\
||\ \textbf{throw}\ ||_{cbv} &= \lambda c.\lambda k.k(\lambda x.\lambda l.cx)
\end{aligned}
$$

Note:

- $|\_|_{cbv}$ for terms, $||\_||_{cbv}$ for values.
- **callcc** gets a function f which has k both as input and as continuation
- **throw** discard continuation l and continue with k

Example

Let $|e_1|_{cbv} = \lambda k'.k'||v||_{cbv}$, $|e_2|_{cbv} = \lambda k''.k''||w||_{cbv}$.

We have:

$$
\begin{aligned}
|e_1 e_2|_{cbv} &= \lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda k''.k''||w||_{cbv})(\lambda x_2.x_1 x_2 k)) \\
&\to_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda x_2.x_1 x_2 k)||w||_{cbv}) \\
&\to_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.x_1||w||_{cbv} k) \\
&\to_\beta \lambda k.(\lambda x_1.x_1||w||_{cbv} k)||v||_{cbv} \\
&\to_\beta \lambda k.||v||_{cbv}||w||_{cbv} k
\end{aligned}
$$

Example

Let $|e_1|_{cbv} = \lambda k'.k'||v||_{cbv}$, $|e_2|_{cbv} = \lambda k''.k''||w||_{cbv}$.

We have:

$$
\begin{aligned}
|e_1 e_2|_{cbv} &= \lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda k''.k''||w||_{cbv})(\lambda x_2.x_1 x_2 k)) \\
&\to_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda x_2.x_1 x_2 k)||w||_{cbv}) \\
&\to_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.x_1||w||_{cbv} k) \\
&\to_\beta \lambda k.(\lambda x_1.x_1||w||_{cbv} k)||v||_{cbv} \\
&\to_\beta \lambda k.||v||_{cbv}||w||_{cbv} k
\end{aligned}
$$

Example

Let $|e_1|_{cbv} = \lambda k'.k'||v||_{cbv}$, $|e_2|_{cbv} = \lambda k''.k''||w||_{cbv}$.

We have:

$$\begin{aligned}
|e_1 e_2|_{cbv} &= \lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda k''.k''||w||_{cbv})(\lambda x_2.x_1 x_2 k)) \\
&\to_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda x_2.x_1 x_2 k)||w||_{cbv}) \\
&\to_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.x_1||w||_{cbv} k) \\
&\to_\beta \lambda k.(\lambda x_1.x_1||w||_{cbv} k)||v||_{cbv} \\
&\to_\beta \lambda k.||v||_{cbv}||w||_{cbv} k
\end{aligned}$$

Example

Let $|e_1|_{cbv} = \lambda k'.k'||v||_{cbv}$, $|e_2|_{cbv} = \lambda k''.k''||w||_{cbv}$.

We have:

$$
\begin{aligned}
|e_1 e_2|_{cbv} &= \lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda k''.k''||w||_{cbv})(\lambda x_2.x_1 x_2 k)) \\
&\rightarrow_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda x_2.x_1 x_2 k)||w||_{cbv}) \\
&\rightarrow_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.x_1||w||_{cbv} k) \\
&\rightarrow_\beta \lambda k.(\lambda x_1.x_1||w||_{cbv} k)||v||_{cbv} \\
&\rightarrow_\beta \lambda k.||v||_{cbv}||w||_{cbv} k
\end{aligned}
$$

Example

Let $|e_1|_{cbv} = \lambda k'.k'||v||_{cbv}$, $|e_2|_{cbv} = \lambda k''.k''||w||_{cbv}$.

We have:

$$\begin{aligned}
|e_1 e_2|_{cbv} &= \lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda k''.k''||w||_{cbv})(\lambda x_2.x_1 x_2 k)) \\
&\to_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda x_2.x_1 x_2 k)||w||_{cbv}) \\
&\to_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.x_1||w||_{cbv} k) \\
&\to_\beta \lambda k.(\lambda x_1.x_1||w||_{cbv} k)||v||_{cbv} \\
&\to_\beta \lambda k.||v||_{cbv}||w||_{cbv} k
\end{aligned}$$

Example

Let $|e_1|_{cbv} = \lambda k'.k'||v||_{cbv}$, $|e_2|_{cbv} = \lambda k''.k''||w||_{cbv}$.

We have:

$$\begin{aligned}
|e_1 e_2|_{cbv} &= \lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda k''.k''||w||_{cbv})(\lambda x_2.x_1 x_2 k)) \\
&\to_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.(\lambda x_2.x_1 x_2 k)||w||_{cbv}) \\
&\to_\beta \lambda k.(\lambda k'.k'||v||_{cbv})(\lambda x_1.x_1||w||_{cbv} k) \\
&\to_\beta \lambda k.(\lambda x_1.x_1||w||_{cbv} k)||v||_{cbv} \\
&\to_\beta \lambda k.||v||_{cbv}||w||_{cbv} k
\end{aligned}$$

CPS commutes with substitution

Lemma

$$||[v/x]v'||_{cbv} = [||v||_{cbv}/x]||v'||_{cbv}$$
$$|[v/x]e|_{cbv} = [||v||_{cbv}/x]|e|_{cbv}$$

Proof: simultaneous induction on the structure of v', e.

$v' = x'$

$x' = x$: $||[v/x]x||_{cbv} = ||v||_{cbv} = [||v||_{cbv}/x]||x||_{cbv}$
$x' \neq x$: $||[v/x]x'||_{cbv} = ||x'||_{cbv} = [||v||_{cbv}/x]||x'||_{cbv}$

$e = e_1 e_2$

$$\begin{aligned}
|[v/x](e_1 e_2)|_{cbv} &= |[v/x]e_1 [v/x]e_2|_{cbv} \\
&= \lambda k.|[v/x]e_1|_{cbv}(\lambda x_1.|[v/x]e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.[||v||_{cbv}/x]|e_1|_{cbv}(\lambda x_1.[||v||_{cbv}/x]|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]\lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]|e_1 e_2|_{cbv}
\end{aligned}$$

CPS commutes with substitution

Lemma

$$||[v/x]v'||_{cbv} = [||v||_{cbv}/x]||v'||_{cbv}$$
$$|[v/x]e|_{cbv} = [||v||_{cbv}/x]|e|_{cbv}$$

Proof: simultaneous induction on the structure of v', e.

$v' = x'$

$x' = x$: $||[v/x]x||_{cbv} = ||v||_{cbv} = [||v||_{cbv}/x]||x||_{cbv}$
$x' \neq x$: $||[v/x]x'||_{cbv} = ||x'||_{cbv} = [||v||_{cbv}/x]||x'||_{cbv}$

$e = e_1 e_2$

$$\begin{aligned}
|[v/x](e_1 e_2)|_{cbv} &= |[v/x]e_1 [v/x]e_2|_{cbv} \\
&= \lambda k.|[v/x]e_1|_{cbv}(\lambda x_1.|[v/x]e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.[||v||_{cbv}/x]|e_1|_{cbv}(\lambda x_1.[||v||_{cbv}/x]|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]\lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]|e_1 e_2|_{cbv}
\end{aligned}$$

CPS commutes with substitution

Lemma

$$||[v/x]v'||_{cbv} = [||v||_{cbv}/x]||v'||_{cbv}$$
$$|[v/x]e|_{cbv} = [||v||_{cbv}/x]|e|_{cbv}$$

Proof: simultaneous induction on the structure of v', e.

$v' = x'$

$x' = x$: $||[v/x]x||_{cbv} = ||v||_{cbv} = [||v||_{cbv}/x]||x||_{cbv}$
$x' \neq x$: $||[v/x]x'||_{cbv} = ||x'||_{cbv} = [||v||_{cbv}/x]||x'||_{cbv}$

$e = e_1 e_2$

$$\begin{aligned}
|[v/x](e_1 e_2)|_{cbv} &= |[v/x]e_1 [v/x]e_2|_{cbv} \\
&= \lambda k.|[v/x]e_1|_{cbv}(\lambda x_1.|[v/x]e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.[||v||_{cbv}/x]|e_1|_{cbv}(\lambda x_1.[||v||_{cbv}/x]|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]\lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]|e_1 e_2|_{cbv}
\end{aligned}$$

CPS commutes with substitution

Lemma

$$||[v/x]v'||_{cbv} = [||v||_{cbv}/x]||v'||_{cbv}$$
$$|[v/x]e|_{cbv} = [||v||_{cbv}/x]|e|_{cbv}$$

Proof: simultaneous induction on the structure of v', e.

$v' = x'$

$x' = x$: $||[v/x]x||_{cbv} = ||v||_{cbv} = [||v||_{cbv}/x]||x||_{cbv}$

$x' \neq x$: $||[v/x]x'||_{cbv} = ||x'||_{cbv} = [||v||_{cbv}/x]||x'||_{cbv}$

$e = e_1 e_2$

$$\begin{aligned}
|[v/x](e_1 e_2)|_{cbv} &= |[v/x]e_1 [v/x]e_2|_{cbv} \\
&= \lambda k.|[v/x]e_1|_{cbv}(\lambda x_1.|[v/x]e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.[||v||_{cbv}/x]|e_1|_{cbv}(\lambda x_1.[||v||_{cbv}/x]|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]\lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]|e_1 e_2|_{cbv}
\end{aligned}$$

CPS commutes with substitution

Lemma

$$||[v/x]v'||_{cbv} = [||v||_{cbv}/x]||v'||_{cbv}$$
$$|[v/x]e|_{cbv} = [||v||_{cbv}/x]|e|_{cbv}$$

Proof: simultaneous induction on the structure of v', e.

$v' = x'$

$x' = x$: $||[v/x]x||_{cbv} = ||v||_{cbv} = [||v||_{cbv}/x]||x||_{cbv}$
$x' \neq x$: $||[v/x]x'||_{cbv} = ||x'||_{cbv} = [||v||_{cbv}/x]||x'||_{cbv}$

$e = e_1 e_2$

$$\begin{aligned}
|[v/x](e_1 e_2)|_{cbv} &= |[v/x]e_1 [v/x]e_2|_{cbv} \\
&= \lambda k.|[v/x]e_1|_{cbv}(\lambda x_1.|[v/x]e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.[||v||_{cbv}/x]|e_1|_{cbv}(\lambda x_1.[||v||_{cbv}/x]|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]\lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]|e_1 e_2|_{cbv}
\end{aligned}$$

CPS commutes with substitution

Lemma

$$||[v/x]v'||_{cbv} = [||v||_{cbv}/x]||v'||_{cbv}$$
$$|[v/x]e|_{cbv} = [||v||_{cbv}/x]|e|_{cbv}$$

Proof: simultaneous induction on the structure of v', e.

$v' = x'$

$x' = x$: $||[v/x]x||_{cbv} = ||v||_{cbv} = [||v||_{cbv}/x]||x||_{cbv}$
$x' \neq x$: $||[v/x]x'||_{cbv} = ||x'||_{cbv} = [||v||_{cbv}/x]||x'||_{cbv}$

$e = e_1 e_2$

$$\begin{aligned}
|[v/x](e_1 e_2)|_{cbv} &= |[v/x]e_1 [v/x]e_2|_{cbv} \\
&= \lambda k.|[v/x]e_1|_{cbv}(\lambda x_1.|[v/x]e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.[||v||_{cbv}/x]|e_1|_{cbv}(\lambda x_1.[||v||_{cbv}/x]|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]\lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]|e_1 e_2|_{cbv}
\end{aligned}$$

CPS commutes with substitution

Lemma

$$||[v/x]v'||_{cbv} = [||v||_{cbv}/x]||v'||_{cbv}$$
$$|[v/x]e|_{cbv} = [||v||_{cbv}/x]|e|_{cbv}$$

Proof: simultaneous induction on the structure of v', e.

$v' = x'$

$x' = x$: $||[v/x]x||_{cbv} = ||v||_{cbv} = [||v||_{cbv}/x]||x||_{cbv}$
$x' \neq x$: $||[v/x]x'||_{cbv} = ||x'||_{cbv} = [||v||_{cbv}/x]||x'||_{cbv}$

$e = e_1 e_2$

$$\begin{aligned}
|[v/x](e_1 e_2)|_{cbv} &= |[v/x]e_1[v/x]e_2|_{cbv} \\
&= \lambda k.|[v/x]e_1|_{cbv}(\lambda x_1.|[v/x]e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.[||v||_{cbv}/x]|e_1|_{cbv}(\lambda x_1.[||v||_{cbv}/x]|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]\lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]|e_1 e_2|_{cbv}
\end{aligned}$$

CPS commutes with substitution

Lemma

$$||[v/x]v'||_{cbv} = [||v||_{cbv}/x]||v'||_{cbv}$$
$$|[v/x]e|_{cbv} = [||v||_{cbv}/x]|e|_{cbv}$$

Proof: simultaneous induction on the structure of v', e.

$v' = x'$

$x' = x$: $||[v/x]x||_{cbv} = ||v||_{cbv} = [||v||_{cbv}/x]||x||_{cbv}$

$x' \neq x$: $||[v/x]x'||_{cbv} = ||x'||_{cbv} = [||v||_{cbv}/x]||x'||_{cbv}$

$e = e_1 e_2$

Induction Hypothesis on e_1, e_2

$$\begin{aligned}
|[v/x](e_1 e_2)|_{cbv} &= |[v/x]e_1[v/x]e_2|_{cbv} \\
&= \lambda k.|[v/x]e_1|_{cbv}(\lambda x_1.|[v/x]e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.[||v||_{cbv}/x]|e_1|_{cbv}(\lambda x_1.[||v||_{cbv}/x]|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]\lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]|e_1 e_2|_{cbv}
\end{aligned}$$

CPS commutes with substitution

Lemma

$$||[v/x]v'||_{cbv} = [||v||_{cbv}/x]||v'||_{cbv}$$
$$|[v/x]e|_{cbv} = [||v||_{cbv}/x]|e|_{cbv}$$

Proof: simultaneous induction on the structure of v', e.

$v' = x'$

$x' = x$: $||[v/x]x||_{cbv} = ||v||_{cbv} = [||v||_{cbv}/x]||x||_{cbv}$
$x' \neq x$: $||[v/x]x'||_{cbv} = ||x'||_{cbv} = [||v||_{cbv}/x]||x'||_{cbv}$

$e = e_1 e_2$

$$\begin{aligned}
|[v/x](e_1 e_2)|_{cbv} &= |[v/x]e_1[v/x]e_2|_{cbv} \\
&= \lambda k.|[v/x]e_1|_{cbv}(\lambda x_1.|[v/x]e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.[||v||_{cbv}/x]|e_1|_{cbv}(\lambda x_1.[||v||_{cbv}/x]|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]\lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]|e_1 e_2|_{cbv}
\end{aligned}$$

CPS commutes with substitution

Lemma

$$||[v/x]v'||_{cbv} = [||v||_{cbv}/x]||v'||_{cbv}$$
$$|[v/x]e|_{cbv} = [||v||_{cbv}/x]|e|_{cbv}$$

Proof: simultaneous induction on the structure of v', e.

$v' = x'$

$x' = x$: $||[v/x]x||_{cbv} = ||v||_{cbv} = [||v||_{cbv}/x]||x||_{cbv}$
$x' \neq x$: $||[v/x]x'||_{cbv} = ||x'||_{cbv} = [||v||_{cbv}/x]||x'||_{cbv}$

$e = e_1 e_2$

$$\begin{aligned}
|[v/x](e_1 e_2)|_{cbv} &= |[v/x]e_1[v/x]e_2|_{cbv} \\
&= \lambda k.|[v/x]e_1|_{cbv}(\lambda x_1.|[v/x]e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= \lambda k.[||v||_{cbv}/x]|e_1|_{cbv}(\lambda x_1.[||v||_{cbv}/x]|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]\lambda k.|e_1|_{cbv}(\lambda x_1.|e_2|_{cbv}(\lambda x_2.x_1 x_2 k)) \\
&= [||v||_{cbv}/x]|e_1 e_2|_{cbv}
\end{aligned}$$

Overview

call-by-value (modified)

Define a different call-by-value transform (cbv'), where we require the **let**-bound to be a value.

So:

$$\textbf{let } x \textbf{ be } v \textbf{ in } e$$

instead of the previous

$$\textbf{let } x \textbf{ be } e_1 \textbf{ in } e_2$$

The new transform is given by

$$|\textbf{ let } x \textbf{ be } v \textbf{ in } e|_{cbv'} = \lambda k.\ \textbf{let } x \textbf{ be } ||v||_{cbv'} \textbf{ in } (|e|_{cbv'}\, k)$$

This is the only difference between cbv and cbv'

call-by-value (modified)

Define a different call-by-value transform (cbv'), where we require the **let**-bound to be a value.

So:

$$\textbf{let } x \textbf{ be } v \textbf{ in } e$$

instead of the previous

$$\textbf{let } x \textbf{ be } e_1 \textbf{ in } e_2$$

The new transform is given by

$$|\textbf{ let } x \textbf{ be } v \textbf{ in } e|_{cbv'} = \lambda k.\ \textbf{let } x \textbf{ be } ||v||_{cbv'} \textbf{ in } (|e|_{cbv'}\, k)$$

This is the only difference between cbv and cbv'

call-by-value (modified)

Define a different call-by-value transform (cbv'), where we require the **let**-bound to be a value.

So:

$$\textbf{let } x \textbf{ be } v \textbf{ in } e$$

instead of the previous

$$\textbf{let } x \textbf{ be } e_1 \textbf{ in } e_2$$

The new transform is given by

$$|\textbf{ let } x \textbf{ be } v \textbf{ in } e|_{cbv'} = \lambda k.\ \textbf{let } x \textbf{ be } ||v||_{cbv'} \textbf{ in } (|e|_{cbv'} k)$$

This is the only difference between cbv and cbv'

CPS commutes with substitution (cbv')

The previous lemma also holds for cbv':

Lemma

$$||[v/x]v'||_{cbv'} = [||v||_{cbv'}/x]||v'||_{cbv'}$$
$$|[v/x]e|_{cbv'} = [||v||_{cbv'}/x]|e|_{cbv}$$

Overview

1 Untyped Terms
- call-by-value
- call-by-value (modified)
- call-by-name

2 Simple Type Assignment

CPS tranforms (call-by-name)

Values in call by name:

$$n ::= \lambda x.e \mid \textbf{callcc} \mid \textbf{throw}$$

Definition

$$
\begin{aligned}
|n|_{cbn} &= \lambda k.k||n||_{cbn} \\
|x|_{cbn} &= x \\
|e_1 e_2|_{cbn} &= \lambda k.|e_1|_{cbn}(\lambda x_1.x_1|e_2|_{cbn}k) \\
|\ \textbf{let}\ x\ \textbf{be}\ e_1\ \textbf{in}\ e_2|_{cbn} &= \lambda k.\ \textbf{let}\ x\ \textbf{be}\ |e_1|_{cbn}\ \textbf{in}\ (|e_2|_{cbn}k)
\end{aligned}
$$

$$
\begin{aligned}
||\lambda x.e||_{cbn} &= \lambda x.|e|_{cbn} \\
||\ \textbf{callcc}\ ||_{cbn} &= \lambda f.\lambda k.f(\lambda f'.f'(\lambda l.lk)k) \\
||\ \textbf{throw}\ ||_{cbn} &= \lambda c.\lambda k.k(\lambda x.\lambda l.c(\lambda c'.x(\lambda x'.c'x')))
\end{aligned}
$$

- Note the difference with the cbv definitions of **callcc, throw**
  - $||\ \textbf{callcc}\ ||_{cbv} = \lambda f.\lambda k.fkk$
  - $||\ \textbf{throw}\ ||_{cbv} = \lambda c.\lambda k.k(\lambda x.\lambda l.cx)$

CPS tranforms (call-by-name)

Values in call by name:

$$n ::= \lambda x.e \mid \textbf{callcc} \mid \textbf{throw}$$

Definition

$$|n|_{cbn} = \lambda k.k||n||_{cbn}$$
$$|x|_{cbn} = x$$
$$|e_1 e_2|_{cbn} = \lambda k.|e_1|_{cbn}(\lambda x_1.x_1|e_2|_{cbn}k)$$
$$|\ \textbf{let}\ x\ \textbf{be}\ e_1\ \textbf{in}\ e_2|_{cbn} = \lambda k.\ \textbf{let}\ x\ \textbf{be}\ |e_1|_{cbn}\ \textbf{in}\ (|e_2|_{cbn}k)$$

$$||\lambda x.e||_{cbn} = \lambda x.|e|_{cbn}$$
$$||\ \textbf{callcc}\ ||_{cbn} = \lambda f.\lambda k.f(\lambda f'.f'(\lambda l.lk)k)$$
$$||\ \textbf{throw}\ ||_{cbn} = \lambda c.\lambda k.k(\lambda x.\lambda l.c(\lambda c'.x(\lambda x'.c'x')))$$

- Note the difference with the cbv definitions of **callcc, throw**
  - $||\ \textbf{callcc}\ ||_{cbv} = \lambda f.\lambda k.fkk$
  - $||\ \textbf{throw}\ ||_{cbv} = \lambda c.\lambda k.k(\lambda x.\lambda l.cx)$

CPS tranforms (call-by-name)

Values in call by name:

$$n ::= \lambda x.e \mid \textbf{callcc} \mid \textbf{throw}$$

Definition

$$
\begin{aligned}
|n|_{cbn} &= \lambda k.k||n||_{cbn} \\
|x|_{cbn} &= x \\
|e_1 e_2|_{cbn} &= \lambda k.|e_1|_{cbn}(\lambda x_1.x_1|e_2|_{cbn}k) \\
|\ \textbf{let}\ x\ \textbf{be}\ e_1\ \textbf{in}\ e_2|_{cbn} &= \lambda k.\ \textbf{let}\ x\ \textbf{be}\ |e_1|_{cbn}\ \textbf{in}\ (|e_2|_{cbn}k) \\
\\
||\lambda x.e||_{cbn} &= \lambda x.|e|_{cbn} \\
||\ \textbf{callcc}\ ||_{cbn} &= \lambda f.\lambda k.f(\lambda f'.f'(\lambda l.lk)k) \\
||\ \textbf{throw}\ ||_{cbn} &= \lambda c.\lambda k.k(\lambda x.\lambda l.c(\lambda c'.x(\lambda x'.c'x')))
\end{aligned}
$$

- Note the difference with the cbv definitions of **callcc, throw**
  - $||\ \textbf{callcc}\ ||_{cbv} = \lambda f.\lambda k.fkk$
  - $||\ \textbf{throw}\ ||_{cbv} = \lambda c.\lambda k.k(\lambda x.\lambda l.cx)$

CPS commutes with substitution (cbn)

Lemma

$$||[e/x]n||_{cbn} = [|e|_{cbn}/x]||n||_{cbn}$$
$$|[e/x]e'|_{cbn} = [|e|_{cbn}/x]|e'|_{cbn}$$

Plotkin and Griffin

Plotkin: relation between call-by-value/call-by-name CPS transform and operational semantics of call-by-value/call-by-name for pure λ-terms

Griffin: extended this relation to CPS primitives for the call-by-value case

Theorem (Plotkin,Griffin)

The closed expression e evaluates to v under call-by-value **iff** $|e|_{cbv}(\lambda x.x)$ evaluates to $||v||_{cbv}$ under either call-by-value or call-by-name

Plotkin and Griffin

Plotkin: relation between call-by-value/call-by-name CPS transform and operational semantics of call-by-value/call-by-name for pure λ-terms

Griffin: extended this relation to CPS primitives for the call-by-value case

Theorem (Plotkin,Griffin)

The closed expression e evaluates to v under call-by-value **iff** $|e|_{cbv}(\lambda x.x)$ evaluates to $||v||_{cbv}$ under either call-by-value or call-by-name

Plotkin and Griffin

Plotkin: relation between call-by-value/call-by-name CPS transform and operational semantics of call-by-value/call-by-name for pure λ-terms

Griffin: extended this relation to CPS primitives for the call-by-value case

Theorem (Plotkin,Griffin)

The closed expression e evaluates to v under call-by-value **iff** $|e|_{cbv}(\lambda x.x)$ evaluates to $||v||_{cbv}$ under either call-by-value or call-by-name

Overview

Simple Type Assignment

Definition: $\lambda^{\rightarrow}$ Types and Contexts

$$
\begin{aligned}
\text{types:}\quad & \tau ::= b \mid \tau_1 \rightarrow \tau_2 \\
\text{contexts:}\quad & \Gamma ::= \bullet \mid \Gamma, x : \tau
\end{aligned}
$$

Where

- b is a base type
- There is a α in b which represents the "answer" type of a CPS transform.

Typing Rules for $\lambda^{\rightarrow}$

$$\Gamma \vdash x : \Gamma(x) \quad \text{(VAR)}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \rightarrow \tau_2}(x \notin \mathrm{dom}(\Gamma)) \quad \text{(ABS)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau} \quad \text{(APP)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma, x : \tau_1 \vdash e_2 : \tau}{\Gamma \vdash \textbf{let } x \textbf{ be } e_1 \textbf{ in } e_2 : \tau} \quad \text{(MONO-LET)}$$

The type system $\lambda^{\rightarrow} +$ **cont** is defined by adding the type expression τ **cont** and the following typing rules for the continuation-passing primitives:

$$\Gamma \vdash \textbf{callcc} : (\tau \textbf{ cont} \rightarrow \tau) \rightarrow \tau \quad \text{(CALLCC)}$$

$$\Gamma \vdash \textbf{throw} : \tau \textbf{ cont} \rightarrow \tau \rightarrow \tau' \quad \text{(THROW)}$$

Typing Rules for $\lambda^{\rightarrow}$

$$\Gamma \vdash x : \Gamma(x) \quad \text{(VAR)}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \rightarrow \tau_2} (x \notin \mathrm{dom}(\Gamma)) \quad \text{(ABS)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau} \quad \text{(APP)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma, x : \tau_1 \vdash e_2 : \tau}{\Gamma \vdash \ \mathbf{let}\ x\ \mathbf{be}\ e_1\ \mathbf{in}\ e_2 : \tau} \quad \text{(MONO-LET)}$$

The type system $\lambda^{\rightarrow} +$ **cont** is defined by adding the type expression τ **cont** and the following typing rules for the continuation-passing primitives:

$$\Gamma \vdash \ \mathbf{callcc} : (\tau\ \mathbf{cont} \rightarrow \tau) \rightarrow \tau \quad \text{(CALLCC)} \qquad \Gamma \vdash \ \mathbf{throw} : \tau\ \mathbf{cont} \rightarrow \tau \rightarrow \tau' \quad \text{(THROW)}$$

Typing Rules for $\lambda^{\rightarrow}$

$$\Gamma \vdash x : \Gamma(x) \quad \text{(VAR)}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \rightarrow \tau_2} (x \notin \text{dom}(\Gamma)) \quad \text{(ABS)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau} \quad \text{(APP)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma, x : \tau_1 \vdash e_2 : \tau}{\Gamma \vdash \ \mathbf{let}\ x\ \mathbf{be}\ e_1\ \mathbf{in}\ e_2 : \tau} \quad \text{(MONO-LET)}$$

The type system $\lambda^{\rightarrow} +$ **cont** is defined by adding the type expression τ **cont** and the following typing rules for the continuation-passing primitives:

$$\Gamma \vdash \ \mathbf{callcc} : (\tau\ \mathbf{cont} \rightarrow \tau) \rightarrow \tau \quad \text{(CALLCC)} \qquad \Gamma \vdash \ \mathbf{throw} : \tau\ \mathbf{cont} \rightarrow \tau \rightarrow \tau' \quad \text{(THROW)}$$

Typing Rules for $\lambda^{\rightarrow}$

$$\Gamma \vdash x : \Gamma(x) \quad \text{(VAR)}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \rightarrow \tau_2} (x \notin \mathrm{dom}(\Gamma)) \quad \text{(ABS)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau} \quad \text{(APP)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma, x : \tau_1 \vdash e_2 : \tau}{\Gamma \vdash \ \mathbf{let}\ x\ \mathbf{be}\ e_1\ \mathbf{in}\ e_2 : \tau} \quad \text{(MONO-LET)}$$

The type system $\lambda^{\rightarrow} +$ **cont** is defined by adding the type expression τ **cont** and the following typing rules for the continuation-passing primitives:

$$\Gamma \vdash \ \mathbf{callcc} : (\tau\ \mathbf{cont} \rightarrow \tau) \rightarrow \tau \quad \text{(CALLCC)} \qquad \Gamma \vdash \ \mathbf{throw} : \tau\ \mathbf{cont} \rightarrow \tau \rightarrow \tau' \quad \text{(THROW)}$$

Typing Rules for $\lambda^{\rightarrow}$

$$\Gamma \vdash x : \Gamma(x) \quad \text{(VAR)}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \rightarrow \tau_2} (x \notin \mathrm{dom}(\Gamma)) \quad \text{(ABS)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau} \quad \text{(APP)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma, x : \tau_1 \vdash e_2 : \tau}{\Gamma \vdash \ \textbf{let}\ x\ \textbf{be}\ e_1\ \textbf{in}\ e_2 : \tau} \quad \text{(MONO-LET)}$$

The type system $\lambda^{\rightarrow} + \ \textbf{cont}$ is defined by adding the type expression $\tau\ \textbf{cont}$ and the following typing rules for the continuation-passing primitives:

$$\Gamma \vdash \ \textbf{callcc} \ : (\tau\ \textbf{cont} \rightarrow \tau) \rightarrow \tau \quad \text{(CALLCC)} \qquad \Gamma \vdash \ \textbf{throw} \ : \tau\ \textbf{cont} \rightarrow \tau \rightarrow \tau' \quad \text{(THROW)}$$

Type Transform for $\lambda^{\rightarrow}$ (call-by-value)

Definition

$$\begin{aligned}
|\tau|_{cbv} &= (||\tau||_{cbv} \rightarrow \alpha) \rightarrow \alpha \\
||b||_{cbv} &= b \\
||\tau_1 \rightarrow \tau_2||_{cbv} &= ||\tau_1||_{cbv} \rightarrow |\tau_2|_{cbv} \\
&= ||\tau_1||_{cbv} \rightarrow (||\tau_2||_{cbv} \rightarrow \alpha) \rightarrow \alpha
\end{aligned}$$

Extend to contexts by defining $||\Gamma||_{cbv}(x) = ||\Gamma(x)||_{cbv}$ for each $x \in \mathrm{dom}(\Gamma)$

Theorem

- If $\Gamma \vdash v : \tau$, then $||\Gamma||_{cbv} \vdash ||v||_{cbv} : ||\tau||_{cbv}$
- If $\Gamma \vdash e : \tau$, then $||\Gamma||_{cbv} \vdash |e|_{cbv} : |\tau|_{cbv}$

The call-by-value transform is extended to $\lambda^{\rightarrow} +$ **cont** by defining

$$||\tau\ \textbf{cont}\ ||_{cbv} = ||\tau||_{cbv} \rightarrow \alpha$$

Type Transform for $\lambda^{\rightarrow}$ (call-by-value)

Definition

$$\begin{aligned} |\tau|_{cbv} &= (||\tau||_{cbv} \rightarrow \alpha) \rightarrow \alpha \\ ||b||_{cbv} &= b \\ ||\tau_1 \rightarrow \tau_2||_{cbv} &= ||\tau_1||_{cbv} \rightarrow |\tau_2|_{cbv} \\ &= ||\tau_1||_{cbv} \rightarrow (||\tau_2||_{cbv} \rightarrow \alpha) \rightarrow \alpha \end{aligned}$$

Extend to contexts by defining $||\Gamma||_{cbv}(x) = ||\Gamma(x)||_{cbv}$ for each $x \in \mathrm{dom}(\Gamma)$

Theorem

- If $\Gamma \vdash v : \tau$, then $||\Gamma||_{cbv} \vdash ||v||_{cbv} : ||\tau||_{cbv}$
- If $\Gamma \vdash e : \tau$, then $||\Gamma||_{cbv} \vdash |e|_{cbv} : |\tau|_{cbv}$

The call-by-value transform is extended to $\lambda^{\rightarrow} + \textbf{cont}$ by defining

$$||\tau\ \textbf{cont}\ ||_{cbv} = ||\tau||_{cbv} \rightarrow \alpha$$

Type Transform for $\lambda^{\rightarrow}$ (call-by-value)

Definition

$$
\begin{aligned}
|\tau|_{cbv} &= (||\tau||_{cbv} \rightarrow \alpha) \rightarrow \alpha \\
||b||_{cbv} &= b \\
||\tau_1 \rightarrow \tau_2||_{cbv} &= ||\tau_1||_{cbv} \rightarrow |\tau_2|_{cbv} \\
&= ||\tau_1||_{cbv} \rightarrow (||\tau_2||_{cbv} \rightarrow \alpha) \rightarrow \alpha
\end{aligned}
$$

Extend to contexts by defining $||\Gamma||_{cbv}(x) = ||\Gamma(x)||_{cbv}$ for each $x \in \text{dom}(\Gamma)$

Theorem

- If $\Gamma \vdash v : \tau$, then $||\Gamma||_{cbv} \vdash ||v||_{cbv} : ||\tau||_{cbv}$
- If $\Gamma \vdash e : \tau$, then $||\Gamma||_{cbv} \vdash |e|_{cbv} : |\tau|_{cbv}$

The call-by-value transform is extended to $\lambda^{\rightarrow} + \textbf{cont}$ by defining

$$||\tau \ \textbf{cont}\ ||_{cbv} = ||\tau||_{cbv} \rightarrow \alpha$$

Type Transform for $\lambda^{\rightarrow}$ (call-by-name)

Definition

$$|\tau|_{cbn} = (||\tau||_{cbn} \rightarrow \alpha) \rightarrow \alpha$$

$$||b||_{cbn} = b$$
$$||\tau_1 \rightarrow \tau_2||_{cbn} = |\tau_1|_{cbn} \rightarrow |\tau_2|_{cbn}$$

Extend to contexts by defining $|\Gamma|_{cbn}(x) = |\Gamma(x)|_{cbn}$ for each $x \in \mathrm{dom}(\Gamma)$

Theorem

- If $\Gamma \vdash n : \tau$, then $|\Gamma|_{cbn} \vdash ||v||_{cbn} : ||\tau||_{cbn}$
- If $\Gamma \vdash e : \tau$, then $|\Gamma|_{cbn} \vdash |e|_{cbn} : |\tau|_{cbn}$

The call-by-name transform is extended to $\lambda^{\rightarrow} + \textbf{cont}$ by defining

$$||\tau \textbf{ cont }||_{cbn} = ||\tau||_{cbn} \rightarrow \alpha$$

Type Transform for $\lambda^{\rightarrow}$ (call-by-name)

Definition

$$|\tau|_{cbn} = (||\tau||_{cbn} \rightarrow \alpha) \rightarrow \alpha$$

$$\begin{aligned} ||b||_{cbn} &= b \\ ||\tau_1 \rightarrow \tau_2||_{cbn} &= |\tau_1|_{cbn} \rightarrow |\tau_2|_{cbn} \end{aligned}$$

Extend to contexts by defining $|\Gamma|_{cbn}(x) = |\Gamma(x)|_{cbn}$ for each $x \in \text{dom}(\Gamma)$

Theorem

- If $\Gamma \vdash n : \tau$, then $|\Gamma|_{cbn} \vdash ||v||_{cbn} : ||\tau||_{cbn}$
- If $\Gamma \vdash e : \tau$, then $|\Gamma|_{cbn} \vdash |e|_{cbn} : |\tau|_{cbn}$

The call-by-name transform is extended to $\lambda^{\rightarrow} +$ **cont** by defining

$$||\tau \textbf{ cont }||_{cbn} = ||\tau||_{cbn} \rightarrow \alpha$$

Type Transform for $\lambda^{\rightarrow}$ (call-by-name)

Definition

$$|\tau|_{cbn} = (||\tau||_{cbn} \rightarrow \alpha) \rightarrow \alpha$$

$$||b||_{cbn} = b$$
$$||\tau_1 \rightarrow \tau_2||_{cbn} = |\tau_1|_{cbn} \rightarrow |\tau_2|_{cbn}$$

Extend to contexts by defining $|\Gamma|_{cbn}(x) = |\Gamma(x)|_{cbn}$ for each $x \in \mathrm{dom}(\Gamma)$

Theorem

- If $\Gamma \vdash n : \tau$, then $|\Gamma|_{cbn} \vdash ||v||_{cbn} : ||\tau||_{cbn}$
- If $\Gamma \vdash e : \tau$, then $|\Gamma|_{cbn} \vdash |e|_{cbn} : |\tau|_{cbn}$

The call-by-name transform is extended to $\lambda^{\rightarrow} +$ **cont** by defining

$$||\tau \textbf{ cont }||_{cbn} = ||\tau||_{cbn} \rightarrow \alpha$$