Call-By-Push-Value in Coq: Operational, Equational, and Denotational Theory

Yannick Forster, Steven Schäfer, Simon Spies, Kathrin Stark

CPP 2019
January 14

SAARLAND UNIVERSITY
COMPUTER SCIENCE
SIC Saarland Informatics Campus

Imagine you conceived a new feature for programming languages...

You demonstrate its usefulness by extending the λ-calculus.

Things you want for your extended calculus:

- confluence
- (strong) normalisation
- abstract machine
- sound equational theory
- adequate denotational semantics

Imagine you conceived a new feature for programming languages. . .

You demonstrate its usefulness by extending the λ-calculus.

Things you want for your extended calculus:

- confluence
- (strong) normalisation
- abstract machine
- sound equational theory
- adequate denotational semantics

CBV or CBN?

Imagine you conceived a new feature for programming languages. . .

You demonstrate its usefulness by extending the λ-calculus.

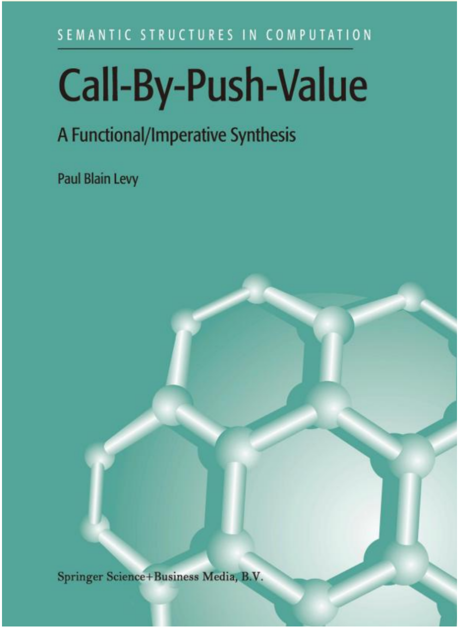
Things you want for your extended calculus:

- confluence
- (strong) normalisation
- abstract machine
- sound equational theory
- adequate denotational semantics

CBV or CBN?

"This duplication of work is tiresome. Furthermore, it makes the languages involved seem inherently arbitrary. We would prefer to study a single, canonical language." Paul B. Levy

CBPV for the rescue



CBPV for the rescue

- Idealised calculus for functional and imperative programming
- Well-suited for the inclusion of computational effects
- Subsuming paradigm for CBV and CBN

CBPV for the rescue

- Idealised calculus for functional and imperative programming
- Well-suited for the inclusion of computational effects
- Subsuming paradigm for CBV and CBN

"Subsuming"?!

CBPV for the rescue

- Idealised calculus for functional and imperative programming
- Well-suited for the inclusion of computational effects
- Subsuming paradigm for CBV and CBN

“Subsuming”?!

- CBV and CBN can be simulated in CBPV

CBPV for the rescue

- Idealised calculus for functional and imperative programming
- Well-suited for the inclusion of computational effects
- Subsuming paradigm for CBV and CBN

"Subsuming"?!

- CBV and CBN can be simulated in CBPV
- The translations preserve operational **and denotational** semantics

CBPV for the rescue

- Idealised calculus for functional and imperative programming
- Well-suited for the inclusion of computational effects
- Subsuming paradigm for CBV and CBN

"Subsuming"?!

- CBV and CBN can be simulated in CBPV
- The translations preserve operational **and denotational** semantics

"[The equational theory of CBPV] trivializes verifying many typical compiler optimizations." Rizkallah et al., ITP 2018

Contribution

Formalisation of

- standard operational semantics for CBPV
  - normalisation using logical relations
  - adequacy of set/algebra semantics
- unrestricted operational semantics for CBPV
  - confluence
  - strong normalisation using Kripke logical relations
  - soundness of equational theory
- translations of CBV/CBN into CBPV
  - preservation of operational semantics
  - confluence for full λ-calculus
  - strong normalisation for strong CBV/CBN
  - soundness of equational theories
  - adequate type-theoretic algebra semantics for CBV/CBN

8000 lines of Coq code, supported by Autosubst 2

Contribution

CBV

weak normalisation
strong normalisation
observational equivalence
denotational semantics

CBPV

confluence
weak normalisation
strong normalisation
observational equivalence
denotational semantics

CBN

confluence
weak normalisation
strong normalisation
observational equivalence
denotational semantics

Outline

- Definition of CBPV and translation of CBV/CBN
- Operational semantics
- Confluence
- Normalisation
- Simulation of CBV/CBN
- Equational Theory
- Denotational Theory

What's CBPV?

Syntax

$$
\begin{array}{lll}
\text{(value types)} & A & ::= 1 \mid \mathsf{U}\,C \mid A_1 \times A_2 \mid 0 \mid A_1 + A_2 \\
\text{(computation types)} & C & ::= \top \mid \mathsf{F}\,A \mid A \to C \mid C_1 \,\&\, C_2 \\
\text{(environments)} & \Gamma & ::= x_1 : A_1, \ldots, x_n : A_n
\end{array}
$$

Value typing $\boxed{\Gamma \vdash V : A}$

$$
\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A} \qquad
\frac{}{\Gamma \vdash () : 1} \qquad
\frac{\Gamma \vdash M : C}{\Gamma \vdash \{M\} : \mathsf{U}\,C} \qquad
\frac{\Gamma \vdash V_1 : A_1 \qquad \Gamma \vdash V_2 : A_2}{\Gamma \vdash (V_1, V_2) : A_1 \times A_2} \qquad
\frac{\Gamma \vdash V : A_i}{\Gamma \vdash \mathbf{inj}_i\, V : A_1 + A_2}
$$

Computation typing $\boxed{\Gamma \vdash M : C}$

$$
\frac{}{\Gamma \vdash \langle\rangle : \top} \qquad
\frac{\Gamma \vdash V : A}{\Gamma \vdash \mathbf{return}\ V : \mathsf{F}\,A} \qquad
\frac{\Gamma \vdash M : \mathsf{F}\,A \qquad \Gamma, x : A \vdash N : C}{\Gamma \vdash \mathbf{let}\ x \leftarrow M\ \mathbf{in}\ N : C} \qquad
\frac{\Gamma, x : A \vdash M : C}{\Gamma \vdash \lambda x.M : A \to C}
$$

$$
\frac{\Gamma \vdash M : A \to C \qquad \Gamma \vdash V : A}{\Gamma \vdash M\ V : C} \qquad
\frac{\Gamma \vdash V : \mathsf{U}\,C}{\Gamma \vdash V! : C} \qquad
\frac{\Gamma \vdash V : A_1 \times A_2 \qquad \Gamma, x_1 : A_1, x_2 : A_2 \vdash M : C}{\Gamma \vdash \mathbf{split}(V, x_1.x_2.M) : C}
$$

$$
\frac{\Gamma \vdash V : 0}{\Gamma \vdash \mathbf{case_0}(V) : C} \qquad
\frac{\Gamma \vdash V : A_1 + A_2 \qquad \Gamma, x_1 : A_1 \vdash M_1 : C \qquad \Gamma, x_2 : A_2 \vdash M_2 : C}{\Gamma \vdash \mathbf{case}(V, x_1.M_1, x_2.M_2) : C}
$$

$$
\frac{\Gamma \vdash M_1 : C_1 \qquad \Gamma \vdash M_2 : C_2}{\Gamma \vdash \langle M_1, M_2 \rangle : C_1 \,\&\, C_2} \qquad
\frac{\Gamma \vdash M : C_1 \,\&\, C_2}{\Gamma \vdash \mathbf{prj}_i\, M : C_i}
$$

What's CBPV?

Syntax

$$
\begin{array}{lll}
\text{(value types)} & A & ::= 1 \mid \mathsf{U}\,C \mid A_1 \times A_2 \mid 0 \mid A_1 + A_2 \\
\text{(computation types)} & C & ::= \top \mid \mathsf{F}\,A \mid A \to C \mid C_1 \,\&\, C_2 \\
\text{(environments)} & \Gamma & ::= x_1 : A_1, \ldots, x_n : A_n
\end{array}
$$

Value typing $\boxed{\Gamma \vdash V : A}$

$$
\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}
\qquad
\frac{}{\Gamma \vdash () : 1}
\qquad
\frac{\Gamma \vdash M : C}{\Gamma \vdash \{M\} : \mathsf{U}\,C}
\qquad
\frac{\Gamma \vdash V_1 : A_1 \qquad \Gamma \vdash V_2 : A_2}{\Gamma \vdash (V_1, V_2) : A_1 \times A_2}
\qquad
\frac{\Gamma \vdash V : A_i}{\Gamma \vdash \mathbf{inj}_i\ V : A_1 + A_2}
$$

Computation typing $\boxed{\Gamma \vdash M : C}$

$$
\frac{}{\Gamma \vdash \langle\rangle : \top}
\qquad
\frac{\Gamma \vdash V : A}{\Gamma \vdash \mathbf{return}\ \ V : \mathsf{F}\,A}
\qquad
\frac{\Gamma \vdash M : \mathsf{F}\,A \qquad \Gamma, x : A \vdash N : C}{\Gamma \vdash \mathbf{let}\ x \leftarrow M\ \mathbf{in}\ N : C}
\qquad
\frac{\Gamma, x : A \vdash M : C}{\Gamma \vdash \lambda x.M : A \to C}
$$

$$
\frac{\Gamma \vdash M : A \to C \qquad \Gamma \vdash V : A}{\Gamma \vdash M\ V : C}
\qquad
\frac{\Gamma \vdash V : \mathsf{U}\,C}{\Gamma \vdash V! : C}
\qquad
\frac{\Gamma \vdash V : A_1 \times A_2 \qquad \Gamma, x_1 : A_1, x_2 : A_2 \vdash M : C}{\Gamma \vdash \mathbf{split}(\ V, x_1.x_2.M\ ) : C}
$$

$$
\frac{\Gamma \vdash V : 0}{\Gamma \vdash \mathbf{case}_0\,(\ V\ ) : C}
\qquad
\frac{\Gamma \vdash V : A_1 + A_2 \qquad \Gamma, x_1 : A_1 \vdash M_1 : C \qquad \Gamma, x_2 : A_2 \vdash M_2 : C}{\Gamma \vdash \mathbf{case}(\ V, x_1.M_1, x_2.M_2\ ) : C}
$$

$$
\frac{\Gamma \vdash M_1 : C_1 \qquad \Gamma \vdash M_2 : C_2}{\Gamma \vdash \langle M_1, M_2\rangle : C_1 \,\&\, C_2}
\qquad
\frac{\Gamma \vdash M : C_1 \,\&\, C_2}{\Gamma \vdash \mathbf{prj}_i\ M : C_i}
$$

What's CBPV?

Syntax

$$
\begin{array}{llcl}
\text{(value types)} & A & ::= & 1 \mid \mathsf{U}\,C \mid A_1 \times A_2 \mid 0 \mid A_1 + A_2 \\
\text{(computation types)} & C & ::= & \top \mid \mathsf{F}\,A \mid A \to C \mid C_1 \,\&\, C_2 \\
\text{(environments)} & \Gamma & ::= & x_1 : A_1, \ldots, x_n : A_n
\end{array}
$$

Value typing $\boxed{\Gamma \vdash V : A}$

$$
\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}
\qquad
\frac{}{\Gamma \vdash () : 1}
\qquad
\frac{\Gamma \vdash M : C}{\Gamma \vdash \{M\} : \mathsf{U}\,C}
\qquad
\frac{\Gamma \vdash V_1 : A_1 \qquad \Gamma \vdash V_2 : A_2}{\Gamma \vdash (V_1, V_2) : A_1 \times A_2}
\qquad
\frac{\Gamma \vdash V : A_i}{\Gamma \vdash \mathbf{inj}_i\, V : A_1 + A_2}
$$

Computation typing $\boxed{\Gamma \vdash M : C}$

$$
\frac{}{\Gamma \vdash \langle\rangle : \top}
\qquad
\frac{\Gamma \vdash V : A}{\Gamma \vdash \mathbf{return}\;\, V : \mathsf{F}\,A}
\qquad
\frac{\Gamma \vdash M : \mathsf{F}\,A \qquad \Gamma, x : A \vdash N : C}{\Gamma \vdash \mathbf{let}\ x \leftarrow M\ \mathbf{in}\ N : C}
\qquad
\frac{\Gamma, x : A \vdash M : C}{\Gamma \vdash \lambda x.M : A \to C}
$$

$$
\frac{\Gamma \vdash M : A \to C \qquad \Gamma \vdash V : A}{\Gamma \vdash M\ V : C}
\qquad
\frac{\Gamma \vdash V : \mathsf{U}\,C}{\Gamma \vdash V! : C}
\qquad
\frac{\Gamma \vdash V : A_1 \times A_2 \qquad \Gamma, x_1 : A_1, x_2 : A_2 \vdash M : C}{\Gamma \vdash \mathbf{split}(V, x_1.x_2.M) : C}
$$

$$
\frac{\Gamma \vdash V : 0}{\Gamma \vdash \mathbf{case}_0(V) : C}
\qquad
\frac{\Gamma \vdash V : A_1 + A_2 \qquad \Gamma, x_1 : A_1 \vdash M_1 : C \qquad \Gamma, x_2 : A_2 \vdash M_2 : C}{\Gamma \vdash \mathbf{case}(V, x_1.M_1, x_2.M_2) : C}
$$

$$
\frac{\Gamma \vdash M_1 : C_1 \qquad \Gamma \vdash M_2 : C_2}{\Gamma \vdash \langle M_1, M_2\rangle : C_1 \,\&\, C_2}
\qquad
\frac{\Gamma \vdash M : C_1 \,\&\, C_2}{\Gamma \vdash \mathbf{prj}_i\, M : C_i}
$$

What's CBPV?

Syntax

$$
\begin{array}{lll}
\text{(value types)} & A & ::= 1 \mid \mathsf{U}\, C \mid A_1 \times A_2 \mid 0 \mid A_1 + A_2 \\
\text{(computation types)} & C & ::= \top \mid \mathsf{F}\, A \mid A \to C \mid C_1 \,\&\, C_2 \\
\text{(environments)} & \Gamma & ::= x_1 : A_1, \ldots, x_n : A_n
\end{array}
$$

Value typing $\boxed{\Gamma \vdash V : A}$

$$
\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}
\qquad
\frac{}{\Gamma \vdash () : 1}
\qquad
\frac{\Gamma \vdash M : C}{\Gamma \vdash \{M\} : \mathsf{U}\, C}
\qquad
\frac{\Gamma \vdash V_1 : A_1 \qquad \Gamma \vdash V_2 : A_2}{\Gamma \vdash (V_1, V_2) : A_1 \times A_2}
\qquad
\frac{\Gamma \vdash V : A_i}{\Gamma \vdash \mathbf{inj}_i\, V : A_1 + A_2}
$$

Computation typing $\boxed{\Gamma \vdash M : C}$

$$
\frac{}{\Gamma \vdash \langle\rangle : \top}
\qquad
\frac{\Gamma \vdash V : A}{\Gamma \vdash \mathbf{return}\ V : \mathsf{F}\, A}
\qquad
\frac{\Gamma \vdash M : \mathsf{F}\, A \qquad \Gamma, x : A \vdash N : C}{\Gamma \vdash \mathbf{let}\ x \leftarrow M\ \mathbf{in}\ N : C}
\qquad
\frac{\Gamma, x : A \vdash M : C}{\Gamma \vdash \lambda x.M : A \to C}
$$

$$
\frac{\Gamma \vdash M : A \to C \qquad \Gamma \vdash V : A}{\Gamma \vdash M\ V : C}
\qquad
\frac{\Gamma \vdash V : \mathsf{U}\, C}{\Gamma \vdash V! : C}
\qquad
\frac{\Gamma \vdash V : A_1 \times A_2 \qquad \Gamma, x_1 : A_1, x_2 : A_2 \vdash M : C}{\Gamma \vdash \mathbf{split}(V, x_1.x_2.M) : C}
$$

$$
\frac{\Gamma \vdash V : 0}{\Gamma \vdash \mathbf{case_0}(V) : C}
\qquad
\frac{\Gamma \vdash V : A_1 + A_2 \qquad \Gamma, x_1 : A_1 \vdash M_1 : C \qquad \Gamma, x_2 : A_2 \vdash M_2 : C}{\Gamma \vdash \mathbf{case}(V, x_1.M_1, x_2.M_2) : C}
$$

$$
\frac{\Gamma \vdash M_1 : C_1 \qquad \Gamma \vdash M_2 : C_2}{\Gamma \vdash \langle M_1, M_2 \rangle : C_1 \,\&\, C_2}
\qquad
\frac{\Gamma \vdash M : C_1 \,\&\, C_2}{\Gamma \vdash \mathbf{prj}_i\, M : C_i}
$$

Translations of CBN and CBV

CBN translation $\boxed{\overline{A} \text{ and } \overline{s}}$

$$\overline{1} := \mathsf{F}\,1 \qquad \overline{A \to B} := \mathsf{U}\,\overline{A} \to \overline{B}$$

$$\overline{x} := x! \qquad \overline{()} := \textbf{return}\ () \qquad \overline{\lambda x.s} := \lambda x.\overline{s} \qquad \overline{s\ t} := \overline{s}\,\{\,\overline{t}\,\}$$

Translations of CBN and CBV

CBN translation $\boxed{\overline{A} \text{ and } \overline{s}}$

$$\overline{1} := \mathsf{F}\,1 \qquad \overline{A \to B} := \mathsf{U}\,\overline{A} \to \overline{B}$$

$$\overline{x} := x! \qquad \overline{()} := \textbf{return }() \qquad \overline{\lambda x.s} := \lambda x.\overline{s} \qquad \overline{s\ t} := \overline{s}\,\{\,\overline{t}\,\}$$

fine-grained CBV translation $\boxed{\overline{A} \text{ and } \overline{v} \text{ and } \overline{s}}$

$$\overline{1} := 1 \qquad \overline{A \to B} := U\,(\overline{A} \to \mathsf{F}\,\overline{B})$$

$$\overline{x} := x \qquad \overline{()} := () \qquad \overline{\lambda x.s} := \{\lambda x.\overline{s}\} \qquad \overline{\textbf{val}\,v} := \textbf{return }\overline{v}$$

$$\overline{s\ t} := \textbf{let } x \Leftarrow \overline{s} \textbf{ in let } y \Leftarrow \overline{t} \textbf{ in } (x!)\ y$$

Operational Semantics

Primitive CBN reduction $\boxed{M \succ_n M'}$

$$(\lambda x.s)\ t \succ_n s[t/x]$$

Weak CBN reduction frames

$$\mathcal{C} := [\ \ ]\ s$$

Weak CBN reduction $\boxed{M \leadsto_n M'}$

$$\frac{M \succ_n M'}{M \leadsto_n M'}$$

$$\frac{M \leadsto_n M'}{\mathcal{C}[M] \leadsto_n \mathcal{C}[M']}$$

Operational Semantics

Primitive CBN reduction $\boxed{M \succ_n M'}$

$$(\lambda x.s)\ t \succ_n s[t/x]$$

Weak CBN reduction frames

$$\mathcal{C} := [\ \ ]\, s$$

Weak CBN reduction $\boxed{M \rightsquigarrow_n M'}$

$$\frac{M \succ_n M'}{M \rightsquigarrow_n M'}$$

$$\frac{M \rightsquigarrow_n M'}{\mathcal{C}[M] \rightsquigarrow_n \mathcal{C}[M']}$$

Primitive CBPV reduction $\boxed{M \succ M'}$

$$\begin{aligned} (\lambda x.M)\ V &\succ M[V/x] \\ \{M\}! &\succ M \\ \textbf{let } x \leftarrow \textbf{return } V \textbf{ in } M &\succ M[V/x] \end{aligned}$$

Weak CBPV reduction frames

$$\mathcal{C} := \textbf{let } x \leftarrow [\,] \textbf{ in } N \mid [\,]\ V$$

Weak CBPV reduction $\boxed{M \rightsquigarrow M'}$

$$\frac{M \succ M'}{M \rightsquigarrow M'}$$

$$\frac{M \rightsquigarrow M'}{\mathcal{C}[M] \rightsquigarrow \mathcal{C}[M']}$$

Unrestricted Operational Semantics

Primitive CBN reduction $\boxed{M \succ_n M'}$

$$(\lambda x.s)\ t \succ_n s[t/x]$$

Strong CBN reduction frames

$$\mathcal{C} \ := s\ [\,]\ |\ [\,]\ s\ |\ \lambda x.[\,]$$

Strong CBN reduction $\boxed{M \rightsquigarrow_n M'}$

$$\frac{M \succ_n M'}{M \rightsquigarrow_n M'}$$

$$\frac{M \rightsquigarrow_n M'}{\mathcal{C}[M] \rightsquigarrow_n \mathcal{C}[M']}$$

Unrestricted Operational Semantics

Primitive CBN reduction $\boxed{M \succ_n M'}$

$$(\lambda x.s)\ t \succ_n s[t/x]$$

Strong CBN reduction frames

$$\mathcal{C} \ := s\ [\,]\ |\ [\,]\ s\ |\ \lambda x.[\,]$$

Strong CBN reduction $\boxed{M \rightsquigarrow_n M'}$

$$\frac{M \succ_n M'}{M \rightsquigarrow_n M'}$$

$$\frac{M \rightsquigarrow_n M'}{\mathcal{C}[M] \rightsquigarrow_n \mathcal{C}[M']}$$

Strong CBPV reduction frames

$$\begin{array}{lll}
\text{(v/v contexts)} & \mathcal{C}_{v,v} & := [\ ] \\
\text{(v/c contexts)} & \mathcal{C}_{v,c} & := \{[\,]\} \\
\text{(c/c contexts)} & & \\
 & \mathcal{C}_{c,c} & := [\ ] \\
 & & \ \ |\ \ \textbf{let}\ x \leftarrow [\,]\ \textbf{in}\ N \\
 & & \ \ |\ \ \textbf{let}\ x \leftarrow M\ \textbf{in}\ [\,] \\
 & & \ \ |\ \ \lambda x.[\,]\ |\ [\,]\ V \\
\text{(c/v contexts)} & & \\
 & \mathcal{C}_{c,v} & := [\,]!\ |\ \textbf{return}\ [\,]\ |\ M\ [\,]
\end{array}$$

Strong CBPV reduction

$\boxed{M \rightsquigarrow M' \ \ and \ \ V \rightsquigarrow V'}$

$$\frac{M \succ M'}{M \rightsquigarrow M'} \qquad \frac{M \rightsquigarrow M'}{\mathcal{C}_{c,c}[M] \rightsquigarrow \mathcal{C}_{c,c}[M']}$$

$$\frac{V \rightsquigarrow V'}{\mathcal{C}_{c,v}[V] \rightsquigarrow \mathcal{C}_{c,v}[V']} \qquad \frac{M \rightsquigarrow M'}{\mathcal{C}_{v,c}[M] \rightsquigarrow \mathcal{C}_{v,c}[M']}$$

$$\frac{V \rightsquigarrow V'}{\mathcal{C}_{v,v}[V] \rightsquigarrow \mathcal{C}_{v,v}[V']}$$

Confluence

Standard approach following Tait / Martin-Löf / Takahashi:

- Define parallel reduction $\ggg$.
- $\leadsto \subseteq \ggg \subseteq \leadsto^*$
- Compatibility of $\ggg$ under renaming and substitution.
- Define maximal parallel reduction function ρ s.t.
  if $M \ggg N$ then $N \ggg \rho\ M$.

Theorem

Strong CBPV reduction is confluent.

Weak and strong normalisation

Value Semantic Typing $\boxed{V \in \mathcal{V}[A]}$

$$\mathcal{V}[0] := \emptyset \qquad \mathcal{V}[1] := \{()\}$$

$$\mathcal{V}[A_1 \times A_2] := \{(V_1, V_2) \mid V_1 \in \mathcal{V}[A_1], V_2 \in \mathcal{V}[A_2]\}$$

$$\mathcal{V}[A_1 + A_2] := \{\mathbf{inj}_i\ V \mid V \in \mathcal{V}[A_i]\}$$

$$\mathcal{V}[\mathsf{U}\ C] := \{\{M\} \mid M \in \mathcal{E}[C]\}$$

Computation Semantic Typing $\boxed{M \in \mathcal{C}[C]}$

$$\mathcal{C}[\top] := \{\langle\rangle\} \qquad \mathcal{C}[\mathsf{F}\ A] := \{\mathbf{return}\ \ V \mid V \in \mathcal{V}[A]\}$$

$$\mathcal{C}[A \to C] := \{\lambda x.M \mid \forall V \in \mathcal{V}[A].\ M[V/x] \in \mathcal{E}[C]\}$$

$$\mathcal{C}[C_1 \,\&\, C_2] := \{(M_1, M_2) \mid M_1 \in \mathcal{E}[C_1], M_2 \in \mathcal{E}[C_2]\}$$

Semantic Typing (Expression relation)

$$\mathcal{E}[C] := \{M \mid \exists N.\ M \Downarrow N \land N \in \mathcal{C}[C]\}$$

$$\mathcal{G}[\Gamma] := \{\gamma \mid \forall (x : A) \in \Gamma, \gamma x \in \mathcal{V}[A]\}$$

$$\Gamma \vDash V : A := \forall \gamma \in \mathcal{G}[\Gamma].\ V[\gamma] \in \mathcal{V}[A]$$

$$\Gamma \vDash M : C := \forall \gamma \in \mathcal{G}[\Gamma].\ M[\gamma] \in \mathcal{E}[C]$$

[Dreyer at al., 2016]

Weak and strong normalisation

Value Semantic Typing $\boxed{V \in \mathcal{V}[A]}$

$$\mathcal{V}[0] := \emptyset \qquad \mathcal{V}[1] := \{()\}$$

$$\mathcal{V}[A_1 \times A_2] := \{(V_1, V_2) \mid V_1 \in \mathcal{V}[A_1], V_2 \in \mathcal{V}[A_2]\}$$

$$\mathcal{V}[A_1 + A_2] := \{\textbf{inj}_i\ V \mid V \in \mathcal{V}[A_i]\}$$

$$\mathcal{V}[\mathsf{U}\ C] := \{\{M\} \mid M \in \mathcal{E}[C]\}$$

Computation Semantic Typing $\boxed{M \in \mathcal{C}[C]}$

$$\mathcal{C}[\top] := \{\langle\rangle\} \qquad \mathcal{C}[\mathsf{F}\ A] := \{\textbf{return}\ \ V \mid V \in \mathcal{V}[A]\}$$

$$\mathcal{C}[A \to C] := \{\lambda x.M \mid \forall V \in \mathcal{V}[A].\ M[V/x] \in \mathcal{E}[C]\}$$

$$\mathcal{C}[C_1 \,\&\, C_2] := \{(M_1, M_2) \mid M_1 \in \mathcal{E}[C_1], M_2 \in \mathcal{E}[C_2]\}$$

Semantic Typing (Expression relation)

$$\mathcal{E}[C] := \{M \mid \exists N.\ M \Downarrow N \wedge N \in \mathcal{C}[C]\}$$

$$\mathcal{G}[\Gamma] := \{\gamma \mid \forall (x : A) \in \Gamma, \gamma x \in \mathcal{V}[A]\}$$

$$\Gamma \vDash V : A := \forall \gamma \in \mathcal{G}[\Gamma].\ V[\gamma] \in \mathcal{V}[A]$$

$$\Gamma \vDash M : C := \forall \gamma \in \mathcal{G}[\Gamma].\ M[\gamma] \in \mathcal{E}[C]$$

Value Semantic Typing $\boxed{V \in \mathcal{V}[A], V \in \mathcal{V}^\circ[A]}$

$$\mathcal{V}[0] := \emptyset \qquad \mathcal{V}[1] := \{()\}$$

$$\mathcal{V}[A_1 \times A_2] := \left\{(V_1, V_2) \,\middle|\, V_1 \in \mathcal{V}^\circ[A_1], V_2 \in \mathcal{V}^\circ[A_2]\right\}$$

$$\mathcal{V}[A_1 + A_2] := \left\{\textbf{inj}_i\ V \,\middle|\, V \in \mathcal{V}^\circ[A_i]\right\} \qquad \mathcal{V}[\mathsf{U}\ C] := \{\{M\} \mid M \in \mathcal{E}[C]\}$$

$$\frac{}{x \in \mathcal{V}^\circ[A]} \qquad \frac{V \in \mathcal{V}[A]}{V \in \mathcal{V}^\circ[A]}$$

Computation Semantic Typing $\boxed{M \in \mathcal{C}[C]}$

$$\mathcal{C}[\top] := \{\langle\rangle\} \qquad \mathcal{C}[\mathsf{F}\ A] := \left\{\textbf{return}\ \ V \,\middle|\, V \in \mathcal{V}^\circ[A]\right\}$$

$$\mathcal{C}[A \to C] := \left\{\lambda x.M \,\middle|\, \forall \rho (V \in \mathcal{V}^\circ[A]).\ M[\rho[x := V]] \in \mathcal{E}[C]\right\}$$

$$\mathcal{C}[C_1 \,\&\, C_2] := \{(M_1, M_2) \mid M_1 \in \mathcal{E}[C_1], M_2 \in \mathcal{E}[C_2]\}$$

Semantic Typing (Expression relation)

$$\frac{\text{active } M \to M \in \mathcal{C}[C] \qquad \forall N.\ M \rightsquigarrow N \to N \in \mathcal{E}[C]}{M \in \mathcal{E}[C]} \qquad \mathcal{G}[\Gamma] := \left\{\gamma \,\middle|\, \forall (x : A) \in \Gamma, \gamma x \in \mathcal{V}^\circ[A]\right\}$$

$$\Gamma \vDash V : A := \forall \gamma \in \mathcal{G}[\Gamma].\ V[\gamma] \in \mathcal{V}^\circ[A] \qquad \Gamma \vDash M : C := \forall \gamma \in \mathcal{G}[\Gamma].\ M[\gamma] \in \mathcal{E}[C]$$

[Dreyer at al., 2016]

Simulation for CBN

Theorem

If $s \rightsquigarrow_n t$ *then* $\overline{s} \rightsquigarrow^+ \overline{t}$.

Theorem

If $\overline{s} \rightsquigarrow N$ *then* $N \rightsquigarrow^* \overline{t}$ *and* $s \rightsquigarrow_n^* t$ *for some* t.

Corollary

The full λ*-calculus (with sums and products) is confluent.*

Corollary

The simply-typed full λ*-calculus (with sums and products) is SN.*

Simulation for CBN

Theorem

If $s \rightsquigarrow_n t$ then $\overline{s} \rightsquigarrow^+ \overline{t}$.

Theorem

If $\overline{s} \rightsquigarrow N$ then $N \rightsquigarrow^* \overline{t}$ and $s \rightsquigarrow_n^* t$ for some t.

Corollary

The full λ-calculus (with sums and products) is confluent.

Corollary

The simply-typed full λ-calculus (with sums and products) is SN.

Untyped simulation can be reused for other type systems!

Equational and denotational theory

Observational equivalence

Definition (Ground types)

$$G \;::= 0 \mid 1 \mid G_1 \times G_2 \mid G_1 + G_2$$

Definition (Observational Equivalence)

$\Gamma \vdash M \simeq N : C$ for $\Gamma \vdash M, N : C$ if for all $\emptyset[\Gamma] \vdash \mathcal{X}_{c,c} : \mathsf{F}\, G[C]$ and V:

$$\mathcal{X}_{c,c}[M] \rightsquigarrow^* \textbf{return } V \quad \textit{iff} \quad \mathcal{X}_{c,c}[N] \rightsquigarrow^* \textbf{return } V$$

ITP 2018:

A Formal Equational Theory for Call-By-Push-Value

Christine Rizkallah[1(✉)], Dmitri Garbuzov[2], and Steve Zdancewic[2]

[1] University of New South Wales, Sydney, Australia
c.rizkallah@unsw.edu.au

[2] University of Pennsylvania, Philadelphia, USA
{dmitri,stevez}@cis.upenn.edu

Abstract. Establishing that two programs are contextually equivalent is hard, yet essential for reasoning about semantics preserving program transformations such as compiler optimizations. We adapt Lassen's normal form bisimulations technique to establish the soundness of equational theories for both an untyped call-by-value λ-calculus and a variant of Levy's call-by-push-value language. We demonstrate that our equational theory significantly simplifies the verification of optimizations.

"Our CBPV equational theory [. . .] *trivializes* verifying many typical compiler optimizations."

Equational theory

Rizkallah et al.: Equational theory with β-laws for CBPV with `letrec` is sound, using normal form bisimulation.

Levy proves βη-laws using denotational semantics, e.g

$$\Gamma \vdash V \simeq \{V!\} : \mathsf{U}\, C$$

or

$$\Gamma \vdash M \simeq \textbf{let } x \leftarrow M \textbf{ in return } x : \mathsf{F}\, A$$

We use logical equivalence, a straightforward extension of the logical relation used for normalisation.

Logical equivalence

Value Semantic Typing $\boxed{V \in \mathcal{V}[A]}$

$$\mathcal{V}[0] := \emptyset \qquad \mathcal{V}[1] := \{()\}$$

$$\mathcal{V}[A_1 \times A_2] := \{(V_1, V_2) \mid V_1 \in \mathcal{V}[A_1], V_2 \in \mathcal{V}[A_2]\}$$

$$\mathcal{V}[A_1 + A_2] := \{\mathbf{inj}_i\ V \mid V \in \mathcal{V}[A_i]\}$$

$$\mathcal{V}[\mathsf{U}\ C] := \{\{M\} \mid M \in \mathcal{E}[C]\}$$

Computation Semantic Typing $\boxed{M \in \mathcal{C}[C]}$

$$\mathcal{C}[\top] := \{\langle\rangle\} \qquad \mathcal{C}[\mathsf{F}\ A] := \{\mathbf{return}\ \ V \mid V \in \mathcal{V}[A]\}$$

$$\mathcal{C}[A \to C] := \{\lambda x.M \mid \forall V \in \mathcal{V}[A].\ M[V/x] \in \mathcal{E}[C]\}$$

$$\mathcal{C}[C_1\ \&\ C_2] := \{(M_1, M_2) \mid M_1 \in \mathcal{E}[C_1], M_2 \in \mathcal{E}[C_2]\}$$

Semantic Typing (Expression relation)

$$\mathcal{E}[C] := \{M \mid \exists N.\ M \Downarrow N \wedge N \in \mathcal{C}[C]\}$$

$$\mathcal{G}[\Gamma] := \{\gamma \mid \forall (x : A) \in \Gamma, \gamma x \in \mathcal{V}[A]\}$$

$$\Gamma \vDash V : A := \forall \gamma \in \mathcal{G}[\Gamma].\ V[\gamma] \in \mathcal{V}[A]$$

$$\Gamma \vDash M : C := \forall \gamma \in \mathcal{G}[\Gamma].\ M[\gamma] \in \mathcal{E}[C]$$

Logical equivalence

Value Semantic Typing $\boxed{V \in \mathcal{V}[A]}$

$$\mathcal{V}[0] := \emptyset \qquad \mathcal{V}[1] := \{()\}$$

$$\mathcal{V}[A_1 \times A_2] := \{(V_1, V_2) \mid V_1 \in \mathcal{V}[A_1], V_2 \in \mathcal{V}[A_2]\}$$

$$\mathcal{V}[A_1 + A_2] := \{\mathbf{inj}_i\ V \mid V \in \mathcal{V}[A_i]\}$$

$$\mathcal{V}[\mathsf{U}\ C] := \{\{M\} \mid M \in \mathcal{E}[C]\}$$

Computation Semantic Typing $\boxed{M \in \mathcal{C}[C]}$

$$\mathcal{C}[\top] := \{\langle\rangle\} \qquad \mathcal{C}[\mathsf{F}\ A] := \{\mathbf{return}\ \ V \mid V \in \mathcal{V}[A]\}$$

$$\mathcal{C}[A \to C] := \{\lambda x.M \mid \forall V \in \mathcal{V}[A].\ M[V/x] \in \mathcal{E}[C]\}$$

$$\mathcal{C}[C_1 \,\&\, C_2] := \{(M_1, M_2) \mid M_1 \in \mathcal{E}[C_1], M_2 \in \mathcal{E}[C_2]\}$$

Semantic Typing (Expression relation)

$$\mathcal{E}[C] := \{M \mid \exists N.\ M \Downarrow N \wedge N \in \mathcal{C}[C]\}$$

$$\mathcal{G}[\Gamma] := \{\gamma \mid \forall (x : A) \in \Gamma, \gamma x \in \mathcal{V}[A]\}$$

$$\Gamma \vDash V : A := \forall \gamma \in \mathcal{G}[\Gamma].\ V[\gamma] \in \mathcal{V}[A]$$

$$\Gamma \vDash M : C := \forall \gamma \in \mathcal{G}[\Gamma].\ M[\gamma] \in \mathcal{E}[C]$$

Value Relation $\boxed{(V, W) \in \mathcal{V}[A]}$

$$\mathcal{V}[0] := \emptyset \qquad \mathcal{V}[1] := \{((), ())\}$$

$$\mathcal{V}[A_1 \times A_2] := \{((V_1, V_2), (W_1, W_2)) \mid (V_1, W_1) \in \mathcal{V}[A_1], (V_2, W_2) \in \mathcal{V}[A_2]\}$$

$$\mathcal{V}[A_1 + A_2] := \{(\mathbf{inj}_i\ V, \mathbf{inj}_i\ W) \mid (V, W) \in \mathcal{V}[A_i]\}$$

$$\mathcal{V}[\mathsf{U}\ C] := \{(\{M\}, \{N\}) \mid (M, N) \in \mathcal{E}[C]\}$$

Computation Relation $\boxed{(M, N) \in \mathcal{C}[C]}$

$$\mathcal{C}[\top] := \{(\langle\rangle, \langle\rangle)\} \qquad \mathcal{C}[\mathsf{F}\ A] := \{(\mathbf{return}\ \ V, \mathbf{return}\ \ W) \mid (V, W) \in \mathcal{V}[A]\}$$

$$\mathcal{C}[A \to C] := \{(\lambda x.M, \lambda y.N) \mid \forall (V, W) \in \mathcal{V}[A].\ (M[V/x], N[W/y]) \in \mathcal{E}[C]\}$$

$$\mathcal{C}[C_1 \,\&\, C_2] := \{(\langle M_1, M_2\rangle, \langle N_1, N_2\rangle) \mid (M_1, N_1) \in \mathcal{E}[C_1], (M_2, N_2) \in \mathcal{E}[C_2]\}$$

Logical Equivalence

$$\mathcal{E}[C] := \left\{(M, N) \,\middle|\, \exists M'N'.\ M \Downarrow M' \text{ and } N \Downarrow N' \text{ and } (M', N') \in \mathcal{C}[C]\right\}$$

$$\mathcal{G}[\Gamma] := \{(\gamma_1, \gamma_2) \mid \forall (x : A) \in \Gamma.\ (\gamma_1 x, \gamma_2 x) \in \mathcal{V}[A]\}$$

$$\Gamma \vDash V \sim W : A := \forall (\gamma_1, \gamma_2) \in \mathcal{G}[\Gamma].\ (V[\gamma_1], W[\gamma_2]) \in \mathcal{V}[A]$$

$$\Gamma \vDash M \sim N : C := \forall (\gamma_1, \gamma_2) \in \mathcal{G}[\Gamma].\ (M[\gamma_1], N[\gamma_2]) \in \mathcal{E}[C]$$

Soundness of equational theory

$$\Gamma \vDash M \sim N : C \text{ implies } \Gamma \vdash M \simeq N : C$$

Soundness of equational theory

$$\Gamma \vDash M \sim N : C \text{ implies } \Gamma \vdash M \simeq N : C$$

$$\Gamma \vDash V \sim \{V!\} : \mathsf{U}\, C$$

and

$$\Gamma \vDash M \sim \textbf{let } x \leftarrow M \textbf{ in return } x : \mathsf{F}\, A$$

are easy to prove

ICFP 2017:

On the Expressive Power of User-Defined Effects: Effect Handlers, Monadic Reflection, Delimited Control

YANNICK FORSTER, Saarland University, Germany and University of Cambridge, England
OHAD KAMMAR, University of Oxford, England and University of Cambridge, England
SAM LINDLEY, University of Edinburgh, Scotland
MATIJA PRETNAR, University of Ljubljana, Slovenia

We compare the expressive power of three programming abstractions for user-defined computational effects: Plotkin and Pretnar's effect handlers, Filinski's monadic reflection, and delimited control without answer-type-modification. This comparison allows a precise discussion about the relative expressiveness of each programming abstraction. It also demonstrates the sensitivity of the relative expressiveness of user-defined effects to seemingly orthogonal language features.

We present three calculi, one per abstraction, extending Levy's call-by-push-value. For each calculus, we present syntax, operational semantics, a natural type-and-effect system, and, for effect handlers and monadic reflection, a set-theoretic denotational semantics. We establish their basic metatheoretic properties: safety, termination, and, where applicable, soundness and adequacy. Using Felleisen's notion of a macro translation, we show that these abstractions can macro-express each other, and show which translations preserve typeability. We use the adequate finitary set-theoretic denotational semantics for the monadic calculus to show that effect handlers cannot be macro-expressed while preserving typeability either by monadic reflection or by delimited control. Our argument fails with simple changes to the type system such as polymorphism and inductive types. We supplement our development with a mechanised Abella formalisation.

Denotational theory

We give a type-theoretic denotational semantics for CBPV. Values types have Coq-types as denotations, computation types have T-algebras for a monad T.

Type	Denotation
1	$\mathtt{unit}$
$A \times B$	$[\![A]\!] \times [\![B]\!]$
$\mathsf{U}\ C$	carrier of $[\![C]\!]$
$\top$	singleton algebra
$C \mathbin{\&} D$	product algebra
$A \to C$	exponential algebra
$\mathsf{F}\ A$	free algebra on $[\![A]\!]$

Theorem

If $\Gamma \vdash M, N : C$ *and* $[\![M]\!] = [\![N]\!]$ *then* $\Gamma \vdash M \simeq N : C$.

Formalisation

Contents	Spec	Proofs
Setup	350	250
Translating CBV and CBN to CBPV	1250	500
Weak Reduction	200	450
Weak Normalisation	100	150
Strong Reduction	800	950
Strong Normalisation	200	300
Observational Equivalence	250	350
Equational Theory	100	200
Denotational Semantics	700	600
Total	3,950	3,750

Low overhead compared to detailed paper proofs

Formalisation

Contents	Spec	Proofs
Setup	350	250
Translating CBV and CBN to CBPV	1250	500
Weak Reduction	200	450
Weak Normalisation	100	150
Strong Reduction	800	950
Strong Normalisation	200	300
Observational Equivalence	250	350
Equational Theory	100	200
Denotational Semantics	700	600
Total	3,950	3,750

Low overhead compared to detailed paper proofs

- Autosubst 2

Formalisation

Contents	Spec	Proofs
Setup	350	250
Translating CBV and CBN to CBPV	1250	500
Weak Reduction	200	450
Weak Normalisation	100	150
Strong Reduction	800	950
Strong Normalisation	200	300
Observational Equivalence	250	350
Equational Theory	100	200
Denotational Semantics	700	600
Total	3,950	3,750

Low overhead compared to detailed paper proofs

- Autosubst 2
- custom tactics

Formalisation

Contents	Spec	Proofs
Setup	350	250
Translating CBV and CBN to CBPV	1250	500
Weak Reduction	200	450
Weak Normalisation	100	150
Strong Reduction	800	950
Strong Normalisation	200	300
Observational Equivalence	250	350
Equational Theory	100	200
Denotational Semantics	700	600
Total	3,950	3,750

Low overhead compared to detailed paper proofs

- Autosubst 2
- custom tactics
- setoid rewriting

Autosubst 2

```
valtype : Type
comptype : Type
value : Type
comp : Type
bool : Type

one: valtype
U: comptype -> valtype

F: valtype -> comptype
arrow: valtype -> comptype -> comptype

u: value
thunk: comp -> value

force: value -> comp
lambda: (value -> comp) -> comp
app: comp -> value -> comp
ret: value -> comp
letin: comp -> (value -> comp) -> comp
```

- Autosubst generates well-scoped de Bruijn syntax, lemmas concerning substitutions and provides `asimpl` tactic simplifying substitutions expressions. . .
- . . . and supports mutually inductive syntax.

Autosubst 2

```
valtype : Type
comptype : Type
value : Type
comp : Type
bool : Type

one: valtype
U: comptype -> valtype

F: valtype -> comptype
arrow: valtype -> comptype -> comptype

u: value
thunk: comp -> value

force: value -> comp
lambda: (value -> comp) -> comp
app: comp -> value -> comp
ret: value -> comp
letin: comp -> (value -> comp) -> comp
```

- Autosubst generates well-scoped de Bruijn syntax, lemmas concerning substitutions and provides `asimpl` tactic simplifying substitutions expressions. . .
- . . . and supports mutually inductive syntax.
- Advertisement: Tomorrow, 11:00, Autosubst 2: Reasoning with Multi-Sorted de Bruijn Terms and Vector Substitutions

Autosubst 2

```
valtype : Type
comptype : Type
value : Type
comp : Type
bool : Type

one: valtype
U: comptype -> valtype

F: valtype -> comptype
arrow: valtype -> comptype -> comptype

u: value
thunk: comp -> value

force: value -> comp
lambda: (value -> comp) -> comp
app: comp -> value -> comp
ret: value -> comp
letin: comp -> (value -> comp) -> comp
```

Wrap-up

Main novel results

- First confluent strong operational semantics
- Translations from CBV/CBN untyped and small-step
- General proof method for SN

Future Work

- Include effects natively and formalise CBPV + algebraic effects
- Investigate logical equivalence for soundness proofs
- Extend CBPV with polymorphism

Wrap-up

Main novel results

- First confluent strong operational semantics
- Translations from CBV/CBN untyped and small-step
- General proof method for SN

Future Work

- Include effects natively and formalise CBPV + algebraic effects
- Investigate logical equivalence for soundness proofs
- Extend CBPV with polymorphism

Take home messages

- CBPV is a canonical base language for PL theory
- Doing (CBPV) semantics *in Coq* is beneficial
- Autosubst 2 will make your life easier (tomorrow, 11:00!)

`https://ps.uni-saarland.de/extras/cbpv-in-coq`

Related work

- Levy, Paul B. "Call-by-push-value: A subsuming paradigm." International Conference on Typed Lambda Calculi and Applications. Springer, Berlin, Heidelberg, 1999.
- Rizkallah, C., Garbuzov, D., Zdancewic, S. (2018, July). A Formal Equational Theory for Call-By-Push-Value. In International Conference on Interactive Theorem Proving (pp. 523-541). Springer, Cham.
- Forster, Y., Kammar, O., Lindley, S., Pretnar, M. (2017). On the expressive power of user-defined effects: effect handlers, monadic reflection, delimited control. Proceedings of the ACM on Programming Languages, 1(ICFP), 13.
- Doczkal, Christian, and Schwinghammer, Jan. "A Proof of Strong Normalization for Call-by-push-value." Unpublished note.