Linear Types for Continuations

Alex Simpson

LFCS, School of Informatics
University of Edinburgh, UK

Joint work with: Jeff Egger (LFCS, Edinburgh)
Rasmus Møgelberg (ITU, Copenhagen)
Part 1

Some CPS Translations
Call-by-value CPS translation

Type translation:

$$(\sigma \to \tau)^{Rv} = \sigma^{Rv} \to (\tau^{Rv} \to R) \to R$$

$$(\sigma \times \tau)^{Rv} = \sigma^{Rv} \times \tau^{Rv}$$

Term translation:

$$\Gamma \vdash t : \tau \implies \Gamma^{Rv} \vdash t^{Rv} : (\tau^{Rv} \to R) \to R$$

Faithful but not full (can also model control operators)

(Kleisli category of ((\_ \to R) \to R) \to R monad)
Call-by-name CPS translation

Type translation:

\[(\sigma \to \tau)^{Rn} = (\sigma^{Rn} \to R) \times \tau^{Rn}\]

\[(\sigma \times \tau)^{Rn} = \sigma^{Rn} + \tau^{Rn}\]

Term translation:

\[\Gamma \vdash t : \tau \implies \Gamma^{Rn} \to R \vdash t^{Rn} : \tau^{Rn} \to R\]

Faithful but not full (can also model control operators)

(Opposite of Kleisli category of \(((\cdot) \to R) \to R\) monad)
Cbv linearly-used CPS translation

Type translation:

$$(\sigma \to \tau)^{Rv} = \sigma^{Rv} \to (\tau^{Rv} \to R) \circ R$$

$$(\sigma \times \tau)^{Rv} = \sigma^{Rv} \times \tau^{Rv}$$

Term translation:

$$\Gamma \vdash t : \tau \implies \Gamma^{Rv} \vdash t^{Rv} : (\tau^{Rv} \to R) \circ R$$

Full and faithful [Hasegawa 2002]

(Kleisli category of $((-) \to R) \circ R$ monad on intuitionistic category)
Cbn linearly-used CPS translation

Type translation:

$$(\sigma \to \tau)^{\text{Rn}} = !\left(\sigma^{\text{Rn}} \to R\right) \otimes \tau^{\text{Rn}}$$

$$(\sigma \times \tau)^{\text{Rn}} = \sigma^{\text{Rn}} \oplus \tau^{\text{Rn}}$$

Term translation:

$$\Gamma \vdash t : \tau \implies \Gamma^{\text{Rn}} \to R \vdash t^{\text{Rn}} : \tau^{\text{Rn}} \to R$$

Full and faithful [Hasegawa 2004]

(Opposite of Kleisli category of $((-) \to R) \to R$ monad on linear category)
Part 2

The Enriched Effect Calculus
The Enriched Effect Calculus (EEC) [EMS 2009]
A more general target language than ILL

Based on distinction between values and computations.

— A value is a pervasive static object, it just is. Values can be copied and discarded. There is no natural notion of linear function between general value types.

— A computation has a dynamic side: it can be computed/invoked/evaluated/executed. There is a natural intuition of linear functions between computation types: the argument computation is executed exactly once.

— Linear functions are generalised evaluation contexts/stacks.
EEC type system

Value types:

\[ A, B, \ldots ::= \alpha | \alpha | 1 | A \times B | A \to B | !A \]
\[ | A \multimap B | !A \otimes B | 0 | A \oplus B \]

Computation types:

\[ A, B, \ldots ::= \alpha | 1 | A \times B | A \to B | !A \]
\[ | !A \otimes B | 0 | A \oplus B \].

Typing judgements:

\[ \Gamma \mid - \vdash t : A \]
\[ \Gamma \mid z : A \vdash t : B \]

The linearly-used CPS translations land in the EEC fragment of ILL.
Example rules

Rules are inherited from ILL. e.g.,

\[\Gamma, x : A \vdash t : B\]
\[\Gamma \vdash \lambda x : A. t : A \rightarrow B\]
\[\Gamma \vdash \lambda z : A. t : A \rightarrow \circ B\]
\[\Gamma \vdash \lambda x : A. t : A \rightarrow B\]
\[\Gamma \vdash s : A \rightarrow B\]
\[\Gamma \vdash t : A\]
\[\Gamma \vdash !t : \circ A\]
\[\Gamma | - \vdash t : !A\]
\[\Gamma, x : A | - \vdash u : B\]
\[\Gamma \vdash \text{let } !x \leftarrow t \text{ in } u : B\]
\[\Gamma \vdash s : A \rightarrow \circ B\]
\[\Gamma \vdash t : A\]

We have the Girard isomorphism:

\[A \rightarrow B \cong \circ !A \rightarrow B\]
Other isomorphisms of EEC

\((!A \otimes B) \rightarrow C \cong A \rightarrow (B \rightarrow C)\)

\(\cong B \rightarrow (A \rightarrow C)\)  

(As value types)

\(!A \otimes !B \cong !!(A \times B)\)

\(!1 \otimes A \cong !A\)

\(!!(A \times B) \otimes C \cong !!A \otimes !!B \otimes C\)

\(!A \otimes 0 \cong !0\)

\(!A \otimes (B \oplus C) \cong !!(A \otimes B) \oplus (!A \otimes C)\)  

(As computation types)

These isomorphisms are all familiar from ILL
Differences from linear logic

1. Distinction between computation types and value types.

2. \(A \rightarrow B\) is not assumed to be a computation type itself. Thus

\[
\begin{align*}
(A \rightarrow B) \rightarrow C & \quad \text{value type} \\
(A \rightarrow B) \rightarrow C & \quad \text{computation (hence value) type} \\
(A \rightarrow B) \rightarrow C & \quad \text{not available} \\
A \rightarrow B \rightarrow C & \quad \text{not available}
\end{align*}
\]

3. \(!A \otimes B\) is the application of a single primitive type-constructor \(!(-) \otimes (-)\) to \(A\) and \(B\). (There is no symmetric \(\otimes\).)

4. Commutativity does not hold, i.e.,

\[
\text{let } !x \leftarrow s \text{ in let } !y \leftarrow t \text{ in } u \neq \text{ let } !y \leftarrow t \text{ in let } !x \leftarrow s \text{ in } u
\]
Models of EEC

A model of the enriched effect calculus is given by [EMS 2009]:

— categories $\mathcal{V}$ (value types) and $\mathcal{C}$ (computation types)

— $\mathcal{V}$ is cartesian closed (models $1, A \times B, A \to B$)

— $\mathcal{C}$ is $\mathcal{V}$-enriched (models $A \multimap B$)

— $\mathcal{C}$ has $\mathcal{V}$-powers (models $A \rightarrow B$) and $\mathcal{V}$-copowers (models $!A \otimes B$)

— $\mathcal{C}$ has finite $\mathcal{V}$-enriched products (models $1, A \times B$) and coproducts (models $0, A \oplus B$)

— a $\mathcal{V}$-enriched adjunction $F \dashv U : \mathcal{C} \to \mathcal{V}$ (models $!A$)

We write the entire model as $F \dashv U : \mathcal{C} \to \mathcal{V}$.

(All structure other than the adjunction is determined by universal properties.)
Models of ILL are models of EEC

A linear/nonlinear model [Benton 1995] is given by symmetric monoidal closed category $\mathcal{C}$ (the linear category), a cartesian closed category $\mathcal{V}$ (the intuitionistic category) and a monoidal adjunction $F \dashv U : \mathcal{C} \rightarrow \mathcal{V}$. If $\mathcal{C}$ also has finite products and coproducts then the model is said to have additives.

Proposition [EMS 2009] If $F \dashv U : \mathcal{C} \rightarrow \mathcal{V}$ is a linear/nonlinear model with additives then it is a model of the enriched effect calculus.

N.B. The linear-logic syntax of EEC agrees with the interpretation of ILL in a linear/nonlinear model.

EEC is a fragment of ILL interpretable in a wider class of models.
Models of computational effects expand to models of EEC

Call-by-push-value (CBPV) [Levy 1999,2004] is a type theory for computational effects extending Moggi’s computational metalanguage [Moggi 1991] to include call-by-name.

Adjunction models [Levy 2005] are the natural models of call-by-push-value (CBPV), generalising Moggi’s strong monads.

Every model of EEC is an adjunction model of CBPV

Theorem [EMS 2009] Every adjunction model of CBPV fully embeds in a model of the enriched effect calculus.

EEC is a conservative extension of CBPV (hence of Moggi’s computational metalanguage) with a gain in expressivity and (essentially) no loss in range of applicability
EEC as a metalanguage for effects

Understand $!A$ as Moggi’s type $TA$ of computations (with effects) that produce results of type $A$.

The call-by-value (cbv) translation [Moggi] translates a simple type $\sigma$ to a value type $\sigma^v$.

The call-by-name (cbn) translation (cf. [Filinski, Levy]) translates $\sigma$ to a computation type $\sigma^n$.

\[
\begin{align*}
(\sigma \times \tau)^v &= \sigma^v \times \tau^v & \quad (\sigma \times \tau)^n &= \sigma^n \times \tau^n \\
(\sigma \to \tau)^v &= \sigma^v \to \!\tau^v & \quad (\sigma \to \tau)^n &= \sigma^n \to \tau^n
\end{align*}
\]

Typing judgements $\Gamma \vdash t: \sigma$ get translated to:

\[
\begin{align*}
\Gamma^v | - \vdash t^v : !\sigma^v & \quad \Gamma^n | - \vdash t^n : \sigma^n
\end{align*}
\]

The new linear connectives are not used!
Part 3

Linearly-used CPS in EEC
Linearly-used CPS translation of EEC to itself [EMS 2010]

Let $\mathbf{R}$ be a chosen computation type.

A value type $\mathbf{A}$ translates to a value type $\mathbf{A}^\mathbf{R}$.

A computation type $\mathbf{A}$ translates to a computation type $\mathbf{A}^\mathbf{CR}$.

These translations satisfy $\mathbf{A}^\mathbf{R} \simeq \mathbf{A}^\mathbf{CR} \rightarrow \mathbf{R}$.

A typing judgement $\Gamma \vdash t : \mathbf{A}$ translates to

$$\Gamma^\mathbf{R} \vdash t^\mathbf{R} : \mathbf{A}^\mathbf{R}$$

A typing judgement $\Gamma \vdash z : \mathbf{A} \vdash u : \mathbf{B}$ translates to

$$\Gamma^\mathbf{R} \vdash z : \mathbf{B}^\mathbf{CR} \vdash u^\mathbf{CR} : \mathbf{A}^\mathbf{CR}$$
\[
\begin{align*}
\alpha^\nu_R &= \alpha \\
\alpha^\nu_R &= \alpha^\text{CR} \to \text{R} \\
1^\nu_R &= 1 \\
(A \times B)^\nu_R &= A^\nu_R \times B^\nu_R \\
(A \to B)^\nu_R &= A^\nu_R \to B^\nu_R \\
(!A)^\nu_R &= (!A)^\text{CR} \to \text{R} \\
(A \to B)^\nu_R &= B^\text{CR} \to A^\text{CR} \\
(!A \otimes B)^\nu_R &= (!A \otimes B)^\text{CR} \to \text{R} \\
(0)^\nu_R &= (0)^\text{CR} \to \text{R} \\
(A \oplus B)^\nu_R &= (A \oplus B)^\text{CR} \to \text{R} \\
R^\nu_R &= \text{R} \\
\alpha^\text{CR} &= \alpha \\
1^\text{CR} &= 0 \\
(A \times B)^\text{CR} &= A^\text{CR} \oplus B^\text{CR} \\
(A \to B)^\text{CR} &= !A^\nu_R \otimes B^\text{CR} \\
(!A)^\text{CR} &= A^\nu_R \to \text{R} \\
(!A \otimes B)^\text{CR} &= A^\nu_R \to B^\text{CR} \\
(0)^\text{CR} &= 1 \\
(A \oplus B)^\text{CR} &= A^\text{CR} \times B^\text{CR} \\
R^\text{CR} &= !1
\end{align*}
\]
Recovering cbv linearly-used CPS translation

\[
(\sigma \to \tau)^{Rv} = \sigma^{Rv} \to (\tau^{Rv} \to R) \to R
\]

\[
(\sigma \times \tau)^{Rv} = \sigma^{Rv} \times \tau^{Rv}
\]

\[
\Gamma \vdash t : \tau \implies \Gamma^{Rv} \mid - \vdash t^{Rv} : (\tau^{Rv} \to R) \to R
\]

Recovered as:

\[
\sigma^{Rv} = (\sigma^v)^{\nu R}
\]

\[
t^{Rv} = (t^v)^{\nu R}
\]
Recovering cbn linearly-used CPS translation

\[
(\sigma \rightarrow \tau)^{\text{Rn}} = ! (\sigma^{\text{Rn}} \rightarrow \mathcal{R}) \otimes \tau^{\text{Rn}}
\]

\[
(\sigma \times \tau)^{\text{Rn}} = \sigma^{\text{Rn}} \oplus \tau^{\text{Rn}}
\]

\[
\Gamma \vdash t : \tau \quad \Rightarrow \quad \Gamma^{\text{Rn}} \rightarrow \mathcal{R} \mid - \vdash t^{\text{Rn}} : \tau^{\text{Rn}} \rightarrow \mathcal{R}
\]

Recovered as:

\[
\sigma^{\text{Rn}} \equiv^\circ (\sigma^n)^{\mathcal{C} \mathcal{R}}
\]

\[
t^{\text{Rn}} \equiv (t^n)^{\mathcal{V} \mathcal{R}}
\]
Fundamental theorem of EEC translation

Theorem (Involution property [EMS 2010]) We have isomorphisms

\[ A \cong A^\nu \nu \] 
\[ A \cong^\circ A^{CR} CR \]

modulo which \( t = t^{\nu \nu} \) and \( u = u^{CR} CR \).

Our proof of this is semantic. The translation arises as a self-duality of the initial (syntactic) model of EEC. (Obtaining a syntactic proof looks like a formidable exercise.)

Corollary The translations \((\cdot)^\nu\nu\) and \((\cdot)^{CR}\) are full and faithful.

Corollary The translations \((\cdot)^{Rv}\) and \((\cdot)^{Rn}\) are full and faithful.

[Hasegawa 2002, 2004] has analogous results for the \((\cdot)^{Rv}\) and \((\cdot)^{Rn}\) translations into ILL rather than EEC.
Part 4

Adding Control to EEC
**EEC + Control**

New value type: \( \text{stk } A \).

New typing rules:

\[
\begin{align*}
\Gamma, x : \text{stk } A \mid \Delta & \vdash t : A & \Gamma \mid \Delta \vdash \text{letstk } x. t : A \\
\Gamma \mid \Delta \vdash t : \text{stk } A & \Gamma \mid \Delta \vdash u : A & \Gamma \mid \Delta \vdash \text{chngstk } t. u : B \\
\Gamma \mid \Delta \vdash t : \text{stk } B & \Gamma \mid \Delta \vdash z : A \vdash u : B & \Gamma \mid \Delta \vdash \text{ignored} : \text{stk } 0 \\
\Gamma \mid \Delta \vdash t \circ [z]u : \text{stk } A & \Gamma \mid \Delta \vdash \text{letstk } x. t : A
\end{align*}
\]

Plus suitable equations.

Modelled on CBPV + Control [Levy 2004]
A reformulation of EEC + Control

Interpret stk $A$ as $A \to 0$ and add canonical isomorphisms

$$A \cong (A \to 0) \to 0$$

I.e., add constants:

$$r_A : ((A \to 0) \to 0) \to A$$

and equations making them inverse to the canonical

$$\lambda z : A. \lambda y : A \to 0. y[z] : A \to (A \to 0) \to 0$$

This appears in [MS 2007] in the context of a polymorphic language.

(N.B. Canonical $A \cong (A \to 0) \to 0$ is incompatible with ILL)
Jump With Argument (JWA)

If we add canonical isomorphisms for computation and value types

\[
A \cong^\circ (A \rightarrow 0) \rightarrow 0 \quad (1)
\]

\[
A \cong (A \rightarrow 0) \rightarrow 0 \quad (2)
\]

we obtain Jump With Argument [Levy 2004].

Principle (2) is harmless in the sense that EEC + (1) + (2) can be interpreted in EEC + (1) without changing the computation types.

This is a reformulation of the equivalence between CBPV + Control and JWA in [Levy 2004]
Summary

- The enriched effect calculus conservatively extends effect calculi (Moggi’s computational metalanguage, CBPV) with much of the expressivity of intuitionistic linear logic.

- Models of EEC strictly generalises models of ILL.

- The cbv and cbn linearly-used CPS translations extend to a translation of EEC into itself. (Such a self translation is not available for for ILL or CBPV.)

- Control can be added to EEC by simple axioms involving $\geq$.
  (Such axioms do not make sense in ILL.)

In general EEC is a promising language for representing linear usage of effects. E.g., linear-usage of (local) state [Møgelberg & Staton].
EMS literature references

