Constructive Completeness Theorems and Delimited Control

Danko Ilik

École Polytechnique, INRIA and Université Paris 7, France

(joint work with Hugo Herbelin, partly Gyesik Lee)

Kanazawa, March 8, 2010

Motivation

- Make meta-theory in Coq easier
  - A completeness theorem = a link between shallow and deep embeddings
- Unresolved foundational questions
  - What is the algorithm behind some completeness proofs?
  - Is intuitionistic logic complete for Kripke models?
  - What is a good logical specification for `shift` and `reset`?
  - Is bar recursion constructive?

Part I

Classical Completeness à la Krivine

Part II

Classical Completeness via Kripke-style Models

Kripke-style models

- Boolean completeness not canonical
- Would be nice to get a completeness theorem for **computational** classical calculi – reduction relation should be preserved
- Following normalization-by-evaluation (NBE) methodology, we define a non-Boolean notion of model for classical logic

Kripke-style models (Call-by-value)

Like with Kripke models, start with a structure $(K, \leq, D, \Vdash_s, \Vdash_\bot)$, and extend $\Vdash_s$ to non-atomic formulas:

$\mathbf{w \Vdash_s}$	
$A \wedge B$	$w \Vdash A$ and $w \Vdash B$
$A \vee B$	$w \Vdash A$ or $w \Vdash B$
$A \to B$	for any $w' \geq w$, if $w' \Vdash A$ then $w' \Vdash B$
$\forall x P(x)$	for any $w' \geq w$ and any $a \in D(w')$, $w' \Vdash P(a)$
$\exists x P(x)$	there is $a \in D(w)$ such that $w \Vdash P(a)$

where the non-s-annotated $\Vdash$ is **(non-strong) forcing**:

$$w \Vdash A := \forall w_1 \geq w . (\underbrace{\forall w_2 \geq w_1 . w_2 \Vdash_s A \to w_2 \Vdash_\bot}_{\text{"refutation"} \; w_1 : A \Vdash}) \to w_1 \Vdash_\bot$$

Completeness for Kripke-style models and $\mathrm{LK}_{\mu\tilde{\mu}}$

Theorem (Soundness)

$c : (\Gamma \vdash \Delta) \implies$ *for any* w, $w \Vdash \Gamma$ *and* $w : \Delta \Vdash$ *implies* $w \Vdash_{\perp}$
$\Gamma \vdash t : A|\Delta \implies$ *for any* w, $w \Vdash \Gamma$ *and* $w : \Delta \Vdash$ *implies* $w \Vdash A$
$\Gamma|e : A \vdash \Delta \implies$ *for any* w, $w \Vdash \Gamma$ *and* $w : \Delta \Vdash$ *implies* $w : A \Vdash$

Theorem (Completeness)

$(\Gamma, \Delta) \Vdash A \implies$ *there is a term* t *such that* $\Gamma \vdash_{cf} t : A|\Delta$
$(\Gamma, \Delta) : A \Vdash \implies$ *there is an ev. context* e *such that* $\Gamma|e : A \vdash_{cf} \Delta$

Theorem (NBE)

The composition (Completeness $\circ$ *Soundness) normalizes proof terms into* η*-long* β*-normal form*

Kripke-style models, final remarks

- Proof formalised in Coq
- Dual notion of model that gives call-by-**name** normalization strategy
- Article to appear in "Classical Logic & Computation 2008", with Hugo Herbelin (INRIA, France) and Gyesik Lee (ROSAEC, Korea)

Part III

Intuitionistic Completeness

Constructive Completeness of Intuitionistic Logic

- Constructive completeness w.r.t Beth models, Topological models, Sambin-Macedonio models, ...
- **No** Curry-Howard-constructive proofs for Kripke models:
  - classical Henkin-style proofs (eg. Troelstra-van Dalen)
  - using fan theorem (Veldman 1976)
  - a constructive proof would imply MP (Kreisel 1962)
- Well-typed functional program for NBE of $\lambda^{\rightarrow\vee}$ (Danvy 1996)
  - using delimited-control operators *shift* and *reset* (Danvy-Filinski 1989)

Completeness/NBE for $\lambda^{\to\vee}$

What the problem is

Theorem (NBE)

$\downarrow^A_\Gamma$ ("*reify*") : $\quad \Gamma \Vdash A \longrightarrow \Gamma \vdash^{nf} A$
$\uparrow^A_\Gamma$ ("*reflect*") : $\quad \Gamma \vdash^{ne} A \longrightarrow \Gamma \Vdash A$

Proof of case $\uparrow^{A_1 \vee A_2}$.

Given a derivation $\Gamma \vdash^{\mathrm{ne}} A_1 \vee A_2$, decide: $\Gamma \Vdash A_1$ **or** $\Gamma \Vdash A_2$. $\square$

Delimited Control Operators [Feleisen 1986]; `shift` ($\mathcal{S}$) and `reset` ($\#$) [Danvy-Filinski 1989] – Example

$$5 + \#\,(3 + \mathcal{S}k.\,(k \cdot 0 + k \cdot 1))$$

Delimited Control Operators [Feleisen 1986]; `shift` ($\mathcal{S}$) and `reset` ($\#$) [Danvy-Filinski 1989] – Example

$$
\begin{array}{ll}
 & 5 + \#\,(3 + \mathcal{S}k.\,(k \cdot 0 + k \cdot 1)) \\
\mapsto & 5 + \#\,(k \cdot 0 + k \cdot 1)\,\{k := \lambda x.\#(3 + x)\}
\end{array}
$$

Delimited Control Operators [Feleisen 1986]; `shift` ($\mathcal{S}$) and `reset` ($\#$) [Danvy-Filinski 1989] – Example

$$
\begin{array}{rl}
 & 5 + \#\,(3 + \mathcal{S}k.\,(k \cdot 0 + k \cdot 1)) \\
\mapsto & 5 + \#\,(k \cdot 0 + k \cdot 1)\,\{k := \lambda x.\#(3 + x)\} \\
\equiv & 5 + \#\,((\lambda x.\#(3 + x)) \cdot 0 + (\lambda x.\#(3 + x)) \cdot 1)
\end{array}
$$

Delimited Control Operators [Feleisen 1986]; shift ($\mathcal{S}$) and reset (#) [Danvy-Filinski 1989] – Example

$$
\begin{array}{rc}
 & 5 + \#\,(3 + \mathcal{S}k.\,(k \cdot 0 + k \cdot 1)) \\
\mapsto & 5 + \#\,(k \cdot 0 + k \cdot 1)\,\{k := \lambda x.\#(3 + x)\} \\
\equiv & 5 + \#\,((\lambda x.\#(3 + x)) \cdot 0 + (\lambda x.\#(3 + x)) \cdot 1) \\
\mapsto^{+} & 5 + \#\,(\#3 + \#4)
\end{array}
$$

Delimited Control Operators [Feleisen 1986]; `shift` ($\mathcal{S}$) and `reset` ($\#$) [Danvy-Filinski 1989] – Example

$$
\begin{array}{rc}
 & 5 + \#\,(3 + \mathcal{S}k.\,(k \cdot 0 + k \cdot 1)) \\
\mapsto & 5 + \#\,(k \cdot 0 + k \cdot 1)\,\{k := \lambda x.\#(3 + x)\} \\
\equiv & 5 + \#\,((\lambda x.\#(3 + x)) \cdot 0 + (\lambda x.\#(3 + x)) \cdot 1) \\
\mapsto^{+} & 5 + \#\,(\#3 + \#4) \\
\mapsto & 5 + \#\,(3 + 4)
\end{array}
$$

Delimited Control Operators [Feleisen 1986]; shift ($\mathcal{S}$) and reset ($\#$) [Danvy-Filinski 1989] – Example

$$
\begin{array}{rc}
 & 5 + \#\,(3 + \mathcal{S}k.\,(k \cdot 0 + k \cdot 1)) \\
\mapsto & 5 + \#\,(k \cdot 0 + k \cdot 1)\,\{k := \lambda x.\#(3 + x)\} \\
\equiv & 5 + \#\,((\lambda x.\#(3 + x)) \cdot 0 + (\lambda x.\#(3 + x)) \cdot 1) \\
\mapsto^{+} & 5 + \#\,(\#3 + \#4) \\
\mapsto & 5 + \#\,(3 + 4) \\
\mapsto & 5 + \#7
\end{array}
$$

Delimited Control Operators [Feleisen 1986]; `shift` ($\mathcal{S}$) and `reset` ($\#$) [Danvy-Filinski 1989] – Example

$$
\begin{array}{rc}
 & 5 + \#\,(3 + \mathcal{S}k.\,(k \cdot 0 + k \cdot 1)) \\
\mapsto & 5 + \#\,(k \cdot 0 + k \cdot 1)\,\{k := \lambda x.\#(3 + x)\} \\
\equiv & 5 + \#\,((\lambda x.\#(3 + x)) \cdot 0 + (\lambda x.\#(3 + x)) \cdot 1) \\
\mapsto^{+} & 5 + \#\,(\#3 + \#4) \\
\mapsto & 5 + \#\,(3 + 4) \\
\mapsto & 5 + \#7 \\
\mapsto & 5 + 7 \\
\mapsto & 12
\end{array}
$$

Completeness/NBE for $\lambda^{\to\vee}$

Solution of Danvy: use delimited control operators shift ($\mathcal{S}$) and reset ($\#$)

Theorem (NBE)

$\downarrow^A_\Gamma$ ("*reify*") : $\quad \Gamma \Vdash A \longrightarrow \Gamma \vdash^{nf} A$
$\uparrow^A_\Gamma$ ("*reflect*") : $\quad \Gamma \vdash^{ne} A \longrightarrow \Gamma \Vdash A$

Proof of case $\uparrow^{A_1\vee A_2}$.

Given a derivation e of $\Gamma \vdash^{\mathrm{ne}} A_1 \vee A_2$, decide: $\Gamma \Vdash A_1$ **or** $\Gamma \Vdash A_2$, by

$$\mathcal{S}k.\ \vee_E\ \ e\ (x \hookrightarrow k(\mathrm{left} \uparrow^{A_1}_{x:A_1,\Gamma} x))\ (y \hookrightarrow k(\mathrm{right} \uparrow^{A_2}_{y:A_2,\Gamma} y))$$

where

$$\#V \mapsto V$$

$$\#F[\mathcal{S}k.p] \mapsto \#p\{k := \lambda x.\#F[x]\}$$

□

Completeness/NBE for $\lambda^{\rightarrow\vee}$

Solution of Danvy: Issues

- We are convinced the **program** computes correctly
- There should be a corresponding **proof** for Kripke completeness
- Type-and-effect system: types $A \rightarrow B$ become $A/\alpha \rightarrow B/\beta$, what is the logical meaning?

Completeness for IQC

Extracting a notion of model from Danvy's Solution

Like with Kripke models, start with a structure $(K, \leq, D, \Vdash_s, \Vdash_\perp^{(-)})$, and extend $\Vdash_s$ to non-atomic formulas:

$\mathbf{w} \Vdash_\mathbf{s}$	
$A \wedge B$	$w \Vdash A$ and $w \Vdash B$
$A \vee B$	$w \Vdash A$ or $w \Vdash B$
$A \to B$	for any $w' \geq w$, if $w' \Vdash A$ then $w' \Vdash B$
$\forall x P(x)$	for any $w' \geq w$ and any $a \in D(w')$, $w' \Vdash P(a)$
$\exists x P(x)$	there is $a \in D(w)$ such that $w \Vdash P(a)$

where the non-s-annotated $\Vdash$ is **(non-strong) forcing**:

$$w \Vdash A := \forall C. \forall w_1 \geq w. (\forall w_2 \geq w_1 . w_2 \Vdash_s A \to w_2 \Vdash_\perp^C) \to w_1 \Vdash_\perp^C$$

Completeness for IQC via Kripke-style models

Theorem (NBE)

$\downarrow^A_\Gamma$ ("*reify*") : $\quad \Gamma \Vdash A \longrightarrow \Gamma \vdash^{nf} A$
$\uparrow^A_\Gamma$ ("*reflect*") : $\quad \Gamma \vdash^{ne} A \longrightarrow \Gamma \Vdash A$

Proof of case $\uparrow^{A_1 \vee A_2}$.

Given a derivation e of $\Gamma \vdash^{\mathrm{ne}} A_1 \vee A_2$, prove $\Gamma \Vdash A_1 \vee A_2$ by

$$\lambda k.\ \vee_E\ \ e\ (x \hookrightarrow k(\mathrm{left} \uparrow^{A_1}_{x:A_1,\Gamma} x))\ (y \hookrightarrow k(\mathrm{right} \uparrow^{A_2}_{y:A_2,\Gamma} y))$$

$\square$

formalised in Coq, computes "normal forms" of $\lambda^{\to\vee}$-terms.

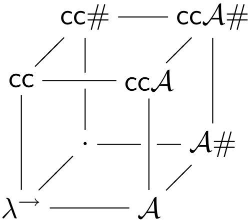
Part IV

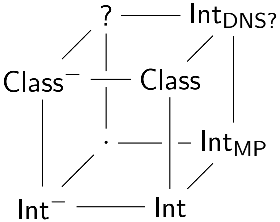
Direct Style Proofs (work in progress)

Direct Style Proofs

- Kripke-style models work, but:
  - semantics not satisfying
  - using them requires bureaucratic “lifting”
- Also interested in computational effects:
  - int. logic supports only effect-free (“pure”) computation
  - shift and reset can simulate *any* monadic effect (Filinski 1994)
- “An intuitionistic logic that proves Markov’s principle” (Herbelin 2010)

Delimited-Control Decomposition Cube





cc	call/cc
$\mathcal{A}$	abort
#	reset

$\mathsf{Int}^{(-)}$	(min.) intuitionistic
$\mathsf{Class}^{(-)}$	(min.) classical
MP	Markov's principle
DNS	Double-negation shift

The system $\mathrm{Int}_{\mathrm{DNS?}}$

Natural deduction rules for Int, but writing $\vdash_\Delta$ instead of $\vdash$, plus two rules:

$$\frac{\Gamma \vdash_{T,\Delta} t : T}{\Gamma \vdash_\Delta \#t : T}\ \text{reset} \qquad\qquad \frac{\Gamma, k : A \to T \vdash_{T,\Delta} t : T}{\Gamma \vdash_{T,\Delta} \mathcal{S}k.t : A}\ \text{shift}$$

where T, Δ is a stack of Σ-formulas ($\{\to, \forall\}$-free)

We proved:

- Translation to intuitionistic logic (equi-consistency, normalization)
- Subject reduction: $\Gamma \vdash_\Delta t : A \;\&\; t \mapsto t' \longrightarrow \Gamma \vdash_\Delta t' : A$

To be proved:

- Characterization of normal forms, Progress

The system $\mathrm{Int}_{\mathrm{DNS?}}$

- A constructive logic – Disjunction and Existence Properties
- Herbelin: `shift` derives Double Negation Shift

$$
\dfrac{\cdots \qquad \dfrac{\dfrac{\dfrac{\cdots}{H : \forall n.(A(n) \to X) \to X, k : A(n) \to X \vdash_X Hn(\lambda a.ka) : X}}{H : \forall n.(A(n) \to X) \to X, \vdash_X \mathcal{S}k.Hn(\lambda a.ka) : A(n)}\,\textit{shift}}{H : \forall n.(A(n) \to X) \to X, \vdash_X \lambda n.\mathcal{S}k.Hn(\lambda a.ka) : \forall n.A(n)}}{\dfrac{H : \forall n.(A(n) \to X) \to X, f : (\forall n.A(n)) \to X \vdash_X f(\lambda n.\mathcal{S}k.Hn(\lambda a.ka)) : X}{\dfrac{H : \forall n.(A(n) \to X) \to X, f : (\forall n.A(n)) \to X \vdash \# f(\lambda n.\mathcal{S}k.Hn(\lambda a.ka)) : X}{\vdash \lambda H.\lambda f.\# f(\lambda n.\mathcal{S}k.Hn(\lambda a.ka)) : (\forall n.(A(n) \to X) \to X) \to ((\forall n.A(n)) \to X) \to X}}\,\textit{reset}}
$$

The system $\mathrm{Int}_{\text{D-DNS?}}$ (work-in-progress)

What about completeness w.r.t. Kripke models? We need a stronger version:

$$\frac{\Gamma, k : \forall w' \geq w. A(w') \to T(w') \vdash_{T(w),\Delta} t : T(w)}{\Gamma \vdash_{T(w),\Delta} \mathcal{S}k.t : A(w)} \text{ D-shift}$$

The system $\mathsf{Int}_{\mathsf{DNS?}}$

Translation to 2nd-order int. logic with predicative polymorphism

$$\overline{A} = \quad \forall T[(\underline{A} \to T) \to T]$$

$$\begin{array}{rcl}
\underline{A} = & A & \text{if } A \text{ is atomic} \\
\underline{A \Box B} = & \overline{A} \Box \overline{B} & \text{for } \Box = \vee, \wedge, \to \\
\underline{\Box A} = & \Box \overline{A} & \text{for } \Box = \exists, \forall
\end{array}$$

$$\begin{array}{rcl}
B_1, \ldots, B_n \vdash_{T,\Delta} A & \rightsquigarrow & \overline{B_1} \cdot T, \ldots, \overline{B_n} \cdot T \vdash_{\Delta} \overline{A} \cdot T \\
B_1, \ldots, B_n \vdash_{\emptyset} A & \rightsquigarrow & B_1, \ldots, B_n \vdash A
\end{array}$$

The system $\mathrm{Int}_{\mathrm{DNS?}}$

Translation to 2nd-order int. logic with predicative polymorphism

The key: we should be able to "run" the monad at Σ-formulas

Theorem

For any Σ-formula T we have $\vdash_\Delta \underline{T} \to T$

Proof.

Induction on complexity of T:

T-atomic $\quad \mathbf{\Theta}_T(x) = x$

$T = U \wedge V \quad \mathbf{\Theta}_{U \wedge V}(f, g) = f \cdot (x \mapsto g \cdot (y \mapsto (\mathbf{\Theta}_U(x), \mathbf{\Theta}_V(y))))$

Being $\{\to, \forall\}$-free is crucial; using a polymorphic λ-calculus as target also crucial. $\square$

Future work

- Work out D-DNS version
- Compare to bar recursion (open induction)
- Compare to Krivine's calculi for Dependent Choice