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 \((K, \leq, D, \models_s, \models_\bot)\), and extend \(\models_s\) to non-atomic formulas:

- \(w \models_s\)
- \(A \land B\) \(w \models A\) and \(w \models B\)
- \(A \lor B\) \(w \models A\) or \(w \models B\)
- \(A \rightarrow B\) for any \(w' \geq w\), if \(w' \models A\) then \(w' \models B\)
- \(\forall x P(x)\) for any \(w' \geq w\) and any \(a \in D(w')\), \(w' \models P(a)\)
- \(\exists x P(x)\) there is \(a \in D(w)\) such that \(w \models P(a)\)

where the non-s-annotated \(\models\) is (non-strong) forcing:

\[
\text{where the non-s-annotated } \models \text{ is (non-strong) forcing:}
\]

\[
w \models A := \forall w_1 \geq w. (\forall w_2 \geq w_1. w_2 \models_s A \rightarrow w_2 \models_\bot) \rightarrow w_1 \models_\bot
\]

"refutation" \(w_1: A \models_\bot\)
Completeness for Kripke-style models and LK_{\mu \bar{\mu}}

**Theorem (Soundness)**

\[ c : (\Gamma \vdash \Delta) \implies \text{for any } w, w \models \Gamma \text{ and } w : \Delta \vdash \text{ implies } w \models \bot \]
\[ \Gamma \vdash t : \Delta \implies \text{for any } w, w \models \Gamma \text{ and } w : \Delta \vdash \text{ implies } w \models A \]
\[ \Gamma \vdash e : A \vdash \Delta \implies \text{for any } w, w \models \Gamma \text{ and } w : \Delta \vdash \text{ implies } w : A \models \]

**Theorem (Completeness)**

\[(\Gamma, \Delta) \vdash A \implies \text{there is a term } t \text{ such that } \Gamma \vdash_{cf} t : A \mid \Delta \]
\[(\Gamma, \Delta) : A \models \implies \text{there is an ev. context } e \text{ such that } \Gamma \vdash e : A \vdash_{cf} \Delta \]

**Theorem (NBE)**

*The composition (Completeness \circ Soundness) normalizes proof terms into \eta-long \beta-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 $\lambda \rightarrow^\gamma$ (Danvy 1996)
  - using delimited-control operators $\text{shift}$ and $\text{reset}$ (Danvy-Filinski 1989)
Completeness/NBE for $\lambda \to \lor$

What the problem is

Theorem (NBE)

$\downarrow^A_\Gamma ("\text{reify}") : \quad \Gamma \vdash A \longrightarrow \Gamma \vdash^{nf} A$

$\uparrow^A_\Gamma ("\text{reflect}") : \quad \Gamma \vdash^{ne} A \longrightarrow \Gamma \Vdash A$

Proof of case $\uparrow^{A_1 \lor A_2}$. Given a derivation $\Gamma \vdash^{ne} A_1 \lor A_2$, decide: $\Gamma \Vdash A_1$ or $\Gamma \Vdash A_2$. □
Delimited Control Operators [Feleisen 1986]; shift ($S$) and reset ($#$) [Danvy-Filinski 1989] – Example

$$5 + # (3 + S k. (k \cdot 0 + k \cdot 1))$$
Delimited Control Operators [Feleisen 1986]; shift ($S$) and reset ($#$) [Danvy-Filinski 1989] – Example

\[
5 + \#(3 + Sk. (k \cdot 0 + k \cdot 1)) \\
\mapsto 5 + \#(k \cdot 0 + k \cdot 1) \{ k := \lambda x.\#(3 + x) \}
\]
Delimited Control Operators [Feleisen 1986]; shift ($S$) and reset ($\#$) [Danvy-Filinski 1989] – Example

\[
5 + \# (3 + S k. (k \cdot 0 + k \cdot 1)) \\
\mapsto 5 + \# (k \cdot 0 + k \cdot 1) \{ k := \lambda x.\#(3 + x) \} \\
\equiv 5 + \# ((\lambda x.\#(3 + x)) \cdot 0 + (\lambda x.\#(3 + x)) \cdot 1)
\]
Delimited Control Operators [Feleisen 1986]; shift ($S$) and reset ($#$) [Danvy-Filinski 1989] – Example

\[ 5 + # (3 + Sk. (k \cdot 0 + k \cdot 1)) \]
\[ \mapsto 5 + # (k \cdot 0 + k \cdot 1) \{ k := \lambda x. #(3 + x) \} \]
\[ \equiv 5 + # ((\lambda x. #(3 + x)) \cdot 0 + (\lambda x. #(3 + x)) \cdot 1) \]
\[ \mapsto^+ 5 + # (#3 + #4) \]
Delimited Control Operators [Feleisen 1986]; shift (S) and reset (#) [Danvy-Filinski 1989] – Example

$$5 + # (3 + Sk. (k \cdot 0 + k \cdot 1))$$

$$\mapsto 5 + # (k \cdot 0 + k \cdot 1) \{ k := \lambda x. #(3 + x) \}$$

$$\equiv 5 + # ((\lambda x. #(3 + x)) \cdot 0 + (\lambda x. #(3 + x)) \cdot 1)$$

$$\mapsto^+ 5 + # (#3 + #4)$$

$$\mapsto 5 + # (3 + 4)$$
Delimited Control Operators [Feleisen 1986]; shift ($S$) and reset ($#$) [Danvy-Filinski 1989] – Example

\[
5 + #\left(3 + Sk.\left(k \cdot 0 + k \cdot 1\right)\right)
\]

\[
\mapsto 5 + #\left(k \cdot 0 + k \cdot 1\right) \{ k := \lambda x.\#(3 + x) \}
\]

\[
\equiv 5 + #\left((\lambda x.\#(3 + x)) \cdot 0 + (\lambda x.\#(3 + x)) \cdot 1\right)
\]

\[
\mapsto^+ 5 + #\left(#3 + #4\right)
\]

\[
\mapsto 5 + #\left(3 + 4\right)
\]

\[
\mapsto 5 + #7
\]

\[
5 + 7 \mapsto 12
\]
Delimited Control Operators [Feleisen 1986]; shift ($S$) and reset (#) [Danvy-Filinski 1989] – Example

$$5 + \# (3 + Sk. (k \cdot 0 + k \cdot 1))$$

$\mapsto$ $5 + \# (k \cdot 0 + k \cdot 1) \{ k := \lambda x.\#(3 + x) \}$

$\equiv$ $5 + \# (\lambda x.\#(3 + x)) \cdot 0 + (\lambda x.\#(3 + x)) \cdot 1$

$\mapsto^+$ $5 + \# (\#3 + \#4)$

$\mapsto$ $5 + \# (3 + 4)$

$\mapsto$ $5 + \#7$

$\mapsto$ $5 + 7$

$\mapsto$ $12$
Completeness/NBE for $\lambda \rightarrow^\lor$

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

**Theorem (NBE)**

$\Downarrow^A_{\Gamma} (" \text{reify}\") : \Gamma \vdash A \longrightarrow \Gamma \vdash^{nf} A$

$\Uparrow^A_{\Gamma} (" \text{reflect}\") : \Gamma \vdash^{ne} A \longrightarrow \Gamma \dashv A$

**Proof of case $\Uparrow^{A_1 \lor A_2}_\Gamma$.**

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

$$S \ k \ \lor_E \ e \ (x \leftrightarrow k(\text{left } \Uparrow^{A_1}_{x:A_1,\Gamma} x)) \ (y \leftrightarrow k(\text{right } \Uparrow^{A_2}_{y:A_2,\Gamma} y))$$

where

$$\# V \mapsto V$$

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

\((K, \leq, D, \models_s, \models_{(\bot)})\), and extend \(\models_s\) to non-atomic formulas:

\[ w \models_s \]

\[ A \land B \quad w \models A \text{ and } w \models B \]

\[ A \lor B \quad w \models A \text{ or } w \models B \]

\[ A \rightarrow B \quad \text{for any } w' \geq w, \text{ if } w' \models A \text{ then } w' \models B \]

\[ \forall xP(x) \quad \text{for any } w' \geq w \text{ and any } a \in D(w'), w' \models P(a) \]

\[ \exists xP(x) \quad \text{there is } a \in D(w) \text{ such that } w \models P(a) \]

where the non-s-annotated \(\models\) is \textbf{(non-strong) forcing}:

\[ w \models A := \forall C.\forall w_1 \geq w.(\forall w_2 \geq w_1.w_2 \models_s A \rightarrow w_2 \models_{\bot}) \rightarrow w_1 \models_{\bot} \]
Completeness for IQC via Kripke-style models

Theorem (NBE)
\[ \downarrow^A ("\text{reify}") : \Gamma \vdash A \longrightarrow \Gamma \vdash^{nf} A \]
\[ \uparrow^A ("\text{reflect}") : \Gamma \vdash^{ne} A \longrightarrow \Gamma \models A \]

Proof of case \( \uparrow^{A_1 \lor A_2} \).

Given a derivation \( e \) of \( \Gamma \vdash^{ne} A_1 \lor A_2 \), prove \( \Gamma \models A_1 \lor A_2 \) by

\[ \lambda k. \lor_E e (x \leftarrow k(\text{left } \uparrow^{A_1}_{x:A_1, \Gamma} x)) (y \leftarrow k(\text{right } \uparrow^{A_2}_{y:A_2, \Gamma} y)) \]

formalised in Coq, computes "normal forms" of \( \lambda \to \lor \)-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

\[ \lambda \rightarrow A \]

cc call/cc

\( \# \) reset

\( \text{cc} \) cc

\( \text{cc} \) A

\( \text{cc} \cdot \) A #

Class

\( \text{?} \) \( \rightarrow \) \( \text{Int}_{\text{DNS}}? \)

Class

\( \text{?} \) \( \rightarrow \) \( \text{Int}_{\text{MP}} \)

\( \text{?} \) \( \rightarrow \) \( \text{Int} \)

Int (−) (min.) intuitionistic

Class (−) (min.) classical

MP Markov’s principle

DNS Double-negation shift
The system \( \text{Int}_{DNS?} \)

Natural deduction rules for Int, but writing \( \vdash_{\Delta} \) instead of \( \vdash \), plus two rules:

\[
\frac{\Gamma \vdash_{T,\Delta} t : T}{\Gamma \vdash_{\Delta} \#t : T} \quad \text{reset}
\]

\[
\frac{\Gamma, k : A \to T \vdash_{T,\Delta} t : T}{\Gamma \vdash_{T,\Delta} Sk.t : A} \quad \text{shift}
\]

where \( T, \Delta \) is a stack of \( \Sigma \)-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 \( \text{Int}_{\text{DNS}} \)?

- A constructive logic – Disjunction and Existence Properties
- Herbelin: \texttt{shift} derives Double Negation Shift

\[
\begin{align*}
\vdash \lambda H. \lambda f. \#f(\lambda n.Sk.Hn(\lambda a.ka)) : (\forall n.(A(n) \rightarrow X) \rightarrow X) \rightarrow ((\forall n.A(n)) \rightarrow X) \rightarrow X
\end{align*}
\]
The system \( \text{Int}_{D\text{-DNS?}} \) (work-in-progress)

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

\[
\Gamma, \ k : \forall w' \geq w. A(w') \rightarrow T(w') \vdash_{T(w), \Delta} t : T(w)
\]

\[
\Gamma \vdash_{T(w), \Delta} Sk.t : A(w)
\]

D-shift
The system $\text{Int}_{\text{DNS}}$?
Translation to 2nd-order int. logic with predicative polymorphism

\[
\overline{A} = \forall T[(A \rightarrow T) \rightarrow T]
\]

\[
\overline{A} = \begin{cases} 
A & \text{if } A \text{ is atomic} \\
A \square B & \text{for } \square = \lor, \land, \rightarrow \\
\square A & \text{for } \square = \exists, \forall
\end{cases}
\]

\[
\begin{array}{c}
B_1, \ldots, B_n \vdash_{T, \Delta} A \\
B_1, \ldots, B_n \vdash_{\emptyset} A
\end{array}
\Rightarrow
\begin{array}{c}
\overline{B_1} \cdot T, \ldots, \overline{B_n} \cdot T \vdash_{\Delta} \overline{A} \cdot T \\
B_1, \ldots, B_n \vdash A
\end{array}
The key: we should be able to “run” the monad at $\Sigma$-formulas

**Theorem**

For any $\Sigma$-formula $T$ we have $\vdash_\Delta T \rightarrow T$

**Proof.**

Induction on complexity of $T$:

- $T$-atomic: $\Theta_T(x) = x$
- $T = U \land V$: $\Theta_{U \land V}(f, g) = f \cdot (x \mapsto g \cdot (y \mapsto (\Theta_U(x), \Theta_V(y))))$

Being $\{\rightarrow, \forall\}$-free is crucial; using a polymorphic $\lambda$-calculus as target also crucial.
Future work

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