Intuitionistically Proving Markov's principle Thanks to Delimited control

Hugo Herbelin

30 January 2010

3rd workshop on formal and automated theorem proving and applications - COST IC0901

The Curry-Howard proofs-as-programs correspondence

Hilbert's style logic	=	simply-typed combinatory logic	(Curry [1958])
natural deduction	=	simply-typed λ-calculus	(Howard [1969])
classical logic	=	control operators	(Griffin [1990], Parigot [1992])
sequent calculus	=	terms/evaluation contexts + call-by-name/call-by-value duality	
Markov's principle	=	?	

Markov's principle

$$\neg\neg\exists x\, A(x) \to \exists x\, A(x) \qquad \text{for } A(x) \text{ decidable}$$

- classically provable

- not provable in (standard) intuitionistic logic (no simply-typed realiser, Kreisel [1958])

- asserts that classical logic is conservative over intuitionistic logic for $\exists x\, A(x)$ statements ($A(x)$ decidable)

- useful for program extraction in constructive analysis (implies $\neg x = y \to x \# y$ on real numbers)

  $\hookrightarrow$ conventional realiser is unbounded search, testing $A(0)$, $A(1)$, ... until finding some $A(n_0)$ that holds

- admissible as a rule (Friedman's A-translation [1978])

  $\hookrightarrow$ Friedman's result applies to any formula intuitionistically equivalent to an $\to$-$\forall$-formula

From now on, we implicitly assume that A is $\to$-free in $\exists x\, A(x)$ and we place ourselves in predicate logic

Main result

Intuitionistic logic + classical reasoning limited to $\exists x\, A(x)$ formulae

- still satisfies the characteristic disjunction and existence properties of intuitionistic logic

$$\vdash A \vee B \text{ implies } \vdash A \text{ or } \vdash B$$
$$\vdash \exists x\, A(x) \text{ implies } \vdash A(t) \text{ for some } t$$

- proves Markov's principle and there are two possible proof-terms with standard computational contents

$$\lambda H.\, \mathtt{callcc}(\lambda k.H\,(\lambda x.\mathtt{throw}\, k\, x))$$
$$\lambda H.\, \mathtt{try}_E\, H(\lambda x.\mathtt{raise}_E\, x)$$

where $H : \neg\neg\exists x\ A(x)$

IQC_{MP} annotated with proof-terms

$$\frac{(a : A) \in \Gamma}{\Gamma \vdash_\Delta a : A} \qquad \frac{}{\Gamma \vdash_\Delta () : \top} \qquad \frac{\Gamma \vdash_\Delta p : \bot}{\Gamma \vdash_\Delta \mathtt{efq}\ p : C}$$

$$\frac{\Gamma \vdash_\Delta p_1 : A_1 \qquad \Gamma \vdash_\Delta p_2 : A_2}{\Gamma \vdash_\Delta (p_1, p_2) : A_1 \wedge A_2} \qquad \frac{\Gamma \vdash_\Delta p : A_1 \wedge A_2}{\Gamma \vdash_\Delta \pi_i\, p : A_i}$$

$$\frac{\Gamma \vdash_\Delta p : A_i}{\Gamma \vdash_\Delta \iota_i(p) : A_1 \vee A_2} \qquad \frac{\Gamma \vdash_\Delta p : A_1 \vee A_2 \qquad \Gamma, a_1 : A_1 \vdash_\Delta p_1 : B \qquad \Gamma, a_2 : A_2 \vdash_\Delta p_2 : B}{\Gamma \vdash_\Delta \mathtt{case}\ p\ \mathtt{of}\ [a_1.p_1 \mid a_2.p_2] : B}$$

$$\frac{\Gamma, a : A \vdash_\Delta p : B}{\Gamma \vdash_\Delta \lambda a.p : A \to B} \qquad \frac{\Gamma \vdash_\Delta p : A \to B \qquad \Gamma \vdash_\Delta q : A}{\Gamma \vdash_\Delta p\, q : B}$$

$$\frac{\Gamma \vdash_\Delta p : A(x) \qquad x \text{ fresh}}{\Gamma \vdash_\Delta \lambda x.p : \forall x\, A(x)} \qquad \frac{\Gamma \vdash_\Delta p : \forall x\, A(x)}{\Gamma \vdash_\Delta p\, t : A(t)}$$

$$\frac{\Gamma \vdash_\Delta p : A(t)}{\Gamma \vdash_\Delta (t, p) : \exists x\, A(x)} \qquad \frac{\Gamma \vdash_\Delta p : \exists x\, A(x) \qquad \Gamma, a : A(x) \vdash_\Delta q : B \qquad x \text{ fresh}}{\Gamma \vdash_\Delta \mathtt{dest}\ p\ \mathtt{as}\ (x, a)\ \mathtt{in}\ q : B}$$

$$\frac{\Gamma \vdash_{\alpha:\exists x\, A(x),\Delta} p : \exists x\, A(x)}{\Gamma \vdash_\Delta \mathtt{catch}_\alpha\, p : \exists x\, A(x)} \qquad \frac{\Gamma \vdash_\Delta p : \exists x\, A(x) \qquad (\alpha : \exists x\, A(x)) \in \Delta}{\Gamma \vdash_\Delta \mathtt{throw}_\alpha\, p : C}$$

First observation

Completeness $\Gamma \vdash A$ in IQC_{MP} iff $MP, \Gamma \vdash A$ in IQC.

Normalisation rules for IQC_{MP}: a call-by-value semantics

$$
\begin{array}{lcl}
V & ::= & a \mid \iota_i(V) \mid (V,V) \mid (t,V) \mid \lambda a.p \mid \lambda x.p \mid () \\
F[\,] & ::= & \mathtt{case}\ [\,]\ \mathtt{of}\ [a_1.p_1 \mid a_2.p_2] \mid \pi_i([\,]) \mid \mathtt{dest}\ [\,]\ \mathtt{as}\ (x,a)\ \mathtt{in}\ p \\
 & \mid & [\,]\,q \mid (\lambda x.q)\,[\,] \mid [\,]\,t \mid \mathtt{efq}\ [\,] \mid \mathtt{throw}_\alpha\ [\,] \mid \iota_i([\,]) \mid ([\,],p) \mid (V,[\,]) \mid (t,[\,])
\end{array}
$$

$$
\begin{array}{lcl}
(\lambda a.p)\,V & \to & p[V/a] \\
(\lambda x.p)\,t & \to & p[t/x] \\
\mathtt{case}\ \iota_i(V)\ \mathtt{of}\ [a_1.p_1 \mid a_2.p_2] & \to & p_i[V/a_i] \\
\mathtt{dest}\ (t,V)\ \mathtt{as}\ (x,a)\ \mathtt{in}\ p & \to & p[t/x][V/a] \\
\pi_i(V_1,V_2) & \to & V_i \\
F[\mathtt{efq}\ p] & \to & \mathtt{efq}\ p \\
F[\mathtt{throw}_\alpha p] & \to & \mathtt{throw}_\alpha p \\
\mathtt{catch}_\alpha \mathtt{throw}_\alpha p & \to & \mathtt{catch}_\alpha p \\
\mathtt{catch}_\alpha \mathtt{throw}_\beta V & \to & \mathtt{throw}_\beta\, V \quad (\alpha \neq \beta) \\
\mathtt{catch}_\alpha V & \to & V
\end{array}
$$

$\exists x\, A(x)$ formulae for A $\to$-free are "datatypes"... call-by-value ensures that any closed proof of such a formula reduces to value and that any "`throw`" initially present in the proof has been raised

Properties of the reduction system

The resulting reduction system is rich enough to ensure the normalisation of closed proofs

Subject reduction If $\Gamma \vdash p : A; \Delta$ and $p \longrightarrow q$ then $\Gamma \vdash q : A; \Delta$

Progress If $\vdash p : A; \Delta$ and p is not a (closed) value then p is reducible

Normalisation If $\vdash p : A; \Delta$ then p is normalisable (by either monadic-style interpretation or embedding in classical logic)

Existence property $\vdash \exists x\, A(x)$ implies $\vdash A(t)$ for some t

Disjunction property $\vdash A_1 \vee A_2$ implies $\vdash A_1$ or $\vdash A_2$

How it works

The general form of a closed proof of $\neg\neg\exists x\, B(x)$ is

$$\lambda k.k\,(t_1, \ldots (k\,(t_2, \ldots (k\,(t_n, V))\ldots))\ldots)$$

Applying Markov's principle gives

$$\mathtt{catch}_\alpha\,\mathtt{efq}\;\mathtt{throw}_\alpha\,(t_1, \ldots (\mathtt{throw}_\alpha\,(t_2, \ldots (\mathtt{throw}_\alpha\,(t_n, V))\ldots))\ldots)$$

and the evaluation strategy forces the evaluation to

$$(t_n, V)$$

Connection with Friedman's A-translation

Friedman proved that the following variant of Markov's rule for predicate logic is intuitionistically admissible for A a $\rightarrow$-free propositional formula:

$$\frac{\vdash \neg\neg\exists x\, A(x)}{\vdash \exists x\, A(x)}$$

The trick is to observe that any proof of $\Gamma \vdash B$ that uses $\bot$-elimination can be mapped to a $\bot$-free proof of $\Gamma_A \vdash B_A$ where the A-translation of B, written B_A, is defined compositionally except for $\bot$ and atoms:

$$\begin{aligned} \bot_A &\triangleq A \\ X_A &\triangleq X \vee A \end{aligned}$$

In a second step one observes that if B is $\forall$-$\rightarrow$-free, then $B_A \rightarrow B \vee A$ and hence $B_B \rightarrow B$. But then, if A is $\exists x\, B(x)$, from a proof of $\vdash \neg\neg A$, one gets a proof of $\vdash (A_A \rightarrow A) \rightarrow A$, hence a proof of $\vdash A$.

Direct-style vs monadic transformation

intuitionism + effect $\xrightarrow{\quad T \quad}$ pure intuitionism

control operators	$T(B) \triangleq \neg\neg B$
states	$T(B) \triangleq S \to B \wedge S$
exceptions	$T(B) \triangleq B \vee \mathtt{exn}$
???	Friedman's A-translation of B

Friedman's A-translation modified as a generalised exception monad transformation

$$
\begin{array}{lcl}
\top_\Delta & \triangleq & \top \\
\bot_\Delta & \triangleq & \bot \\
P(\vec{t})_\Delta & \triangleq & P(\vec{t}) \\
(B \wedge C)_\Delta & \triangleq & B_\Delta \wedge C_\Delta \\
(B \vee C)_\Delta & \triangleq & B_\Delta \vee C_\Delta \\
(\exists x\, B(x))_\Delta & \triangleq & \exists x\, B(x)_\Delta \\
(\forall x\, B(x))_\Delta & \triangleq & \forall x\, (B(x)_\Delta \vee \bigvee \Delta) \\
(B \to C)_\Delta & \triangleq & (\forall X\, B_{\Delta,X}) \to (C_\Delta \vee \bigvee \Delta)
\end{array}
$$

Theorem $\Gamma \vdash_\Delta A$ in IQC_{MP} implies $\Gamma_\emptyset \vdash A_\Delta$ in IQC_2

About delimited control

Felleisen's # operators [1988] and Danvy and Filinski's $\langle\ \rangle$ operator [1989] delimit the extent of the evaluation context captured by control operators.

Delimiters also *block* the interaction between a control operator and its surrounding context.

This is what is implicitly used in IQC_{MP}: the interaction of `catch` with its context is blocked so to ensure that the "exception" types in Δ remain "datatypes".

Replacing catch/throw by try/raise

Rules are apparently the same...

$$
\begin{array}{lcl}
(\lambda a.p)\,V & \to & p[V/a] \\
(\lambda x.p)\,t & \to & p[t/x] \\
\texttt{case}\ \iota_i(V)\ \texttt{of}\ [a_1.p_1 \mid a_2.p_2] & \to & p_i[V/a_i] \\
\texttt{dest}\ (t,V)\ \texttt{as}\ (x,a)\ \texttt{in}\ p & \to & p[t/x][V/a] \\
\pi_i(V_1,V_2) & \to & V_i \\
F[\texttt{raise}_E\,p] & \to & \texttt{raise}_E\,p \\
\texttt{try}_E\,\texttt{raise}_E\,p & \to & \texttt{try}_E\,p \\
\texttt{try}_E\,\texttt{raise}_{E'}\,V & \to & \texttt{raise}_{E'}\,V \quad (E \neq E') \\
\texttt{try}_E\,V & \to & V
\end{array}
$$

... except that substitution $p[V/a]$ is no longer *capture-free* (no α-conversion on exception names).

Subject reduction, progress, normalisation, disjunction property and existence property still hold.

catch/throw vs try/raise

For the catch/throw mechanism, bindings are *static* (α-conversion is used to avoid capture)

For the try/raise mechanism, bindings are *dynamic* (no α-conversion)

Example:

$$
\begin{array}{lll}
\mathtt{H}_1 : \exists x\, \top & \triangleq & (x_1, p) \\
\mathtt{H}_2 : \exists x\, \top & \triangleq & (x_2, p) \\
\mathtt{H}_2 : \exists x\, \top \to \exists x\, \top & \triangleq & \lambda a.H_2 \\
\mathtt{G} \ : (((\exists x\, \top \to \exists x\, \top) \to \exists x\, \top) \to \exists x\, \top) \to \exists x\, \top & \triangleq & \lambda F.F(\lambda a.H_2'(F\lambda a'.aH_1)) \\
\mathtt{F}_1 : ((\exists x\, \top \to \exists x\, \top) \to \exists x\, \top) \to \exists x\, \top & \triangleq & \lambda f.\mathtt{catch}_\alpha f(\lambda c.\mathtt{throw}_\alpha c) \\
\mathtt{F}_2 : ((\exists x\, \top \to \exists x\, \top) \to \exists x\, \top) \to \exists x\, \top & \triangleq & \lambda f.\mathtt{try}_E f(\lambda c.\mathtt{throw}_E c)
\end{array}
$$

Then, letting $J_q \triangleq \lambda c.\mathtt{throw}_q c$:

$$
\begin{array}{lll}
GF_1 & \to & \mathtt{catch}_\alpha((\lambda a.H_2'(\mathtt{catch}_\alpha((\lambda a'.aH_1)J_\alpha)))J_\alpha) \\
& = & \mathtt{catch}_\alpha((\lambda a.H_2'(\mathtt{catch}_\beta((\lambda a'.aH_1)J_\beta)))J_\alpha) \\
& \to & \mathtt{catch}_\alpha(H_2'(\mathtt{catch}_\beta\mathtt{throw}_\alpha H_1)) \\
& \to & H_1
\end{array}
\quad \text{while} \quad
\begin{array}{lll}
GF_2 & \to & \mathtt{try}_E((\lambda a.H_2'(\mathtt{try}_E((\lambda a'.aH_1)J \\
& \to & \mathtt{try}_E(H_2'(\mathtt{try}_E\mathtt{throw}_E H_1)) \\
& \to & H_2
\end{array}
$$

Summary

We gave an evidence that Markov's principle *is* undoubtedly constructive and that it has a more clever computational content than just unbounded search.

We are making formal the intuitive connection between Friedman's A-translation and the exception monad transformation.

We are showing that when it turns to $\exists x\, A(x)$ formulae, both `callcc`-style control and `try`-style exception handling are correct realisers (even though they do not have the same computational content).

Arithmetic case to be done formally.

Connections exist with the codereliction rule of differential interaction nets.

Which notion of realisability would capture the expressiveness of IQC_{MP}?