An Intuitionistic Logic that Proves Markov's principle

(an analysis of the computational content of Markov's principle)

Hugo Herbelin

11 July 2010

LICS

Edinburgh

Markov's principle in arithmetic

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

- classically trivial
- entails 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)
- not provable in (standard) intuitionistic logic (no simply-typed realiser, Kreisel [1958])
- preserves the disjunction and existence properties (Smorynski [1973])
- admissible as a rule (Friedman's A-translation [1978], see also Dragalin, generalised by Coquand-Hofmann [1999])
- standard for Russian intuitionism but not considered to be intuitionistic in Brouwer and Bishop

Computing with Markov's principle

Kleene's realisability

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

Gödel's functional interpretation (Dialectica)

$\hookrightarrow$ realisable by identity

Curry-Howard proof-as-program correspondence

$\hookrightarrow$ this work: **Markov's principle = exception mechanism**

More precisely: Markov's principle = statically-bound (as with `callcc`) or dynamically-bound (as with `try`/`with`) exception mechanism with exceptions on *datatypes* only

We focus hereafter on a `catch`/`throw` mechanism for statically-bound exceptions

Preliminary analysis of Friedman's A-translation

Friedman's A-translation: B_A is B in which any atom X (including $\bot$) is replaced by $X \vee A$

$$
\begin{array}{cl}
\vdash_I B & \\
\downarrow & \text{making exceptional calls to ``ex falso quodlibet'' explicit} \\
\vdash_M B_A & \\
\downarrow & \text{moving exceptions up to the surface } (\star) \\
\vdash_M B \vee A & \\
\downarrow & \text{taking } B \text{ for } A \text{ at toplevel} \\
\vdash_M B \vee B & \\
\downarrow & \text{catching the possibility of an exception} \\
\vdash_M B &
\end{array}
$$

Warning! step $(\star)$ only applies when B is intuitionistically equivalent to an $\to$-$\forall$-free formula

$$
\begin{array}{lll}
(B_1 \vee A) \diamond (B_2 \vee A) & \text{iff} \;\; (B_1 \diamond B_2) \vee A & \text{holds for } \vee \text{ and } \wedge \text{ but not } \to \\
\diamond_x (B(x) \vee A) & \text{iff} \;\; (\diamond_x B(x)) \vee A & \text{holds for } \exists \text{ but not for } \forall
\end{array}
$$

(observed, at least, by U. Berger [2004])

This suggests a formulation of Markov's principle dedicated to predicate logic...

Markov's principle in predicate logic

We call Markov's principle for (intuitionistic) predicate logic the principle:

$$\neg\neg T \to T \qquad \text{for } T \text{ strictly positive (i.e. } \to\text{-}\forall\text{-free)}$$

Example: $\exists x\, X(x) \vee \exists y\, Y(y)$ is strictly positive but $\exists x\, (X(x) \to Y(x))$ is not.

Remark: from the point of view of linear/differential logic, this boils down to $?T \to T$ for T strictly positive (an instance of *codereliction*)

... or more generally to $T \to |T|$ where T is strictly positive up to the presence of "?" and $|T|$ erases the "?"

Main results

Intuitionistic logic + classical reasoning limited to strictly positive formulae

- provides with a proof-as-program interpretation of Markov's principle based on exceptions

- using statically-bound exceptions, the proof is

$$\lambda H.\, \mathtt{catch}_k.\mathtt{efq}\ (H\ (\lambda x.\mathtt{throw}\, k\, x))$$

  where $H : \neg\neg T$ and $k : \neg T$

- *internally* satisfies the characteristic disjunction and existence properties of intuitionistic logic

  a proof of $\vdash A \vee B$ comes from a proof either of $\vdash A$ or of $\vdash B$
  a proof of $\vdash \exists x\, A(x)$ comes from a proof of $\vdash A(t)$ for some t

Call-by-name exceptions implement Coquand-Hofmann's generalisation of Friedman's A-translation in *direct style*.

Extended Intuitionistic Predicate Logic: IQC_{MP}

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

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

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

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

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

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

$$\frac{\Gamma \vdash_{\alpha:T,\Delta} p : T}{\Gamma \vdash_\Delta \texttt{catch}_\alpha\, p : T} \qquad \frac{\Gamma \vdash_\Delta p : T \qquad (\alpha : T) \in \Delta}{\Gamma \vdash_\Delta \texttt{throw}_\alpha\, p : C}$$

IQC_{MP} characterises intuitionistic predicate logic + Markov's principle

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

Normalisation rules for IQC_{MP}: at least four possible semantics

call-by-value	*or*	call-by-name
statically-bound exceptions (based on `catch`/`throw`)	*or*	dynamically-bound exceptions (based on `try`/`raise`)

The call-by-value normalisation semantics with static exceptions

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

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

$\forall \rightarrow$-free formulae 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

Note: No rule to capture the context, i.e. catch is used as a degenerated control operator

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 $p \rightarrow q$ 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, for static exceptions, embedding in classical logic)

Internal existence property $\vdash p : \exists x\, A(x)$ implies $\vdash q : A(t)$ with $p \stackrel{*}{\rightarrow} (t, q)$

Internal disjunction property $\vdash p : A_1 \vee A_2$ implies $\vdash q : A_1$ with $p \stackrel{*}{\rightarrow} \iota_1(q)$ or $\vdash q : A_2$ with $p \stackrel{*}{\rightarrow} \iota_2(q)$

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

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

and the evaluation strategy forces the evaluation to

$$(t_n, V)$$

A connection with 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".

In an extended work with Danko Ilik, the full continuation monad is considered and intuitionistic logic with both Markov's principle and double negation shift ($\forall x \neg\neg A(x) \rightarrow \neg\neg\forall x\, A(x)$) is captured

More generally: not only classical logic but any other kind of side-effects could be supported in logic and the logical role of *delimiters* is to purge the effects, something that is possible as soon as the effects are used to ultimately prove a small $\rightarrow$-$\forall$-free formulae.

Summary, ongoing works, remarks

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

The intuitive observation that Friedman's A-translation is a form of exception monad transformation becomes concrete.

Not only `callcc`-style (statically-bound) control but `try`-style (dynamically-bound) exception handling are adequate to prove Markov's principle (even though they do not have the same computational content).

Extension to arithmetic under study.

Design of a notion of Σ-*evasive* modified realisability that validates Markov's principle.

Alternative normalisation proof by embedding to intuitionistic logic using Coquand-Hofmann's generalisation of Friedman's A-translation.

Connections exist with the codereliction rule of differential interaction nets.

Purely intuitionistic proofs of completeness of intuitionistic or classical logic made possible without requiring Veldman-Friedman-Krivine "fallible" ("exploding") models.

A possible alternative to Dialectica for extracting programs from proofs in constructive analysis.

Additional contents

Σ-evasive realisability
(work in progress)

Based on the monadic transformation, we can adapt realisability so that it captures Markov's principle:

$$\begin{array}{lll} p \Vdash_\Delta A & \text{reads as} & p\ \Sigma\text{-evasively realises } A \text{ over } \Delta \\ p \Vdash A & \text{reads as} & p\ \Sigma\text{-evasively realises } A \end{array}$$

$$\begin{array}{lcl}
p \Vdash_\Delta \top & \triangleq & p \text{ is } \star \\
p \Vdash_\Delta A_1 \wedge A_2 & \triangleq & \pi_1(p_1) \Vdash_\Delta A_1 \text{ and } \pi_2(p_2) \Vdash_\Delta A_2 \\
p \Vdash_\Delta A_1 \vee A_2 & \triangleq & \pi_2(p) \Vdash A_{\pi_1(p)} \\
p \Vdash_\Delta A \rightarrow B & \triangleq & \text{for all } \Delta' \supset \Delta,\ q \Vdash_{\Delta'} A \text{ implies either } p\,q \Vdash_{\Delta'} B \text{ or } p\,q \Vdash T \text{ for some } T \text{ in } \Delta' \\
p \Vdash_\Delta \exists x\, A(x) & \triangleq & \pi_2(p) \Vdash_\Delta A(\pi_1(p)) \\
p \Vdash_\Delta \forall x\, A(x) & \triangleq & \text{for all } t \in \mathcal{D}, \text{ either } p\,t \Vdash_\Delta A(t) \text{ or } p\,t \Vdash T \text{ for some } T \text{ in } \Delta
\end{array}$$

(Δ set of strictly positive formulae)

Remark: Independence of premises is validated by modified realisability but no longer validated by Σ-evasive realisability

Replacing catch/throw by try/raise

Rules are apparently the same...

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

... 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.

$\texttt{catch}/\texttt{throw}$ vs $\texttt{try}/\texttt{raise}$

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

For the $\texttt{try}/\texttt{raise}$ mechanism, bindings are *dynamic* (no α-conversion)

Example:

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

Then, letting $J_\alpha \triangleq \lambda c.\texttt{throw}_\alpha c$ and $J_E \triangleq \lambda c.\texttt{raise}_E c$:

$$
\begin{array}{rcl}
GF_1 & \to & \texttt{catch}_\alpha((\lambda a.H'_2(\texttt{catch}_\alpha((\lambda a'.aH_1)J_\alpha)))J_\alpha) \\
& = & \texttt{catch}_\alpha((\lambda a.H'_2(\texttt{catch}_\beta((\lambda a'.aH_1)J_\beta)))J_\alpha) \\
& \to & \texttt{catch}_\alpha(H'_2(\texttt{catch}_\beta\texttt{throw}_\alpha H_1)) \\
& \to & H_1
\end{array}
\quad \text{while} \quad
\begin{array}{rcl}
GF_2 & \to & \texttt{try}_E((\lambda a.H'_2(\texttt{try}_E((\lambda a'.aH_1)J \\
& \to & \texttt{try}_E(H'_2(\texttt{try}_E\texttt{throw}_E H_1)) \\
& \to & H_2
\end{array}
$$

Friedman's A-translation as a generalised monad transformation for call-by-name static exceptions

(work in progress)

$$
\begin{array}{lcl}
\top_\Delta & \triangleq & T_\Delta(\top) \\
\bot_\Delta & \triangleq & T_\Delta(\bot) \\
P(\vec{t})_\Delta & \triangleq & T_\Delta(P(\vec{t})) \\
(B \wedge C)_\Delta & \triangleq & T_\Delta(B_\Delta) \wedge T_\Delta(C_\Delta) \\
(B \vee C)_\Delta & \triangleq & T_\Delta(B_\Delta) \vee T_\Delta(C_\Delta) \\
(\exists x\, B(x))_\Delta & \triangleq & \exists x\, T_\Delta(B(x)_\Delta) \\
(\forall x\, B(x))_\Delta & \triangleq & \forall x\, (T_\Delta(B(x)_\Delta)) \\
(B \to C)_\Delta & \triangleq & \forall \Delta' \supset \Delta.\, T_{\Delta'}(B_{\Delta'}) \to T_{\Delta'}(C_{\Delta'})
\end{array}
$$

for $T_\Delta(B) \triangleq B \vee \Delta$

(based on Coquand-Hofmann A-translation [1999])

Theorem $\Gamma \vdash A$ in IQC_{MP} implies $(\Gamma \vdash A)_\emptyset$ in IQC_2