Proving with delimited control

delimited pure exceptions at ground type = predicate-logic form of Markov's principle $(\neg\neg\exists x\, f(x) = 0 \to \exists x\, f(x) = 0)$

adding delimited control adds Double-Negation Shift (i.e. $(\forall x\, \neg\neg A(x)) \to \neg\neg\forall x\, A(x)$)

Hugo Herbelin

16 February 2012

Logic and Interaction 2012

Proofs and Programs

Explaining the title: Markov's Principle

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

- classically trivial
- entails that classical arithmetic is conservative over intuitionistic logic for Σ^0_1-statements (i.e. $\exists x\, A(x)$ statements with $A(x)$ 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

Explaining the title: 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

$\hookrightarrow$ **Markov's principle = exception mechanism** [H 2010]

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

We focus hereafter on a `try/raise` mechanism of pure exceptions

Explaining the title: The predicate-logic form of Markov's principle

A close examination at Friedman's proof of conservativity (see Berger [2004]) reveals that the proof essentially shows the following syntactic form of Markov's principle:

$$MP \triangleq \neg\neg T \to T$$

for T a Σ-*formula*, i.e. for T *strictly positive* :

$$T ::= X(t_1, ..., t_n) \mid T \vee T \mid T \wedge T \mid \exists x\, T(x)$$

Remarks:

- Strictly positive formulae in predicate logic = Σ^0_1-formula in arithmetic = ground type / first-order data-type in programming
- Key property: closed proofs of strictly positive formulae are fully evaluable into a *value*
- In linear logic terms, Markov's principle reduces to $?T \multimap T$ (i.e. co-dereliction)

Explaining the title: Double-Negation Shift

$$DNS \triangleq (\forall x \neg\neg A(x)) \to \neg\neg \forall x\, A(x)$$

- classically trivial

- is the principle missing to realise the classical version of dependent choice(s)

$$\forall x \neg\neg \exists y\, P(x,y)) \to \forall x_0 \neg\neg \exists f\, (f(0) = x_0 \wedge P(f(n), f(n+1))$$

  from intuitionistic dependent choice

- is the principle missing to prove Glivenko's theorem for predicate logic:

$$\Gamma \vdash_c A \quad \text{iff} \quad DNS, \Gamma \vdash_I \neg\neg A$$

- not provable in (standard) intuitionistic logic (contradicts Church Thesis "CT_0" hence intuitionistic realisability)

- preserves the disjunction and existence properties

- in linear logic terms, Double-Negation Shift amounts to $\&_i ?P_i \multimap ?\&_i P_i$ for P_i an arbitrary large family of (non strictly) positive formulae

Explaining the title: Computing with Double-Negation Shift

Gödel's functional interpretation of arithmetic (Dialectica)

$\hookrightarrow$ DNS realisable using bar recursion [Spector 1962]

$\hookrightarrow$ studied in terms of product of selection functions [Escardó-Oliva 2010]

Kleene's realisability $\perp$

$\hookrightarrow$ not realisable if $\perp$ not realisable... but probably realisable using realisability combined with Friedman's A-translation

Curry-Howard proof-as-program correspondence

$\hookrightarrow$ $\mathbf{MP + DNS = exceptions + control\ delimited\ at\ ground\ type}$ [Ilik 2010]

More precisely: MP + DNS obtainable from control delimited at ground type and exceptions that can be either statically-scoped or dynamically-scoped. In the first case, exceptions is a subset of static control and the slogan holds.

Explaining the title: Delimited pure exceptions

We consider a class of exception names $\hat{\alpha}$, $\hat{\beta}$ (roughly speaking, these are dynamically variable names for evaluation contexts)

pure catching of a declared exception ↓

Terms $p ::= a \mid \lambda a.p \mid p\,p \mid \mathtt{try}_{\mathtt{new}\,\hat{\alpha}}\,p \mid \mathtt{try}_{\hat{\alpha}}\,p \mid \mathtt{raise}_{\hat{\alpha}}\,p$

↑ *declare and ensure pure catching of a new exception* ↑ *raise an exception*

Pure exceptions vs ML exceptions:

$$\begin{array}{ll}
\mathtt{raise}_{\hat{\alpha}}\,p & \triangleq\ \mathtt{raise}^{ML}\ (\hat{\alpha}\ p) \\
\mathtt{try}_{\hat{\alpha}}\,p & \triangleq\ \mathtt{try}^{ML}\ p\ \mathtt{with}\ \hat{\alpha}\ x\ \rightarrow\ x \\
 & \\
\mathtt{raise}^{ML} p & \triangleq\ \mathtt{raise}_{\hat{\alpha}_0}\ p \\
\mathtt{try}^{ML}\ p\ \mathtt{with}\ \textit{patterns} & \triangleq\ \mathtt{case}\ \mathtt{try}_{\hat{\alpha}_0}\ (\mathtt{Val\,p})\ \mathtt{of} \\
 & \quad \mid \mathtt{Val}\,x \rightarrow x \\
 & \quad \mid \textit{patterns}
\end{array}$$

Explaining the title: Delimited pure exceptions

The semantics of pure exceptions:

$$
\begin{array}{lll}
\mathtt{try}_{\hat{\alpha}} V & \to V & \\
\mathtt{try}_{\hat{\alpha}} \mathtt{raise}_{\hat{\alpha}} V & \to V & \\
\mathtt{try}_{\hat{\beta}} \mathtt{raise}_{\hat{\alpha}}\, V & \to \mathtt{raise}_{\hat{\alpha}}\, V & \hat{\alpha} \neq \hat{\beta} \\
\mathtt{try}_{\mathtt{new}\,\hat{\alpha}} V & \to V & \\
\mathtt{try}_{\mathtt{new}\,\hat{\alpha}} \mathtt{raise}_{\hat{\alpha}} V & \to V & \\
\mathtt{try}_{\mathtt{new}\,\hat{\beta}} \mathtt{raise}_{\hat{\alpha}}\, V & \to \mathtt{raise}_{\hat{\alpha}}\, V & \\
F[\mathtt{raise}_{\hat{\alpha}}\, V] & \to \mathtt{raise}_{\hat{\alpha}}\, V &
\end{array}
$$

where

Elem. ev. cont. $F ::= \ldots \mid \mathtt{raise}_{\hat{\beta}}\,[\,]$

Values $V ::= x \mid \lambda a.p \mid \lambda x.p \mid \iota_i(V) \mid (V, V) \mid (t, V)$

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:

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

Then, letting $J_\alpha \triangleq \lambda c.\mathtt{throw}_\alpha c$ and $J_{\hat\alpha} \triangleq \lambda c.\mathtt{raise}_{\hat\alpha} c$:

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

where GF_2 has to occur in the scope of a $\mathtt{try}_{\mathtt{new}\,\hat\alpha}$ to be well-typed

Explaining the title: Delimited pure exceptions at ground type

T is a ground type (i.e. a Σ-formula)

$$\frac{\Gamma, \hat{\alpha}{:}T^{\perp\!\!\!\perp} \vdash p{:}T}{\Gamma \vdash \mathtt{try}_{\mathtt{new}\,\hat{\alpha}}\, p{:}T}\text{TRY-NEW} \qquad \frac{\Gamma \vdash p{:}T \qquad (\hat{\alpha}{:}T^{\perp\!\!\!\perp}) \in \Gamma}{\Gamma \vdash \mathtt{try}_{\hat{\alpha}}\, p{:}T}\text{TRY}$$

$$\frac{\Gamma \vdash p{:}T \qquad (\hat{\alpha}{:}T^{\perp\!\!\!\perp}) \in \Gamma}{\Gamma \vdash \mathtt{raise}_{\hat{\alpha}}\, p{:}C}\text{RAISE}$$

Logically speaking, TRY is useless. However, it is needed to preserve typing while evaluating exceptions (because of dynamic binding).

Alternatively, for the strict purpose of logical expressiveness, exceptions could have been considered *statically-bound*, as if `try` had behaved like `callcc` and `raise` like `throw`. In this latter case, TRY is useless.

Explaining the title: control delimited at ground type

↗ *this is what ensures delimitation at ground type*

$$\frac{\Gamma, \alpha : A^{\perp\!\!\!\perp} \vdash p : A \qquad (\hat{\alpha} : T^{\perp\!\!\!\perp}) \in \Gamma \text{ for some } T}{\Gamma \vdash \mathtt{catch}_\alpha\, p : A}\,\text{CATCH}$$

$$\frac{\Gamma \vdash p : A \qquad (\alpha : A^{\perp\!\!\!\perp}) \in \Gamma}{\Gamma \vdash \mathtt{throw}_\alpha\, p : B}\,\text{THROW}$$

In terms of $\lambda\mu$: $\mathtt{catch}_\alpha\, p$ behaves the same as $\mu\alpha.[\alpha]p$ and $\mathtt{throw}_\alpha\, p$ the same as $\mu\_.[\alpha]p$

$$\begin{array}{lcll}
F[\mathtt{catch}_\alpha p] & \to & \mathtt{catch}_\alpha p[\alpha \leftarrow [\alpha]F] & \\
F[\mathtt{throw}_\alpha p] & \to & \mathtt{throw}_\alpha p & \\
\mathtt{catch}_\alpha \mathtt{throw}_\alpha\, p & \to & \mathtt{catch}_\alpha p & \\
\mathtt{catch}_\beta \mathtt{catch}_\alpha\, p & \to & \mathtt{catch}_\beta \mathtt{throw}_\alpha\, p[\alpha \leftarrow \beta] & \text{even if } \alpha = \beta \\
\mathtt{throw}_\beta \mathtt{throw}_\alpha\, p & \to & \mathtt{throw}_\alpha p & \text{even if } \alpha = \beta \\
\mathtt{try}_{\hat{\alpha}} \mathtt{catch}_{\hat{\alpha}} p & \to & \mathtt{try}_{\hat{\alpha}} p[\alpha \leftarrow [\hat{\alpha}]p] & \\
\mathtt{try}_{\mathtt{new}\,\hat{\alpha}} \mathtt{catch}_{\hat{\alpha}} p & \to & \mathtt{try}_{\mathtt{new}\,\hat{\alpha}} p[\alpha \leftarrow [\hat{\alpha}]p] &
\end{array}$$

where substitution is extended with

$$\mathtt{throw}_\alpha\, q[\alpha \leftarrow [\hat{\alpha}]F] \triangleq \mathtt{raise}_{\hat{\alpha}} F[q[\alpha \leftarrow [\hat{\alpha}]F]]$$

Proving with delimited control

delimited pure exceptions at ground type = predicate-logic form of Markov's principle
$(\neg\neg\exists x\, f(x) = 0 \to \exists x\, f(x) = 0)$

adding delimited control adds Double-Negation Shift (i.e. $(\forall x\, \neg\neg A(x)) \to \neg\neg\forall x\, A(x)$)

Explaining the title

Proving with delimited control

delimited pure exceptions at ground type = predicate-logic form of Markov's principle $(\neg\neg T \to T)$

adding delimited control adds Double-Negation Shift (i.e. $(\forall x\, \neg\neg A(x)) \to \neg\neg \forall x\, A(x)$)

Proving with delimited control

delimited pure exceptions at ground type = predicate-logic form of Markov's principle $(\neg\neg T \to T)$

adding delimited control adds Double-Negation Shift (i.e. $(\forall x\, \neg\neg A(x)) \to \neg\neg \forall x\, A(x)$)

... but it would have worked also with statically-scoped exceptions (i.e. just `callcc`-style control)

... the reason we consider dynamic exceptions is that they will be necessary for adding memory updates (tomorrow)

Explaining the title

Intuitionistically proving with delimited control

delimited pure exceptions at ground type = predicate-logic form of Markov's principle $(\neg\neg T \to T)$

adding delimited control adds Double-Negation Shift (i.e. $(\forall x\, \neg\neg A(x)) \to \neg\neg \forall x\, A(x)$)

... in the sense that the disjunction and existence properties are preserved

Some general thoughts

Does there exist non-intuitionistic logics?

Functional programming learns us that control is one effect among many others. This suggests the following:

Logic is essentially intuitionistic; any side effect can freely be used and eventually cleared as far as what is eventually proved is a Σ-formula (note that $\bot$, i.e. consistency, is one of them)

And also:

Any ideal object (oracle "deciding" undecidable propositions, non-recursive limit branches in trees, injection from $\aleph_2$ to $2^{\aleph_0}$, ...) provable using effects (classical logic, forcing) can freely and conservatively be used in any proof of a Σ-formula without breaking the intuitionistic character of the underlying logic

But linear logic learns us also to see intuitionistic logic as a co-monadic effect!

$$\text{intuitionistic logic} \quad = \quad \text{co-Kleisli}_{!} \text{ (linear logic)}$$

$$\text{i.e.} \qquad \Gamma \vdash_I A \qquad \text{iff} \qquad !\Gamma \vdash_L A$$

$$\text{where } A \to B \qquad = \qquad !A \multimap B$$

What about forcing?

Forcing can be seen as a comonadic modal effect

$$\Box_x A \quad \triangleq \quad \forall x' \geq x \, A[x \leftarrow x']$$

and we will see tomorrow how to build a "co-Kleisli logic" out of it using monotone memory updates.

Otherwise said, stronger and stronger logics, including forcing, seem just to reduce to applying stronger and stronger effects and co-effects on top of a (pre-)linear logic...

Inner models vs forcing models

When it turns to the properties of the domain of discourse, there are two classes of models:

- *Inner models*, i.e. relativisation, restrict the opponent world
  - Church's Thesis: functions can only be recursive
  - Constructible sets: only "syntactic" sets exist
  - ...
- *"(co)monadic" models* extend the proponent world
  - Classical logic: asserts the existence of "limit" objects
  - Intuitionistic forcing models: allow to define functions/predicates using memory effects
  - Classical forcing models: assert the existence of functions/predicates defined using memory effects
  - ...

Outline

- $IQC_{MP_{dyn}}$: An intuitionistic logic for reasoning in direct-style about Friedman's A-translation seen as an exception monad on top of usual intuitionistic logic

- $IQC_{MP_{dyn}-DNS}$: An intuitionistic logic for reasoning in direct-style about the classical variant of Friedman's A-translation seen as an double-negated exception monad on top of usual intuitionistic logic

For IQC_{MP}, logic with static exceptions, see H [2010], even though the logical expressiveness is the same

$IQC_{MP_{dyn}}$: "Markov's principle" Intuitionistic Predicate Logic

Proofs of $IQC_{MP_{dyn}}$ are defined by:

$$\begin{array}{rcl} p, q & ::= & a \mid \iota_i(p) \mid (p, q) \mid (t, p) \mid \lambda a.p \mid \lambda x.p \mid () \\ & \mid & \texttt{case}\ p\ \texttt{of}\ [a_1.p_1 \mid a_2.p_2] \\ & \mid & \pi_i(p) \mid \texttt{dest}\ p\ \texttt{as}\ (x, a)\ \texttt{in}\ q \\ & \mid & p\,q \mid p\,t \mid \texttt{exfalso}\ p \\ & \mid & \texttt{try}_{\texttt{new}\,\hat{\alpha}}\,p \mid \texttt{try}_{\hat{\alpha}}\,p \mid \texttt{raise}_{\hat{\alpha}}\,p \end{array}$$

with inference rules given as before

$IQC_{MP_{dyn}}$ characterises intuitionistic predicate logic + Markov's principle

$$\vdash_{IQC_{MP_{dyn}}} A \quad \text{iff} \quad MP \vdash_I A$$

The right-to-left direction is easy:

$$\begin{aligned} \mathtt{mp} \;&:\; \neg\neg T \to T \\ &\triangleq\; \lambda H.\, \mathtt{try}_{\mathtt{new}\,\hat{\alpha}}.\mathtt{exfalso}\ (H\,(\lambda x.\mathtt{raise}_{\hat{\alpha}}\,x)) \end{aligned}$$

$IQC_{MP_{dyn}}$ characterises intuitionistic predicate logic + Markov's principle

For the left-to-right direction, there is also an easy proof.

If we turn $T^{\perp\!\!\!\perp}$ into $\neg T$ in Γ:

$$\begin{array}{lcl} \epsilon^+ & \triangleq & \epsilon \\ (\Gamma, T^{\perp\!\!\!\perp})^+ & \triangleq & \Gamma^+, \neg T \\ (\Gamma, A)^+ & \triangleq & \Gamma^+, A \end{array}$$

Then, using MP for the rule TRY-NEW, we get:

$$\Gamma \vdash_{IQC_{MP_{dyn}}} A \quad \text{implies} \quad MP, \Gamma^+ \vdash_I A$$

from which

$$\vdash_{IQC_{MP_{dyn}}} A \quad \text{implies} \quad MP \vdash_I A$$

directly follows

But, wait a minute, this proof will fail to show that reductions are preserved by the translation.

$IQC_{MP_{dyn}-DNS}$ characterises intuitionistic predicate logic + Markov's principle

Let us do a second proof for the left-to-right direction and introduce the exception monad

$$Exc_T\, A \triangleq (A \to T) \to T$$

The Exc_T monad commutes with the positive connectives:

$$\begin{array}{lcl} (Exc_T\, A) \vee (Exc_T\, B) & \to & Exc_T\, (A \vee B) \\ (Exc_T\, A) \wedge (Exc_T\, B) & \to & Exc_T\, (A \wedge B) \\ \exists x\, (Exc_T\, A) & \to & Exc_T\, (\exists x\, A) \end{array}$$

But it does not commute with the negative connectives!

However, if $\neg T$ holds, then it commutes with $\to$ and $\forall$.

In particular, if we had only one use of $\mathrm{TRY\text{-}NEW}$ and Γ be otherwise effect-free (apart from some $A^{\perp\!\!\!\perp}$ turned into $\neg A$), by induction on the derivation, we would have:

$$T^{\perp\!\!\!\perp}, \Gamma \vdash_{IQC_{MP_{dyn}}} A \quad \text{implies} \quad \neg T, \Gamma \vdash_I Exc_T A$$

Addressing the nesting of effects

We have nested effects, hence we need to nest the monad:

$$\begin{array}{lcl} Exc_{\epsilon}\, A & \triangleq & A \\ Exc_{(\Gamma, T^{\perp\!\!\!\perp})}\, A & \triangleq & Exc_{Exc_{\Gamma} T}\, Exc_{\Gamma}\, A \\ Exc_{(\Gamma, B)}\, A & \triangleq & Exc_{\Gamma}\, A \\ Exc_{(\Gamma, B^{\perp\!\!\!\perp})}\, A & \triangleq & Exc_{\Gamma}\, A \end{array}$$

For instance, in context $\Gamma \triangleq a : A, \hat{\alpha} : T, b : B, \hat{\beta} : U$:

$$Exc_{\Gamma} D \quad \triangleq \quad (D \vee (U \vee T)) \vee T$$

(translation has redundancies but this will make the proof more uniform!)

Now, Γ can declare several effects, so we define:

$$\begin{array}{lcl} \epsilon^{+} & \triangleq & \epsilon \\ (\Gamma, T^{\perp\!\!\!\perp})^{+} & \triangleq & \Gamma^{+}, \neg T \\ (\Gamma, B)^{+} & \triangleq & \Gamma^{+}, B \end{array}$$

Last step...

Using MP for the rule TRY-NEW and injection in Exc for the axiom rule, we have the expected result:

$$\Gamma \vdash_{IQC_{MP_{dyn}}} A \quad \text{implies} \quad MP, \Gamma^{+} \vdash_{I} Exc_{\Gamma} A$$

which directly gives

$$\vdash_{IQC_{MP_{dyn}}} A \quad \text{implies} \quad MP \vdash_{I} A$$

$IQC_{MP_{dyn}-DNS}$: "Markov's principle" + "Double-Negation Shift" Intuitionistic Predicate Logic

Proofs of $IQC_{MP_{dyn}-DNS}$ are defined by:

$$\begin{array}{rcl} p, q & ::= & a \mid \iota_i(p) \mid (p, q) \mid (t, p) \mid \lambda a.p \mid \lambda x.p \mid () \\ & \mid & \mathtt{case}\ p\ \mathtt{of}\ [a_1.p_1 \mid a_2.p_2] \\ & \mid & \pi_i(p) \mid \mathtt{dest}\ p\ \mathtt{as}\ (x, a)\ \mathtt{in}\ q \\ & \mid & p\,q \mid p\,t \mid \mathtt{exfalso}\ p \\ & \mid & \mathtt{try}_{\mathtt{new}\,\hat{\alpha}}\,p \mid \mathtt{try}_{\hat{\alpha}}\,p \mid \mathtt{raise}_{\hat{\alpha}}\,p \\ & \mid & \mathtt{catch}_{\alpha} p \mid \mathtt{throw}_{\alpha} p \end{array}$$

with inference rules given as before.

$IQC_{MP_{dyn}-DNS}$ characterises intuitionistic predicate logic + Markov's principle + Double-Negation Shift

$$\vdash_{IQC_{MP_{dyn}-DNS}} A \quad \text{iff} \quad MP, DNS \vdash_I A$$

The right-to-left direction is again easy:

$$\begin{aligned} \texttt{dns} \;:\; & \forall x\, \neg\neg A(x) \to \neg\neg \forall x\, A(x) \\ \triangleq\; & \lambda H.\lambda k.\, \texttt{try}_{\texttt{new}\,\hat{\alpha}}.(k\, \lambda x.\texttt{catch}_{\alpha}.(H\, x\, (\lambda a.\texttt{throw}_{\alpha}\, a))) \end{aligned}$$

where

$k : \neg\forall x\, A(x)$

$\hat{\alpha} : \bot^{\perp\!\!\!\perp}$ (whose sole purpose is to justify the `catch`)

$\alpha : A(x)^{\perp\!\!\!\perp}$

$a : A(x)$

$IQC_{MP_{dyn}-DNS}$ characterises intuitionistic predicate logic + Markov's principle + Double-Negation Shift

For the other direction, we could have considered the continuation monad but, so as to preserve the reduction semantics, we introduce instead the double negation of the exception monad:

$$2Exc_T\, A \triangleq Exc_T \neg_{Exc_T \perp} (Exc_T \neg_{Exc_T \perp} A)$$

The $2Exc_T$ monad commutes with the positive connectives:

$$\begin{array}{lcl} (2Exc_T\, A) \vee (2Exc_T\, B) & \to & 2Exc_T\, (A \vee B) \\ (2Exc_T\, A) \wedge (2Exc_T\, B) & \to & 2Exc_T\, (A \wedge B) \\ \exists x\, (2Exc_T\, A) & \to & 2Exc_T\, (\exists x\, A) \end{array}$$

But it does not commute with the negative connectives!

However, if $\neg T$ holds, then it commutes with $\to$.

And if in addition DNS holds, then it commutes with $\forall$.

In particular, if we had only one use of TRY-NEW and Γ be otherwise effect-free (apart from some $A^{\perp\!\!\!\perp}$ turned into $\neg A$), by induction on the derivation, we would have:

$$T^{\perp\!\!\!\perp}, \Gamma \vdash_{IQC_{MP_{dyn}}} A \quad \text{implies} \quad \neg T, \Gamma \vdash_I 2Exc_T A$$

Addressing the nesting of effects

We have nested effects, hence we need to nest the monad:

$$
\begin{array}{lcl}
2Exc_{\epsilon}\, A & \triangleq & A \\
2Exc_{(\Gamma, T^{\perp\!\!\!\perp})}\, A & \triangleq & 2Exc_{2Exc_{\Gamma} T}\, 2Exc_{\Gamma}\, A \\
2Exc_{(\Gamma, B)}\, A & \triangleq & 2Exc_{\Gamma}\, A \\
2Exc_{(\Gamma, B^{\perp\!\!\!\perp})}\, A & \triangleq & 2Exc_{\Gamma}\, A
\end{array}
$$

(translation is complex but will make the proof more uniform!)

Now, Γ can declare several effects, so we define:

$$
\begin{array}{lcl}
\epsilon^{+} & \triangleq & \epsilon \\
(\Gamma, T^{\perp\!\!\!\perp})^{+} & \triangleq & \Gamma^{+}, \neg T \\
(\Gamma, B)^{+} & \triangleq & \Gamma^{+}, B \\
(\Gamma, B^{\perp\!\!\!\perp})^{+} & \triangleq & \Gamma^{+}, 2Exc_{\Gamma} \neg_{Exc_{\Gamma} \perp} B
\end{array}
$$

Last step...

Using again MP for the rule TRY-NEW, DNS to have $2Exc$ traverse $\forall$, and injection in $2Exc$ for the axiom rule, we have the expected result:

$$\Gamma \vdash_{IQC_{MP_{dyn}}} A \quad \text{implies} \quad MP, DNS, \Gamma^{+} \vdash_{I} 2Exc_{\Gamma} A$$

Properties of the reduction systems

The resulting reduction systems are rich enough to ensure the normalisation of closed proofs. For both $IQC_{MP_{dyn}}$ and $IQC_{MP_{dyn}-DNS}$ we have:

Subject reduction If $\Gamma \vdash p : A$ and $p \to q$ then $\Gamma \vdash q : A$

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

Normalisation If $\Gamma \vdash p : A$ then p is normalisable (by embedding into ordinary intuitionistic logic using Coquand-Hofmann's translation on top of the Exc and $2Exc$ monadic translations)

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

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

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.

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).

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

For Markov's principle, 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

Markov's principle: 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

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

and the evaluation strategy forces the evaluation to

$$(t_n, V)$$

Σ-evasive realisability

(work in progress)

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

$p \Vdash_\Delta A$ reads as p Σ-evasively realises A over Δ
$p \Vdash A$ reads as p Σ-evasively realises A

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

(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...

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

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

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

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

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

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

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

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