Computational interpretation of proofs: Classical realizability and forcing

Alexandre Miquel

UNIVERSIDAD DE LA REPUBLICA URUGUAY
FACULTAD DE INGENIERIA
EQUIPO DE LÓGICA UDELAR

Semantics of proofs and certified mathematics
PhD school – April 11th, 2014 – CIRM – Luminy

Different notions of models

- **Tarski models:** $[\![A]\!] \in \{0;1\}$
  - Interprets classical provability (correctness/completeness)
- **Intuitionistic realizability:** $[\![A]\!] \in \mathfrak{P}(\Lambda)$ [Kleene 45]
  - Interprets intuitionistic proofs
  - Theoretical basis of intuitionistic program extraction
  - Independence results, in intuitionistic theories
  - Definitely incompatible with classical logic
- **Cohen forcing:** $[\![A]\!] \in \mathfrak{P}(C)$ [Cohen 63]
  - Independence results, in classical theories
    (Negation of continuum hypothesis, Solovay's axiom, etc.)
- **Classical realizability** $[\![A]\!] \in \mathfrak{P}(\Lambda_c)$ [Krivine 94]
  - Interprets classical proofs
  - Generalizes Tarski models... and forcing!

Plan

Plan

What is forcing?

- A technique invented by Paul Cohen ('63) to prove the independence of the continuum hypothesis (CH) w.r.t. ZFC

The continuum hypothesis (CH), Hilbert's 1st problem

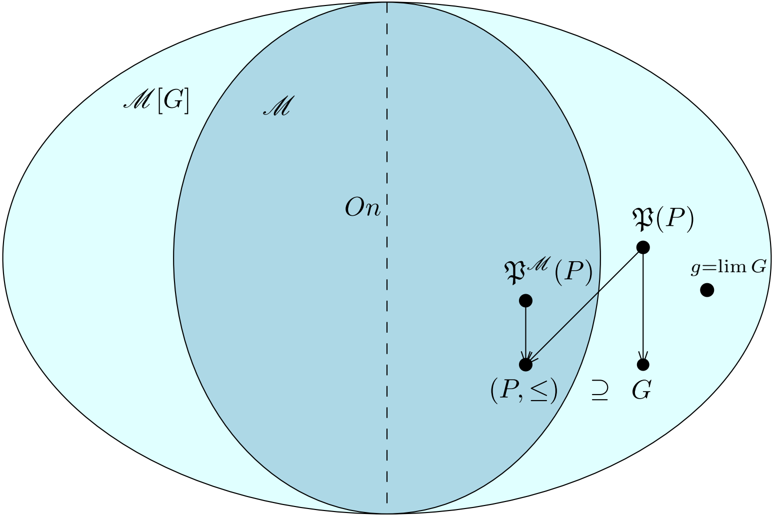
For every infinite subset $S \subseteq \mathbb{R}$:

- Either S is denumerable (i.e. in bijection with $\mathbb{N}$)
- Either S has the power of continuum (i.e. is in bijection with $\mathbb{R}$)

In symbols: $2^{\aleph_0} = \aleph_1$

- Gödel ('38) proved $\text{ZFC} \nvdash \neg\text{CH}$ introducing constructible sets
- Cohen ('63) proved $\text{ZFC} \nvdash \text{CH}$ introducing forcing
- Related to Boolean-valued models [Scott, Solovay, Vopěnka]
- Used to prove the consistency/independence of many axioms [Solovay, Shelah, Woodin, etc.]

How does forcing work?



An analogy with algebra

Set theory	**Algebra**
Start from a ground model $\mathscr{M}$	Start from a ground field F
We want to add a new set approximated by the elements of a given forcing poset $(P, \leq) \in \mathscr{M}$	We want to add a new point that should be a root of a given polynomial $P \in F[X]$
This defines a fictitious generic filter $G \subseteq P$ (outside $\mathscr{M}$)	This defines a fictitious root α of P (outside F)
which generates around $\mathscr{M}$ a generic extension $\mathscr{M}[G]$	which generates around F a field extension $F[\alpha]$
Construction: $\mathscr{M}[G] \ := \ \mathscr{M}^{[P]}/{\sim_{\mathrm{Ext}}}$	Construction: $F[\alpha] \ := \ F[X]/PF[X]$

Example: forcing $\neg$CH

- **Aim:** Force the existence of an injection $h : \aleph_2 \to \mathfrak{P}(\omega)$
  We shall build it as a characteristic function $g : \aleph_2 \times \omega \to 2$
- The ideal object g is approximated in the ground model $\mathscr{M}$ by elements of $(P, \leq) = (\mathrm{Fin}(\aleph_2 \times \omega, 2), \supseteq)$ (forcing poset)
- **Forcing invocation:** Let $\mathscr{M}[G]$ be the generic extension generated by an $\mathscr{M}$-generic filter $G \subseteq P$ (always exists!)
- In $\mathscr{M}[G]$, we let: $g = \lim G = \bigcup G \quad (: \aleph_2 \times \omega \rightharpoonup 2)$
  Using the $\mathscr{M}$-genericity of the filter $G \subseteq P$, we prove that:
  - Partial function $g : \aleph_2 \times \omega \to 2$ is actually total
  - Corresponding function $h : \aleph_2 \to \mathfrak{P}(\omega)$ is injective

Technicalities (countable chain condition) under the carpet

Compared properties of $\mathscr{M}$ and $\mathscr{M}[G]$

Forcing theorem: Given a model $\mathscr{M}$ and a forcing poset $(P, \leq) \in \mathscr{M}$, the generic extension $\mathscr{M}[G]$ always exists

- $\mathscr{M}$ and $\mathscr{M}[G]$ have the very same ordinals
- If Axiom of Choice (AC) holds in $\mathscr{M}$, then it holds in $\mathscr{M}[G]$ too
- Finite cardinals and $\aleph_0 = \omega$ are the same in $\mathscr{M}$ and $\mathscr{M}[G]$
- $\mathscr{M}[G]$ has in general fewer cardinals than $\mathscr{M}$
  - **Intuition:** new bijections may appear in $\mathscr{M}[G]$ between sets in $\mathscr{M}$, thus identifying their cardinals in $\mathscr{M}[G]$
  - Cardinals are preserved if P fulfils the countable chain condition (This was the case for $P = \mathrm{Fin}(E, 2)$ for forcing $\neg\mathrm{CH}$)
  - But in some circumstances, one may use forcing to kill cardinals: Levy collapse, Solovay's axiom, etc.

The proof-theoretic point of view

- Construction of $\mathscr{M}[G]$ parameterized by a forcing poset $(P, \leq)$, whose elements are called forcing conditions
  - $p \leq q$ reads: 'p is stronger than q'
- Internally relies on a logical translation

$$A \quad \mapsto \quad p \Vdash A \qquad \text{('}p\text{ forces }A\text{')}$$

where p is a fresh variable (representing a condition)

  - Complex definition by induction on A, using the poset $(P, \leq)$

Properties

1. $\vdash A$ entails $\vdash (\forall p \in P)\,(p \Vdash A)$
2. But $\vdash (\forall p \in P)\,(p \Vdash A)$ for more formulas A (depending on P)
3. $\vdash (\forall p \in P)\,(p \nVdash \bot)$ (consistency)

- **Remark:** Forcing commutes with $\bot$, $\top$, $\wedge$ and $\forall$, but not with $\Rightarrow$, $\neg$, $\vee$, $\exists$

Kripke forcing versus Cohen forcing

Kripke models for (classical) modal logic (S4)

$$p \Vdash A \Rightarrow B \quad \equiv \quad (p \Vdash A) \Rightarrow (p \Vdash B)$$

$$p \Vdash \Box A \quad \equiv \quad \forall q \le p \; (q \Vdash A)$$

$$\frac{p \Vdash A \Rightarrow B \qquad p \Vdash A}{p \Vdash B}$$

⇑

Gödel's translation from LJ to S4

⇓

$$(A \Rightarrow B)^{\dagger} \equiv \Box(A^{\dagger} \Rightarrow B^{\dagger})$$

Kripke models for intuitionistic logic (LJ)

$$p \Vdash A \Rightarrow B \quad \equiv \quad \forall q \le p \; ((q \Vdash A) \Rightarrow (q \Vdash B))$$

$$\frac{p \Vdash A \Rightarrow B \qquad q \Vdash A}{q \Vdash B}\; q \le p$$

⇑

$\neg\neg$-translation from LK to LJ

⇓

(tricky!)

Forcing in classical logic (LK)

$$p \Vdash A \Rightarrow B \quad \equiv \quad \forall q \; ((q \Vdash A) \Rightarrow \forall r \le p, q \; (r \Vdash B))$$

$$\frac{p \Vdash A \Rightarrow B \qquad q \Vdash A}{r \Vdash B}\; r \le p, q$$

Cohen forcing versus classical realizability

Cohen forcing	Classical realizability
$[\![A]\!] \in \mathfrak{P}(C)$	$\vert A\vert \in \mathfrak{P}(\Lambda_c)$
$p \Vdash A$	$t \Vdash A$
$\dfrac{p \Vdash A \Rightarrow B \qquad q \Vdash A}{\underbrace{pq}_{\text{g.l.b.}} \Vdash B}$	$\dfrac{t \Vdash A \Rightarrow B \qquad u \Vdash A}{\underbrace{tu}_{\text{application}} \Vdash B}$
$\dfrac{p \Vdash A \qquad q \Vdash B}{pq \Vdash A \wedge B}$	$\dfrac{t \Vdash A \qquad u \Vdash B}{\langle t; u\rangle \Vdash A \wedge B}$
$A \wedge B \ = \ A \cap B$	$A \wedge B \ \neq \ A \cap B$

- **Slogan:** Classical realizability $=$ Non commutative forcing

Combining Cohen forcing with classical realizability

- **Forcing in classical realizability** [Krivine 09]
  - Introduce realizability algebras, generalizing the λ_c-calculus
  - Discover the program transformation underlying forcing
  - Extend iterated forcing to classical realizability
  - Show how to force the existence of a well-ordering over $\mathbb{R}$ (while keeping evaluation deterministic)

- **Computational analysis of forcing** [Miquel 11]
  - Focus on the underlying program transformation (no generic filter)
  - Hard-wire the program transformation into the abstract machine

Underlying methodology

Translation of formulas & proofs $\rightsquigarrow$ Classical program transformation $\rightsquigarrow$ New abstract machine (no transformation)

Plan

Higher-order arithmetic (PAω^+)

- A multi-sorted language that allows to express
  - Individuals (kind ι)
  - Propositions (kind o)
  - Functions over individuals ($\iota \to \iota, \quad \iota \to \iota \to \iota, \quad \ldots$)
  - Predicates over individuals ($\iota \to o, \quad \iota \to \iota \to o, \quad \ldots$)
  - Predicates over predicates... ($(\iota \to o) \to o, \quad \ldots$)

Syntax of kinds and higher-order terms

Kinds $\quad \tau, \sigma \ ::= \ \iota \ | \ o \ | \ \tau \to \sigma$

Terms $\quad M, N, A, B \ ::= \ x^\tau \ | \ \lambda x^\tau . M \ | \ MN \ | \ 0 \ | \ \mathsf{s} \ | \ \mathrm{rec}_\tau$
$\quad | \ A \Rightarrow B \ | \ \forall x^\tau A \ | \ M = M' \mapsto A$

- Equational implication: $M = M' \mapsto A$
  - Means: A if $M = M'$ (equality of denotations)
    $\top$ otherwise ($\top$ = type of all proofs)
  - Provably equivalent to: $M =_\tau M' \Rightarrow A$ (Leibniz equality)

Conversion (1/2)

- Conversion $M \cong_{\mathcal{E}} M'$ parameterized by a (finite) set of equations

$$\mathcal{E} \equiv M_1 = M'_1, \ldots, M_k = M'_k \qquad \text{(non oriented, well `kinded')}$$

- Reflexivity, symmetry, transitivity + base case:

$$\frac{}{M \cong_{\mathcal{E}} M'}\ (M{=}M')\in\mathcal{E}$$

- β-conversion, recursion:

$$\begin{aligned} (\lambda x^{\tau} . M)N &\cong_{\mathcal{E}} M\{x := N\} \\ \mathsf{rec}_{\tau}\ M\ M'\ 0 &\cong_{\mathcal{E}} M \\ \mathsf{rec}_{\tau}\ M\ M'\ (\mathsf{s}\ N) &\cong_{\mathcal{E}} M'\ N\ (\mathsf{rec}_{\tau}\ M\ M'\ N) \end{aligned}$$

- Usual context rules + extended rule for $M = M' \mapsto A$:

$$\frac{A \cong_{\mathcal{E}, M=M'} A'}{M = M' \mapsto A \cong_{\mathcal{E}} M = M' \mapsto A'}$$

Conversion (2/2)

- Rules for identifying computationally equivalent propositions:

$$
\begin{array}{rcll}
\forall x^\tau\, \forall y^\sigma\, A & \cong_{\mathcal{E}} & \forall y^\sigma\, \forall x^\tau\, A & \\
\forall x^\tau\, A & \cong_{\mathcal{E}} & A & x^\tau \notin FV(A) \\
A \Rightarrow \forall x^\tau\, B & \cong_{\mathcal{E}} & \forall x^\tau\, (A \Rightarrow B) & x^\tau \notin FV(A) \\
M = M' \mapsto N = N' \mapsto A & \cong_{\mathcal{E}} & N = N' \mapsto M = M' \mapsto A & \\
M = M \mapsto A & \cong_{\mathcal{E}} & A & \\
A \Rightarrow (M = M' \mapsto B) & \cong_{\mathcal{E}} & M = M' \mapsto (A \Rightarrow B) & \\
\forall x^\tau (M = M' \mapsto A) & \cong_{\mathcal{E}} & M = M' \mapsto \forall x^\tau A & x^\tau \notin FV(M,M')
\end{array}
$$

- Example: $\top \;:=\; \mathrm{tt} = \mathrm{ff} \mapsto \bot$ (type of all proof-terms)

  where $\mathrm{tt} \equiv \lambda x^o y^o \,.\, x$, $\mathrm{ff} \equiv \lambda x^o y^o \,.\, y$ and $\bot \equiv \forall z^o\, z$

Deduction system (typing)

- Proof terms: $t, u \quad ::= \quad x \;\mid\; \lambda x\,.\,t \;\mid\; tu \;\mid\; \text{œ}$ (Curry-style)
- Contexts: $\Gamma \quad ::= \quad x_1 : A_1, \ldots, x_n : A_n$ (A_i of sort o)

Deduction/typing rules

$$\frac{}{\mathcal{E};\Gamma \vdash x : A}\;{\scriptstyle (x:A)\in\Gamma} \qquad \frac{\mathcal{E};\Gamma \vdash t : A}{\mathcal{E};\Gamma \vdash t : A'}\;{\scriptstyle A\cong_{\mathcal{E}}A'}$$

$$\frac{\mathcal{E};\Gamma, x : A \vdash t : B}{\mathcal{E};\Gamma \vdash \lambda x\,.\,t : A \Rightarrow B} \qquad \frac{\mathcal{E};\Gamma \vdash t : A \Rightarrow B \qquad \mathcal{E};\Gamma \vdash u : A}{\mathcal{E};\Gamma \vdash tu : B}$$

$$\frac{\mathcal{E}, M = M';\Gamma \vdash t : A}{\mathcal{E};\Gamma \vdash t : M = M' \mapsto A} \qquad \frac{\mathcal{E};\Gamma \vdash t : M = M \mapsto A}{\mathcal{E};\Gamma \vdash t : A}$$

$$\frac{\mathcal{E};\Gamma \vdash t : A}{\mathcal{E};\Gamma \vdash t : \forall x^\tau A}\;{\scriptstyle x^\tau\notin FV(\mathcal{E};\Gamma)} \qquad \frac{\mathcal{E};\Gamma \vdash t : \forall x^\tau A}{\mathcal{E};\Gamma \vdash t : A\{x := N^\tau\}}$$

$$\frac{}{\mathcal{E};\Gamma \vdash \text{œ} : ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

Remark: All proof-terms have type $\top \equiv \mathrm{tt} = \mathrm{ff} \mapsto \bot$ (normalization fails)

From operational semantics...

- Krivine's λ_c-calculus
  - λ-calculus with call/cc and continuation constants:

$$t, u \quad ::= \quad x \quad | \quad \lambda x \,.\, t \quad | \quad tu \quad | \quad \text{æ} \quad | \quad \mathsf{k}_\pi$$

  - An abstract machine with explicit stacks:
    - Stack = list of closed terms (notation: π, π')
    - Process = closed term $\star$ stack

- Evaluation rules (weak head normalization, call by name)

(Grab)	$\lambda x \,.\, t$	$\star$	$u \cdot \pi$	$\succ$	$t\{x := u\}$	$\star$	π
(Push)	tu	$\star$	π	$\succ$	t	$\star$	$u \cdot \pi$
(Save)	æ	$\star$	$t \cdot \pi$	$\succ$	t	$\star$	$\mathsf{k}_\pi \cdot \pi$
(Restore)	k_π	$\star$	$t \cdot \pi'$	$\succ$	t	$\star$	π

… to classical realizability semantics

- Interpreting higher-order terms:
  - Individuals interpreted as natural numbers $[\![\iota]\!] = \mathbb{N}$
  - Propositions interpreted as falsity values $[\![o]\!] = \mathfrak{P}(\Pi)$
  - Functions interpreted set-theoretically $[\![\tau \to \sigma]\!] = [\![\sigma]\!]^{[\![\tau]\!]}$
- Parameterized by a pole $\perp\!\!\!\perp \subseteq \Lambda_c \star \Pi$ (closed under anti-evaluation)
- Interpreting logical constructions:

$$[\![\forall x^\tau A]\!] = \bigcup_{e \in [\![\tau]\!]} [\![A\{x := \dot{e}\}]\!] \qquad [\![A \Rightarrow B]\!] = [\![A]\!]^{\perp\!\!\!\perp} \cdot [\![B]\!]$$

$$[\![M = M' \mapsto A]\!] = \begin{cases} [\![A]\!] & \text{if } [\![M]\!] = [\![M']\!] \\ \varnothing & \text{otherwise} \end{cases}$$

Adequacy

If
- $\mathcal{E}; x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ (in $PA\omega^+$)
- $\rho \models \mathcal{E}, \quad u_1 \Vdash A_1[\rho], \ \ldots, \ u_n \Vdash A_n[\rho]$

then: $t\{x_1 := u_1; \ldots; x_n := u_n\} \Vdash B[\rho]$

Plan

Representing conditions

- **Intuition:** Represent the set of conditions as an upwards closed subset of a meet-semilattice

- Take:
  - A kind κ of conditions, equipped with
  - A binary product $(p, q) \mapsto pq$ (of kind $\kappa \to \kappa \to \kappa$)
  - A unit 1 (of kind κ)
  - A predicate $p \mapsto C[p]$ of well-formedness (of kind $\kappa \to o$)

- **Typical example:** finite functions from τ to σ are modelled by
  - $\kappa \ \equiv\ \tau \to \sigma \to o$ (binary relations $\subseteq \tau \times \sigma$)
  - $pq \ \equiv\ \lambda x^{\tau} y^{\sigma} \,.\, p\,x\,y \vee q\,x\,y$ (union of relations p and q)
  - $1 \ \equiv\ \lambda x^{\tau} y^{\sigma} \,.\, \bot$ (empty relation)
  - $C[p] \ \equiv$ "p is a finite function from τ to σ"

Combinators

- The forcing translation is parameterized by
  - The kind κ + closed terms $\cdot$, 1, C (logical level)
  - 9 closed proof terms $\alpha_*, \alpha_1, \ldots, \alpha_8$ (computational level)

$$
\begin{array}{lcl}
\alpha_* & : & C[1] \\
\alpha_1 & : & \forall p^\kappa\ \forall q^\kappa\ (C[pq] \Rightarrow C[p]) \\
\alpha_2 & : & \forall p^\kappa\ \forall q^\kappa\ (C[pq] \Rightarrow C[q]) \\
\alpha_3 & : & \forall p^\kappa\ \forall q^\kappa\ (C[pq] \Rightarrow C[qp]) \\
\alpha_4 & : & \forall p^\kappa\ (C[p] \Rightarrow C[pp]) \\
\alpha_5 & : & \forall p^\kappa\ \forall q^\kappa\ \forall r^\kappa\ (C[(pq)r] \Rightarrow C[p(qr)]) \\
\alpha_6 & : & \forall p^\kappa\ \forall q^\kappa\ \forall r^\kappa\ (C[p(qr)] \Rightarrow C[(pq)r]) \\
\alpha_7 & : & \forall p^\kappa\ (C[p] \Rightarrow C[p1]) \\
\alpha_8 & : & \forall p^\kappa\ (C[p] \Rightarrow C[1p])
\end{array}
$$

This set is not minimal. One can take $\alpha_*, \alpha_1, \alpha_3, \alpha_4, \alpha_5, \alpha_7$ and define:
$\alpha_2 := \alpha_1 \circ \alpha_3$, $\alpha_6 := \alpha_3 \circ \alpha_5 \circ \alpha_3 \circ \alpha_5 \circ \alpha_3$, $\alpha_8 := \alpha_3 \circ \alpha_7$

Derived combinators

- The combinators $\alpha_1, \ldots, \alpha_8$ can be composed:

  Example: $\alpha_1 \circ \alpha_6 \circ \alpha_3 \quad : \quad \forall p^\kappa\, \forall q^\kappa\, \forall r^\kappa\, (C[(pq)r] \Rightarrow C[rp])$

- We will also use the following derived combinators:

$$
\begin{array}{lcll}
\alpha_9 & := & \alpha_3 \circ \alpha_1 \circ \alpha_6 \circ \alpha_3 & : \forall p^\kappa\, \forall q^\kappa\, \forall r^\kappa\, (C[(pq)r] \Rightarrow C[pr]) \\
\alpha_{10} & := & \alpha_2 \circ \alpha_5 & : \forall p^\kappa\, \forall q^\kappa\, \forall r^\kappa\, (C[(pq)r] \Rightarrow C[qr]) \\
\alpha_{11} & := & \alpha_9 \circ \alpha_4 & : \forall p^\kappa\, \forall q^\kappa\, (C[pq] \Rightarrow C[p(pq)]) \\
\alpha_{12} & := & \alpha_5 \circ \alpha_3 & : \forall p^\kappa\, \forall q^\kappa\, \forall r^\kappa\, (C[p(qr)] \Rightarrow C[q(rp)]) \\
\alpha_{13} & := & \alpha_3 \circ \alpha_{12} & : \forall p^\kappa\, \forall q^\kappa\, \forall r^\kappa\, (C[p(qr)] \Rightarrow C[(rp)q]) \\
\alpha_{14} & := & \alpha_5 \circ \alpha_3 \circ \alpha_{10} \circ \alpha_4 \circ \alpha_2 & : \forall p^\kappa\, \forall q^\kappa\, \forall r^\kappa\, (C[p(qr)] \Rightarrow C[q(rr)]) \\
\alpha_{15} & := & \alpha_9 \circ \alpha_3 & : \forall p^\kappa\, \forall q^\kappa\, \forall r^\kappa\, (C[p(qr)] \Rightarrow C[qp])
\end{array}
$$

- **Important remark:**
  - $C[pq] \Rightarrow C[p] \wedge C[q]$, but $C[p] \wedge C[q] \not\Rightarrow C[pq]$ (in general)
  - Two conditions p and q are compatible when $C[pq]$

Ordering

- Let $\qquad p \le q \quad := \quad \forall r^\kappa (C[pr] \Rightarrow C[qr])$

- $\le$ is a preorder with greatest element 1:

$$\begin{array}{lcl} \lambda c\,.\,c & : & \forall p^\kappa\ (p \le p) \\ \lambda xyc\,.\,y(xc) & : & \forall p^\kappa\ \forall q^\kappa\ \forall r^\kappa\ (p \le q \Rightarrow q \le r \Rightarrow p \le r) \\ \alpha_8 \circ \alpha_2 & : & \forall p^\kappa\ (p \le 1) \end{array}$$

- Product pq is the g.l.b. of p and q:

$$\begin{array}{lcl} \alpha_9 & : & \forall p^\kappa\ \forall q^\kappa\ (pq \le p) \\ \alpha_{10} & : & \forall p^\kappa\ \forall q^\kappa\ (pq \le q) \\ \lambda xy\,.\,\alpha_{13} \circ y \circ \alpha_{12} \circ x \circ \alpha_{11} & : & \forall p^\kappa\ \forall q^\kappa\ \forall r^\kappa\ (r \le p \Rightarrow r \le q \Rightarrow r \le pq) \end{array}$$

- C (set of 'good' conditions) is upwards closed:

$$\lambda xc\,.\,\alpha_1\,(x\,(\alpha_7\,c)) \quad : \quad \forall p^\kappa\ \forall q^\kappa\ (p \le q \Rightarrow C[p] \Rightarrow C[q])$$

- Bad conditions are smallest elements:

$$\lambda xc\,.\,x\,(\alpha_1\,c) \quad : \quad \forall p^\kappa\ (\neg C[p] \Rightarrow \forall q^\kappa\ p \le q)$$

The auxiliary translation $(\_)^*$

- Translating kinds: $\tau \mapsto \tau^*$

$$\iota^* \equiv \iota \qquad o^* \equiv \kappa \to o \qquad (\tau \to \sigma)^* \equiv \tau^* \to \sigma^*$$

Intuition: Propositions become sets of conditions

- Translating terms: $M \mapsto M^*$

$$\begin{array}{rcl} (x^\tau)^* & \equiv & x^{\tau^*} \\ (\lambda x^\tau . M)^* & \equiv & \lambda x^{\tau^*} . M^* \\ (MN)^* & \equiv & M^* N^* \end{array} \qquad \begin{array}{rcl} 0^* & \equiv & 0 \\ \mathsf{s}^* & \equiv & \mathsf{s} \\ \mathsf{rec}_\tau^* & \equiv & \mathsf{rec}_{\tau^*} \end{array}$$

$$\begin{array}{rcl} (\forall x^\tau A)^* & \equiv & \lambda r^\kappa . \forall x^{\tau^*} A^* r \\ (M_1 = M_2 \mapsto A)^* & \equiv & \lambda r^\kappa . M_1^* = M_2^* \mapsto A^* r \\ (A \Rightarrow B)^* & \equiv & \lambda r^\kappa . \forall q^\kappa \forall r'^\kappa [r = qr' \mapsto \forall s^\kappa (C[qs] \Rightarrow A^* s) \Rightarrow B^* r'] \end{array}$$

Lemma

- $(M\{x^\tau := N\})^* \equiv M^*\{x^{\tau^*} := N^*\}$ (substitutivity)
- If $M_1 \cong_{\mathcal{E}} M_2$, then $M_1^* \cong_{\mathcal{E}^*} M_2^*$ (compatibility with conversion)

The forcing translation

- Given a proposition A and a condition p, let:

$$p \Vdash A \quad := \quad \forall r^{\kappa}(C[pr] \Rightarrow A^{*} r)$$

- The forcing translation is trivial on $\forall$ and $\_ = \_ \mapsto \_$:

$$\begin{array}{rcl} p \Vdash \forall x^{\tau} A & \cong_{\varnothing} & \forall x^{\tau^{*}} (p \Vdash A) \\ p \Vdash M_1 = M_2 \mapsto A & \cong_{\varnothing} & M_1^{*} = M_2^{*} \mapsto (p \Vdash A) \end{array}$$

- All the complexity lies in implication! (cf next slide)

General properties

$$\begin{array}{lclcl} \beta_1 & := & \lambda xyc\,.\,y\,(x\,c) & : & \forall p^{\kappa}\ \forall q^{\kappa}\ (q \le p \Rightarrow (p \Vdash A) \Rightarrow (q \Vdash A)) \\ \beta_2 & := & \lambda xc\,.\,x\,(\alpha_1\,c) & : & \forall p^{\kappa}\ (\neg C[p] \Rightarrow p \Vdash A) \\ \beta_3 & := & \lambda xc\,.\,x\,(\alpha_9\,c) & : & \forall p^{\kappa}\ \forall q^{\kappa}\ ((p \Vdash A) \Rightarrow (pq \Vdash A)) \\ \beta_4 & := & \lambda xc\,.\,x\,(\alpha_{10}\,c) & : & \forall p^{\kappa}\ \forall q^{\kappa}\ ((q \Vdash A) \Rightarrow (pq \Vdash A)) \end{array}$$

Forcing an implication

- Definition of $p \Vdash A \Rightarrow B$ looks strange:

$$\begin{aligned} p \Vdash A \Rightarrow B \quad &\equiv \quad \forall r^{\kappa}(C[pr] \Rightarrow (A \Rightarrow B)^{*} r) \\ &\cong_{\varnothing} \quad \forall r^{\kappa}(C[pr] \Rightarrow \forall q^{\kappa} \forall r'^{\kappa}(r = qr' \mapsto (q \Vdash A) \Rightarrow B^{*} r')) \end{aligned}$$

- But it is equivalent to

$$\forall q\,((q \Vdash A) \Rightarrow (pq \Vdash B)) \qquad \left(\textbf{Hint:} \quad \frac{p \Vdash A \Rightarrow B \qquad q \Vdash A}{pq \Vdash B}\right)$$

Coercions between $p \Vdash A \Rightarrow B$ and $\forall q\,((q \Vdash A) \Rightarrow (pq \Vdash B))$

$$\begin{array}{lllcl} \gamma_1 & := & \lambda xcy\,.\,x\,y\,(\alpha_6\,c) & : & (\forall q\,((q \Vdash A) \Rightarrow (pq \Vdash B)) \;\Rightarrow\; p \Vdash A \Rightarrow B) \\ \gamma_2 & := & \lambda xyc\,.\,x\,(\alpha_5\,c)\,y & : & (p \Vdash A \Rightarrow B) \;\Rightarrow\; \forall q\,((q \Vdash A) \Rightarrow (pq \Vdash B)) \\ \gamma_3 & := & \lambda xyc\,.\,x\,(\alpha_{11}\,c)\,y & : & (p \Vdash A \Rightarrow B) \;\Rightarrow\; (p \Vdash A) \;\Rightarrow\; (p \Vdash B) \\ \gamma_4 & := & \lambda xcy\,.\,x\,(y\,(\alpha_{15}\,c)) & : & \neg A^{*}\,p \;\Rightarrow\; p \Vdash A \Rightarrow B \end{array}$$

Translating proof-terms

- Krivine's program transformation $t \mapsto t^*$:

$$
\begin{aligned}
x^* &\equiv x & \text{cc}^* &\equiv \lambda cx\,.\,\text{cc}\,(\lambda k\,.\,x\,(\alpha_{14}\,c)\,(\gamma_4\,k)) \\
(t\,u)^* &\equiv \gamma_3\,t^*\,u^* \\
(\lambda x\,.\,t)^* &\equiv \gamma_1\,(\lambda x\,.\,t^*\,\underbrace{\{x := \beta_4 x\}}_{\text{bounded var}}\,\underbrace{\{x_i := \beta_3 x_i\}_{i=1}^{n}}_{\text{other free vars of } t})
\end{aligned}
$$

$$
\begin{aligned}
\gamma_4 &\equiv \lambda xcy\,.\,x\,(y\,(\alpha_{15}\,c)) \\
\gamma_3 &\equiv \lambda xyc\,.\,x\,(\alpha_{11}\,c)\,y \\
\gamma_1 &\equiv \lambda xcy\,.\,x\,y\,(\alpha_6\,c) \\
\beta_3 &\equiv \lambda xc\,.\,x\,(\alpha_9\,c) \\
\beta_4 &\equiv \lambda xc\,.\,x\,(\alpha_{10}\,c)
\end{aligned}
$$

  - The translation inserts: γ_1 ("fold") in front of each λ
    γ_3 ("apply") in front of each app.
  - A bound occurrence of x in t is translated as $\beta_3^n(\beta_4 x)$, where n is the de Bruijn index of this occurrence

Soundness (in PAω^+)

If $\mathcal{E};\ x_1 : A_1,\ \ldots,\ x_n : A_n \ \vdash\ t \ :\ B$

then $\mathcal{E}^*;\ x_1 : (p \Vdash A_1),\ \ldots,\ x_n : (p \Vdash A_n) \ \vdash\ t^* \ :\ (p \Vdash B)$

Translating proof-terms (optimized)

- The latter program transformation creates bureaucratic β-redexes due to the macros β_3, β_4, γ_3, γ_1 and γ_4
- If we reduce them, we get the following transformation:

$$x^* \equiv x \qquad \text{cc}^* \equiv \lambda cx\,.\,\text{cc}\,(\lambda k\,.\,x\,(\alpha_{14}\;c)\,(\lambda cx\,.\,k\,(x\,(\alpha_{15}\;c))))$$

$$(t\;u)^* \equiv \lambda c\,.\,t^*\,(\alpha_6\;c)\,u^*$$

$$(\lambda x\,.\,t)^* \equiv \lambda cx\,.\,t^*\,\underbrace{\{x := \lambda c\,.\,x\,(\alpha_{10}\;c)\}}_{\text{bounded var}}\,\underbrace{\{x_i := \lambda c\,.\,x_i\,(\alpha_9\;c)\}_{i=1}^{n}}_{\text{other free vars of } t}\,(\alpha_{11}\;c)$$

Soundness (in PAω^+)

If $\mathcal{E};\; x_1 : A_1,\; \ldots,\; x_n : A_n \;\vdash\; t \;:\; B$

then $\mathcal{E}^*;\; x_1 : (p \Vdash A_1),\; \ldots,\; x_n : (p \Vdash A_n) \;\vdash\; t^* \;:\; (p \Vdash B)$

Computational meaning of the transformation

- A proof of $p \Vdash A \equiv \forall r^{\kappa}(C[pr] \Rightarrow A^* r)$ is a function waiting an argument $c : C[pr]$ (for some r) $\rightsquigarrow$ computational condition

$$
\begin{array}{rclcrcl}
(\lambda x\,.\,t)^* & \star & c \cdot u \cdot \pi & \succ & t^*\{x := \beta_4 u\} & \star & \alpha_6\ c \cdot \pi \\
(tu)^* & \star & c \cdot \pi & \succ & t^* & \star & \alpha_{11}\ c \cdot u^* \cdot \pi \\
\text{cc}^* & \star & c \cdot t \cdot \pi & \succ & t & \star & \alpha_{14}\ c \cdot \mathrm{k}_\pi^* \cdot \pi \\
\mathrm{k}_\pi^* & \star & c \cdot t \cdot \pi' & \succ & t & \star & \alpha_{15}\ c \cdot \pi
\end{array}
$$

where: $\mathrm{k}_\pi^* \equiv \gamma_4\, \mathrm{k}_\pi \quad (\approx \lambda cx\,.\,\mathrm{k}_\pi\,(x\,(\alpha_{15}\ c)))$

Evaluation combinators

$$
\begin{array}{rcccc}
\alpha_6 & : & C[p(qr)] & \Rightarrow & C[(pq)r] \\
\alpha_{11} & : & C[pr] & \Rightarrow & C[p(pr)] \\
\alpha_{14} & : & C[p(qr)] & \Rightarrow & C[q(rr)] \\
\alpha_{15} & : & C[p(qr)] & \Rightarrow & C[qp]
\end{array}
$$

Plan

Krivine Forcing Abstract Machine (KFAM) [M.'11]

Terms	t, u	$::=$	$x \mid \lambda x\,.\,t \mid tu \mid \text{œ}$
Environments	e	$::=$	$\emptyset \mid e, x := c$
Closures	c	$::=$	$t[e] \mid \mathsf{k}_\pi \mid \underbrace{t[e]^* \mid \mathsf{k}^*_\pi}_{\text{forcing closures}}$
Stacks	π	$::=$	$\diamond \mid c \cdot \pi$

- Evaluation rules: real mode

$$
\begin{array}{rclcrcll}
x[e, y := c] & \star & \pi & \succ & x[e] & \star & \pi & (y \not\equiv x) \\
x[e, x := c] & \star & \pi & \succ & c & \star & \pi & \\
(\lambda x\,.\,t)[e] & \star & c \cdot \pi & \succ & t[e, x := c] & \star & \pi & \\
(tu)[e] & \star & \pi & \succ & t\,t[e] & \star & u[e] \cdot \pi & \\
\text{œ}[e] & \star & c \cdot \pi & \succ & c & \star & \mathsf{k}_\pi \cdot \pi & \\
\mathsf{k}_\pi & \star & c \cdot \pi' & \succ & c & \star & \pi &
\end{array}
$$

- Evaluation rules: forcing mode

$$
\begin{array}{rclcrcll}
x[e, y := c]^* & \star & c_0 \cdot \pi & \succ & x[e]^* & \star & \alpha_9\ c_0 \cdot \pi & (y \not\equiv x) \\
x[e, x := c]^* & \star & c_0 \cdot \pi & \succ & c & \star & \alpha_{10}\ c_0 \cdot \pi & \\
(\lambda x\,.\,t)[e]^* & \star & c_0 \cdot c \cdot \pi & \succ & t[e, x := c]^* & \star & \alpha_6\ c_0 \cdot \pi & \\
(tu)[e]^* & \star & c_0 \cdot \pi & \succ & t[e]^* & \star & \alpha_{11}\ c_0 \cdot u[e]^* \cdot \pi & \\
\text{œ}[e]^* & \star & c_0 \cdot c \cdot \pi & \succ & c & \star & \alpha_{14}\ c_0 \cdot \mathsf{k}^*_\pi \cdot \pi & \\
\mathsf{k}^*_\pi & \star & c_0 \cdot c \cdot \pi' & \succ & c & \star & \alpha_{15}\ c_0 \cdot \pi &
\end{array}
$$

Adequacy in real and forcing modes

- New abstract machine means:
  - New classical realizability model (based on the KFAM)
  - New adequacy results

Adequacy (real mode)

If
- $\mathcal{E}; x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ (in PAω^+)
- $\rho \models \mathcal{E}$, $c_1 \Vdash A_1[\rho]$, ..., $c_n \Vdash A_n[\rho]$

then: $t[x_1 := c_1, \ldots, x_n := c_n] \Vdash B[\rho]$

- Assuming that $\alpha_i \Vdash$ type of α_i (for $i = 6, 9, 10, 11, 14, 15$)

Adequacy (forcing mode)

If
- $\mathcal{E}; x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ (in PAω^+)
- $\rho \models \mathcal{E}^*$, $c_1 \Vdash (p_1 \text{ IF } A_1[\rho])$, ..., $c_n \Vdash (p_n \text{ IF } A_n[\rho])$

then: $t[x_1 := c_1; \ldots; x_n := c_n]^* \Vdash ((p_0 p_1) \cdots p_n \text{ IF } B[\rho])$

Program extraction in presence of forcing

- Assume that:
  1. We got a proof of B under some axiom A
     $$x : A \ \vdash \ u \ : \ B \qquad \text{(user program)}$$
  2. Axiom A is not provable, but it can be forced using a suitable set of forcing conditions $(C, \leq)$:
     $$\vdash \ s \ : \ (1 \Vdash A) \qquad \text{(system program)}$$
- Then:
  1. We have: $u[x := s[]]^* \ \Vdash \ (1 \Vdash B)$
  2. If moreover B is an arithmetical formula
     $$(\xi_B \ z)[z := u[x := s[]]^*] \ \Vdash \ B$$
     using a suitable wrapper $\xi_B \ \Vdash \ (1 \Vdash B) \Rightarrow B$

Plan

Realizability algebras

[Krivine'10]

Definition

A realizability algebra $\mathscr{A}$ is given by:

- 3 sets $\mathbf{\Lambda}$ ($\mathscr{A}$-terms), $\mathbf{\Pi}$ ($\mathscr{A}$-stacks), $\mathbf{\Lambda} \star \mathbf{\Pi}$ ($\mathscr{A}$-processes)
- 3 functions $(\cdot) : \mathbf{\Lambda} \times \mathbf{\Pi} \to \mathbf{\Pi}$, $(\star) : \mathbf{\Lambda} \times \mathbf{\Pi} \to \mathbf{\Lambda} \star \mathbf{\Pi}$, $(\mathrm{k}\_) : \mathbf{\Pi} \to \mathbf{\Lambda}$
- A compilation function $(t, \sigma) \mapsto t[\sigma]$ that takes
  - an open proof term t
  - a $\mathbf{\Lambda}$-substitution σ closing t

  and returns an $\mathscr{A}$-term $t[\sigma] \in \mathbf{\Lambda}$
- A set of $\mathscr{A}$-processes $\perp\!\!\!\perp \subseteq \mathbf{\Lambda} \star \mathbf{\Pi}$ such that:

$$
\begin{array}{rcl}
\sigma(x) \star \pi \in \perp\!\!\!\perp & \text{implies} & x[\sigma] \star \pi \in \perp\!\!\!\perp \\
t[\sigma, x := a] \star \pi \in \perp\!\!\!\perp & \text{implies} & (\lambda x \,.\, t)[\sigma] \star a \cdot \pi \in \perp\!\!\!\perp \\
t[\sigma] \star u[\sigma] \cdot \pi \in \perp\!\!\!\perp & \text{implies} & (tu)[\sigma] \star \pi \in \perp\!\!\!\perp \\
a \star \mathrm{k}_\pi \cdot \pi \in \perp\!\!\!\perp & \text{implies} & \text{cc}[\sigma] \star a \cdot \pi \in \perp\!\!\!\perp \\
a \star \pi \in \perp\!\!\!\perp & \text{implies} & \mathrm{k}_\pi \star a \cdot \pi' \in \perp\!\!\!\perp
\end{array}
$$

Realizability model of PAω^+ (general case)

- Parameterized by a realizability algebra $\mathscr{A} = (\mathbf{\Lambda}, \mathbf{\Pi}, \mathbf{\Lambda} \star \mathbf{\Pi}, \cdots, \perp\!\!\!\perp)$
- Interpreting higher-order terms:
  - Individuals interpreted as natural numbers $[\![\iota]\!] = \mathbb{N}$
  - Propositions interpreted as $\mathscr{A}$-falsity values $[\![o]\!] = \mathfrak{P}(\mathbf{\Pi})$
  - Functions interpreted set-theoretically $[\![\tau \to \sigma]\!] = [\![\sigma]\!]^{[\![\tau]\!]}$
- Interpreting logical constructions:

$$[\![\forall x^\tau A]\!] = \bigcup_{e \in [\![\tau]\!]} [\![A\{x := \dot{e}\}]\!] \qquad [\![A \Rightarrow B]\!] = [\![A]\!]^{\perp\!\!\!\perp} \cdot [\![B]\!]$$

$$[\![M = M' \mapsto A]\!] = \begin{cases} [\![A]\!] & \text{if } [\![M]\!] = [\![M']\!] \\ \varnothing & \text{otherwise} \end{cases}$$

Adequacy

If
- $\mathcal{E}; x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ (in PAω^+)
- $\rho \models \mathcal{E}, \quad u_1 \Vdash A_1[\rho], \ \ldots, \ u_n \Vdash A_n[\rho]$

then: $t[x_1 := u_1; \ldots; x_n := u_n] \Vdash B[\rho]$

Examples (1/2)

- From an implementation of λ_c:

Standard realizability algebra

- $\boldsymbol{\Lambda} = \Lambda$, $\quad \boldsymbol{\Pi} = \Pi$, $\quad \boldsymbol{\Lambda} \star \boldsymbol{\Pi} = \Lambda \star \Pi$
- k_π, $\quad t \cdot \pi$, $\quad t \star \pi$ defined as themselves
- Compilation function $(t, \sigma) \mapsto t[\sigma]$ defined by substitution
- $\perp\!\!\!\perp \; = \;$ any saturated set of processes

- We can do the same for all classical λ-calculi:
  - Parigot's $\lambda\mu$-calculus
  - Curien-Herbelin's $\bar{\lambda}\mu$-calculus (CBN or CBV)
  - Barbanera-Berardi's symmetric λ-calculus ($\pitchfork$ comes for free)

Examples (2/2)

- From a forcing poset P defined as an upwards closed subset of a meet semi-lattice $\mathcal{L}$: $P \subseteq \mathcal{L}$, $P\!\uparrow$

- $\mathbf{\Lambda} = \mathbf{\Pi} = \mathbf{\Lambda} \star \mathbf{\Pi} = \mathcal{L}$
- $\mathrm{k}_\pi = \pi$, $\quad t \cdot \pi = t \star \pi = t\pi \quad$ (product in $\mathcal{L}$)
- Compilation function $(t, \sigma) \mapsto t[\sigma]$:

$$t[\sigma] \;=\; \prod_{x \in FV(t)} \sigma(x)$$

- $\perp\!\!\!\perp \;=\; \mathcal{L} \setminus P$

- Corresponding realizability model isomorphic to the forcing model defined from the poset P

KFAM: The realizability algebra of real mode

- From a saturated set $\perp\!\!\!\perp$ in the KFAM:

The realizability algebra $\mathscr{A} = (\mathbf{\Lambda}, \mathbf{\Pi}, \mathbf{\Lambda} \star \mathbf{\Pi}, \ldots, \perp\!\!\!\perp)$

- $\mathbf{\Lambda}$, $\mathbf{\Pi}$, $\mathbf{\Lambda} \star \mathbf{\Pi}$ $=$ sets of closures, stacks, processes of the KFAM
- k_π (real mode), $t \cdot \pi$, $t \star \pi$ defined as in the KFAM
- Compilation function $(t, \sigma) \mapsto t[\sigma]$ $=$ closure formation (real mode)
- $\perp\!\!\!\perp$ $=$ itself

- Adequacy w.r.t. the algebra $\mathscr{A}$ $=$

  Adequacy in the KFAM in real mode (w.r.t. the pole $\perp\!\!\!\perp$)

KFAM: The realizability algebra of forcing mode

- Given $\mathscr{A} = (\mathbf{\Lambda}, \mathbf{\Pi}, \mathbf{\Lambda} \star \mathbf{\Pi}, \ldots, \perp\!\!\!\perp)$ (cf prev. slide)
  $+$ a forcing structure $(\kappa, C, \cdot, 1)$

The realizability algebra $\mathscr{A}^* = (\mathbf{\Lambda}^*, \mathbf{\Pi}^*, \mathbf{\Lambda}^* \star \mathbf{\Pi}^*, \ldots, \perp\!\!\!\perp^*)$

- $\mathbf{\Lambda}^* = \mathbf{\Lambda} \times [\![\kappa]\!], \quad \mathbf{\Pi}^* = \mathbf{\Pi} \times [\![\kappa]\!], \quad \mathbf{\Lambda}^* \star \mathbf{\Pi}^* = (\mathbf{\Lambda} \star \mathbf{\Pi}) \times [\![\kappa]\!]$
- $\mathsf{k}_{(\pi,p)} = (\mathsf{k}^*_\pi, p)$ (forcing mode)
- $(t, p) \cdot (\pi, q) = (t \cdot \pi, pq)$
- $(t, p) \star (\pi, q) = (t \star \pi, pq)$
- Compilation function $(t, \sigma) \mapsto t[\sigma]$:

$$t[x_1 := (c_1, p_1); \ldots; x_n := (c_n, p_n)] = \big(t[x_1 := c_1; \ldots; x_n := c_n]^*,\ ((1p_1)\cdots)p_n\big)$$
(forcing mode)

- $\perp\!\!\!\perp^* = \{(t \star \pi, p) \ : \ \forall c \in \mathbf{\Lambda}\ ((c \Vdash_{\mathscr{A}} C[p]) \Rightarrow (t \star c \cdot \pi) \in \perp\!\!\!\perp)\}$

The connection lemma

- Write $[\![\_]\!]$ (resp. $[\![\_]\!]^*$) the interpretation w.r.t. $\mathscr{A}$ (resp. w.r.t. $\mathscr{A}^*$)
- Notice that: $[\![o]\!]^* \;=\; \mathfrak{P}(\mathbf{\Pi} \times [\![\kappa]\!]) \;\simeq\; (\mathfrak{P}(\Pi))^{[\![\kappa]\!]} \;=\; [\![o^*]\!]$

Connection lemma

1. There exists an iso: $\psi_\tau : [\![\tau^*]\!] \xrightarrow{\sim} [\![\tau]\!]^*$
2. For all closed M of kind τ: $[\![M]\!]^* = \psi_\tau([\![M^*]\!])$
3. Given a closed formula A and a pair $(c, p) \in \mathbf{\Lambda}^*$ $(= \mathbf{\Lambda} \times [\![\kappa]\!])$:

$$(c, p) \Vdash_{\mathscr{A}^*} A \quad \Leftrightarrow \quad c \Vdash_{\mathscr{A}} (p \Vdash A)$$

- Connection lemma + Adequacy w.r.t. the algebra $\mathscr{A}^*$ =

  Adequacy in the KFAM in forcing mode (w.r.t. the pole $\perp\!\!\!\perp$)

To sum up

- **From syntax...**
  - The program transform $t \mapsto t^*$ underlying Cohen's forcing:

$$\vdash t : A \quad \rightsquigarrow \quad \vdash t^* : (p \Vdash A)$$

  - A new machine (KFAM) with two execution modes such that

$$t[]^* \quad \text{has the same behavior as} \quad t^*[]$$

- **... to semantics: iterated forcing**
  - Two realizability algebras $\mathscr{A}$ and $\mathscr{A}'$ related by

$$(c, p) \Vdash_{\mathscr{A}^*} A \quad \Leftrightarrow \quad c \Vdash_{\mathscr{A}} (p \Vdash A)$$

  - Two adequacy lemmas (real/forcing) as instances of the general lemma of adequacy

Conclusion (1/2)

Underlying methodology

Translation of formulas & proofs	$\rightsquigarrow$	Program transform	$\rightsquigarrow$	Computation model (transform becomes identity)

- This methodology applies to the forcing translation
  - Computational meaning of the underlying program transformation
  - A new abstract machine: the KFAM
  - Reminiscent from well known tricks of computer architecture (protection rings, virtual memory, hardware tracing, ...)
- New insights in logic:
  - Logical meaning of explicit environments
  - Logical meaning of a particular side effect
  - Backtrack defines the limit between the stack and the memory

Conclusion (2/2)

- Future work:
  1. How this computation model is used in practice?
     - Hint: try simple axioms first!
  2. Extend extraction techniques to the forcing mode
  3. Use this methodology the other way around!
     - Deduce new logical translations from computation models borrowed to computer architecture, operating systems, ...
- Several connections between forcing and side effects
  - Forcing in classical realizability [Krivine'08, '09, '10]
  - Realizability with states and dependent choice [Miquel'09]
- Towards an integration of side effects into the Curry-Howard correspondence?