An introduction to classical realizability

Alexandre Miquel
Plume team – LIP/ENS Lyon

February 14th, 2012
LI2012 – Luminy

What is classical realizability ?

- Developed in the middle of the 90's by Jean-Louis Krivine
  - Based on Griffin's discovery about the connection between classical reasoning and control operators (call/cc)
  - Complete reformulation of the principles of Kleene realizability to take into account classical reasoning
  - Interprets the axiom of dependent choices (DC)
- Initially designed for PA2 (+ DC), but extends to :
  - Higher-order arithmetic (PAω)
  - Zermelo-Fraenkel set theory (ZF)
  - The calculus of constructions with universes (with classical logic in Prop)
- Compatible with Cohen forcing (realizability algebras)

Plan

Plan

Terms, stacks and processes

- Syntax of the language parameterized by
  - A countable set $\mathcal{K} = \{\mathfrak{c}; \ldots\}$ of instructions, containing at least the instruction $\mathfrak{c}$ (call/cc)
  - A countable set Π_0 of stack constants (or stack bottoms)

Terms, stacks and processes

Terms	t, u	$::=$	x	$\mid$	$\lambda x\,.\,t$	$\mid$	tu	$\mid$	κ	$\mid$	k_π	$(\kappa \in \mathcal{K})$
Stacks	π, π'	$::=$	α	$\mid$	$t \cdot \pi$							($\alpha \in \Pi_0$, t closed)
Processes	p, q	$::=$	$t \star \pi$									(t closed)

- A λ-calculus with two kinds of constants :
  - Instructions $\kappa \in \mathcal{K}$, including $\mathfrak{c}$
  - Continuation constants k_π, one for every stack π (generated by $\mathfrak{c}$)
- **Notation :** Λ, Π, $\Lambda \star \Pi$ (sets of closed terms / stacks / processes)

Proof-like terms

- **Proof-like term** $\equiv$ Term containing no continuation constant

Proof-like terms $t, u \quad ::= \quad x \mid \lambda x\,.\,t \mid tu \mid \kappa \qquad (\kappa \in \mathcal{K})$

  - **Idea :** All realizers coming from actual proofs are of this form, continuation constants k_π are treated as paraproofs
  - **Notation :** PL $\equiv$ set of closed proof-like terms

- Natural numbers encoded as proof-like terms by :

Krivine numerals $\overline{n} \equiv \overline{s}^n\,\overline{0} \in \mathrm{PL} \qquad (n \in \mathbb{N})$

writing $\overline{0} \equiv \lambda xy\,.\,x$ and $\overline{s} \equiv \lambda nxy\,.\,y\,(n\,x\,y)$

  - **Note :** Krivine numerals $\not\equiv$ Church numerals, but β-equivalent

The Krivine Abstract Machine (KAM) (1/2)

- We assume that the set $\Lambda \star \Pi$ comes with a preorder $p \succ p'$ of evaluation satisfying the following rules :

Krivine Abstract Machine (KAM)

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

(+ reflexivity & transitivity)

- Evaluation not defined but axiomatized. The preorder $p \succ p'$ is another parameter of the calculus, just like the sets $\mathcal{K}$ and Π_0
- Extensible machinery : can add extra instructions and rules
  (We shall see examples later)

The Krivine Abstract Machine (KAM) (2/2)

- Rules **Push** and **Grab** implement weak head β-reduction (call-by-name strategy) :

Push	$tu \star \pi$	$\succ$	$t \star u \cdot \pi$
Grab	$\lambda x\,.\,t \star u \cdot \pi$	$\succ$	$t\{x := u\} \star \pi$

  - Example : $(\lambda xy\,.\,t)\,u\,v \star \pi \succ \lambda xy\,.\,t \star u \cdot v \cdot \pi \succ t\{x := u\}\{y := v\} \star \pi$

- Rules **Save** and **Restore** implement backtracking :

Save	$\mathfrak{cc} \star u \cdot \pi$	$\succ$	$u \star \mathsf{k}_\pi \cdot \pi$
Restore	$\mathsf{k}_\pi \star u \cdot \pi'$	$\succ$	$u \star \pi$

  - Instruction $\mathfrak{cc}$ creates continuation constants k_π.
    Most often used in the pattern
    $$\mathfrak{cc}\,(\lambda k\,.\,t) \star \pi \succ \cdots \succ t\{k := \mathsf{k}_\pi\} \star \pi$$
  - Continuation constant k_π restores the saved context π

Examples of extra instructions

- The instruction quote

$$\mathrm{quote} \star t \cdot u \cdot \pi \quad \succ \quad u \star \overline{\lceil t \rceil} \cdot \pi$$

where $t \mapsto \lceil t \rceil$ is a fixed bijection from Λ to $\mathbb{N}$

  - Useful to realize the axiom of dependent choices [Krivine'03]

- The instruction eq

$$\mathrm{eq} \star t_1 \cdot t_2 \cdot u \cdot v \cdot \pi \quad \succ \quad \begin{cases} u \star \pi & \text{if } t_1 \equiv t_2 \\ v \star \pi & \text{if } t_1 \not\equiv t_2 \end{cases}$$

  - Tests syntactic equality $t_1 \equiv t_2$
  - Can be implemented using quote

- The instruction $\pitchfork$ (fork)

$$\pitchfork \star u \cdot v \cdot \pi \quad \succ \quad \begin{cases} u \star \pi \\ v \star \pi \end{cases}$$

  - Non deterministic choice operator
  - Useful for pedagogy – bad for realizability (collapses to forcing)

Plan

The language of (minimal) second-order logic

- Second-order logic deals with two kinds of objects :
  - 1st-order objects = individuals (i.e. basic objects of the theory)
  - 2nd-order objects = k-ary relations over individuals

First-order terms and formulas

First-order terms $e, e' ::= x \mid f(e_1, \ldots, e_k)$

Formulas $A, B ::= X(e_1, \ldots, e_k) \mid A \Rightarrow B \mid \forall x\, A \mid \forall X\, A$

- Two kinds of variables
  - 1st-order vars : x, y, z, ... (not to be confused with λ-variables !)
  - 2nd-order vars : X, Y, Z, ... of all arities $k \geq 0$
- Two kinds of substitution :
  - 1st-order subst. : $e\{x := e_0\}$, $A\{x := e_0\}$ (defined as usual)
  - 2nd-order subst. : $A\{X := P_0\}$, $P\{X := P_0\}$ (postponed)

First-order terms

- Defined from a first order signature Σ (as usual) :

First-order terms $\quad e, e' \quad ::= \quad x \quad | \quad f(e_1, \ldots, e_k)$

  - f ranges over k-ary function symbols in Σ

- In what follows we assume that :
  - Each k-ary function symbol f is interpreted in $\mathbb{N}$ by a function
$$f^{\mathbb{N}} \ : \ \mathbb{N}^k \to \mathbb{N}$$
  - The signature Σ contains a function symbol for every primitive recursive function : $\quad 0, s, +, \times, \uparrow, \ldots$

- Denotation (in $\mathbb{N}$) of a closed first-order term e written $[\![e]\!]$

Formulas

- Formulas of minimal second-order logic

Formulas

$$A, B ::= X(e_1, \ldots, e_k) \mid A \Rightarrow B \mid \forall x\, A \mid \forall X\, A$$

only based on implication and 1st/2nd-order universal quantification

- Other connectives/quantifiers are defined (second-order encodings)

$$\begin{array}{rcll}
\bot & \equiv & \forall Z\, Z & \text{(absurdity)} \\
\neg A & \equiv & A \Rightarrow \bot & \text{(negation)} \\
A \wedge B & \equiv & \forall Z\, ((A \Rightarrow B \Rightarrow Z) \Rightarrow Z) & \text{(conjunction)} \\
A \vee B & \equiv & \forall Z\, ((A \Rightarrow Z) \Rightarrow (B \Rightarrow Z) \Rightarrow Z) & \text{(disjunction)} \\
\exists x\, A(x) & \equiv & \forall Z\, (\forall x\, (A(x) \Rightarrow Z) \Rightarrow Z) & \text{(1st-order } \exists) \\
\exists X\, A(X) & \equiv & \forall Z\, (\forall X\, (A(X) \Rightarrow Z) \Rightarrow Z) & \text{(2nd-order } \exists) \\
e_1 = e_2 & \equiv & \forall Z\, (Z(e_1) \Rightarrow Z(e_2)) & \text{(Leibniz equality)}
\end{array}$$

Predicates

- 2nd-order variables represent unknown (abstract) relations
- Concrete relations are represented using predicates (syntactic sugar)

Predicates $P, Q \quad ::= \quad \hat{x}_1 \cdots \hat{x}_k A$ (of arity k)

- Let $P \equiv \hat{x}_1 \cdots \hat{x}_k A$
  - Variables $x_1, \ldots, x_n$ (pairwise $\not\equiv$) are the arguments of P
  - Other free variables of formula A are the parameters of P
  - **Notation :** $FV(P) = FV(A) \setminus \{x_1, \ldots, x_k\}$ (free vars = params)
- Predicates are subject to α-conversion ($\hat{x}_i$s treated as binders)
- 0-ary predicates are formulas

Predicate application / substitution

- Partial/total application of $P \equiv \hat{x}_1 \cdots \hat{x}_k A$ to $e_1, \ldots, e_\ell$:

$$P(e_1, \ldots, e_\ell) \equiv \hat{x}_{\ell+1} \cdots \hat{x}_k A\{x_1 := e_1; \ldots; x_\ell := e_\ell\} \qquad (\ell \leq k)$$

where $x_j \notin FV(e_i)$ for $i \in [1..\ell]$, $j \in [\ell+1..k]$

Result is a $(k-\ell)$-ary predicate, and a formula if $k = \ell$

- Every k-ary 2nd-order variable may be viewed as a predicate :

$$X \equiv \hat{x}_1 \cdots \hat{x}_k X(x_1, \ldots, x_k)$$

- Second-order substitution (X, P of same arity)

$$(X(e_1, \ldots, e_k))\{X := P\} \equiv P(e_1, \ldots, e_k)$$

  - In a formula : $A\{X := P\}$
  - In a predicate : $Q\{X := P\}$

Unary predicates as sets

- Unary predicates represent sets of individuals

  Syntactic sugar : $\{x : A\} \equiv \hat{x}A, \qquad e \in P \equiv P(e)$

Example : The set $\mathbb{N}$ of Dedekind numerals

$$\mathbb{N} \equiv \{x : \forall Z\,(0 \in Z \Rightarrow \forall y\,(y \in Z \Rightarrow s(y) \in Z) \Rightarrow x \in Z\}$$

- Relativized quantifications :

$$\begin{array}{lcl}(\forall x \in P)\,A(x) & \equiv & \forall x\,(x \in P \Rightarrow A(x)) \\ (\exists x \in P)\,A(x) & \equiv & \forall Z\,(\forall x\,(x \in P \Rightarrow A(x) \Rightarrow Z) \Rightarrow Z) \\ & \Leftrightarrow & \exists x\,(x \in P \wedge A(x))\end{array}$$

- Inclusion and extensional equality :

$$\begin{array}{lcl}P \subseteq Q & \equiv & \forall x\,(x \in P \Rightarrow x \in Q) \\ P = Q & \equiv & \forall x\,(x \in P \Leftrightarrow x \in Q)\end{array}$$

- Set constructors : $P \cup Q \equiv \{x : x \in P \vee x \in Q\}$ (etc.)

A type system for second-order logic (λNK2)

- Use proof-like terms as Curry-style proof terms
  Represent the computational contents of classical proofs
- **Typing judgement :** $\underbrace{x_1 : A_1, \ldots, x_n : A_n}_{\text{typing context } \Gamma} \vdash t : B$

Typing rules

$$\frac{}{\Gamma \vdash x : A}\ (x{:}A)\in\Gamma$$

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

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall x\, A}\ x\notin FV(\Gamma) \qquad \frac{\Gamma \vdash t : \forall x\, A}{\Gamma \vdash t : A\{x := e\}}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash t : \forall X\, A}\ X\notin FV(\Gamma) \qquad \frac{\Gamma \vdash t : \forall X\, A}{\Gamma \vdash t : A\{X := P\}}$$

$$\frac{}{\Gamma \vdash \mathfrak{c} : ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

Typing examples

- Intuitionistic principes :

$$\begin{array}{rcll}
\textbf{pair} & \equiv & \lambda xyz\,.\,z\,x\,y & : \quad \forall X\,\forall Y\,(X \Rightarrow Y \Rightarrow X \wedge Y) \\
\textbf{fst} & \equiv & \lambda z\,.\,z\,(\lambda xy\,.\,x) & : \quad \forall X\,\forall Y\,(X \wedge Y \Rightarrow X) \\
\textbf{snd} & \equiv & \lambda z\,.\,z\,(\lambda xy\,.\,y) & : \quad \forall X\,\forall Y\,(X \wedge Y \Rightarrow Y) \\
\textbf{refl} & \equiv & \lambda z\,.\,z & : \quad \forall x\,(x = x) \\
\textbf{trans} & \equiv & \lambda xyz\,.\,y\,(x\,z) & : \quad \forall x\,\forall y\,\forall z\,(x = y \Rightarrow y = z \Rightarrow x = z)
\end{array}$$

- Excluded middle, double negation elimination :

$$\begin{array}{rcll}
\textbf{left} & \equiv & \lambda xuv\,.\,u\,x & : \quad \forall X\,\forall Y\,(X \Rightarrow X \vee Y) \\
\textbf{right} & \equiv & \lambda yuv\,.\,v\,y & : \quad \forall X\,\forall Y\,(Y \Rightarrow X \vee Y) \\
\textbf{EM} & \equiv & \mathfrak{cc}\,(\lambda k\,.\,\textbf{right}\,(\lambda x\,.\,k\,(\textbf{left}\,x))) & : \quad \forall X\,(X \vee \neg X) \\
\textbf{DNE} & \equiv & \lambda z\,.\,\mathfrak{cc}\,(\lambda k\,.\,z\,k) & : \quad \forall X\,(\neg\neg X \Rightarrow X)
\end{array}$$

- De Morgan laws :

$$\begin{array}{rcl}
\lambda zy\,.\,z\,(\lambda x\,.\,yx) & : & \exists x\,A(x) \;\Rightarrow\; \neg\forall x\,\neg A(x) \\
\lambda zy\,.\,\mathfrak{cc}\,(\lambda k\,.\,z\,(\lambda x\,.\,k\,(y\,x))) & : & \neg\forall x\,\neg A(x) \;\Rightarrow\; \exists x\,A(x)
\end{array}$$

Adding axioms

- Defining equations of primitive recursive functions :

$$\forall x\,(x + 0 = x) \qquad \forall x\,\forall y\,(x + s(y) = s(x + y))$$
$$\forall x\,(x \times 0 = 0) \qquad \forall x\,\forall y\,(x \times s(y) = x \times y + x) \qquad \text{(etc.)}$$

- Peano 3rd and 4th axioms :

$$\text{(P3)} \quad \forall x\,\forall y\,(s(x) = s(y) \Rightarrow x = y)$$
$$\text{(P4)} \quad \forall x\,\neg(s(x) = 0)$$

- **Problem :** Induction axiom neither derivable nor realizable :

$$\begin{aligned} \textsc{Ind} \quad &\equiv \quad \forall x\,(x \in \mathbb{N}) \\ &\Leftrightarrow \quad \forall Z\,[0 \in Z \Rightarrow \forall y\,(y \in Z \Rightarrow s(y) \in Z) \Rightarrow \forall x\,(x \in Z)] \end{aligned}$$

- **Solution :** Relativize all 1st-order quantifications to the set $\mathbb{N}$:

Theorem

If $\text{PA2} \vdash A$, then $\text{PA2} - \textsc{Ind} \vdash A^{\mathbb{N}}$ ($A^{\mathbb{N}} \equiv A$ relativized to $\mathbb{N}$)

Plan

Classical realizability : principles

- **Intuitions :**
  - term = "proof" / stack = "counter-proof"
  - process = "contradiction" (slogan : never trust a classical realizer !)
- Classical realizability model parameterized by a pole $\perp\!\!\!\perp$
  = set of processes closed under anti-evaluation (or saturated)

$$\text{If} \quad p \succ p' \quad \text{and} \quad p' \in \perp\!\!\!\perp, \quad \text{then} \quad p \in \perp\!\!\!\perp$$

- Each formula A is interpreted as two sets :
  - A set of stacks $\|A\|$ (falsity value)
  - A set of terms $|A|$ (truth value)
- Falsity value $\|A\|$ defined by induction on A (negative interpretation)
- Truth value $|A|$ defined by orthogonality :

$$|A| \quad = \quad \|A\|^{\perp\!\!\!\perp} \quad = \quad \{t \in \Lambda \ : \ \forall \pi \in \|A\| \quad t \star \pi \in \perp\!\!\!\perp\}$$

Architecture of the realizability model

- The realizability model $\mathscr{M}_{\perp\!\!\!\perp}$ is defined from :
  - The full standard model $\mathscr{M}$ of PA2 : the ground model
    (but we could take any model $\mathscr{M}$ of PA2 as well)
  - An instance $(\mathcal{K}, \Pi_0, \succ)$ of the λ_c-calculus
  - A saturated set of processes $\perp\!\!\!\perp \subseteq \Lambda \star \Pi$ (the pole)
- Architecture :
  - First-order terms/variables interpreted as natural numbers $n \in \mathbb{N}$
  - Formulas interpreted as falsity values $S \in \mathfrak{P}(\Pi)$
  - k-ary 2nd-order variables (and k-ary predicates) interpreted as falsity functions $F : \mathbb{N}^k \to \mathfrak{P}(\Pi)$.

Formulas with parameters $\quad A, B \quad ::= \quad \cdots \quad | \quad \dot{F}(e_1, \ldots, e_k)$

Add a predicate constant $\dot{F}$ for every falsity function $\dot{F} : \mathbb{N}^k \to \mathfrak{P}(\Pi)$

Interpreting closed formulas with parameters

Let A be a closed formula (with parameters)

- Falsity value $\|A\|$ defined by induction on A :

$$\begin{aligned}
\|\dot{F}(e_1,\ldots,e_n)\| &= F([\![e_1]\!],\ldots,[\![e_n]\!]) \\
\|A \Rightarrow B\| &= |A| \cdot \|B\| = \{t \cdot \pi \ : \ t \in |A|, \ \ \pi \in \|B\|\} \\
\|\forall x \ A\| &= \bigcup_{n \in \mathbb{N}} \|A\{x := n\}\| \\
\|\forall X \ A\| &= \bigcup_{F:\mathbb{N}^k \to \mathfrak{P}(\Pi)} \|A\{X := \dot{F}\}\|
\end{aligned}$$

- Truth value $|A|$ defined by orthogonality :

$$|A| = \|A\|^{\perp\!\!\!\perp} = \{t \in \Lambda \ : \ \forall \pi \in \|A\| \quad t \star \pi \in \perp\!\!\!\perp\}$$

Anatomy of the model (1/2)

- **Denotation of universal quantification :**

  Falsity value : $\|\forall x\, A\| \;=\; \bigcup_{n\in\mathbb{N}} \|A\{x := n\}\|$ (by definition)

  Truth value : $|\forall x\, A| \;=\; \bigcap_{n\in\mathbb{N}} |A\{x := n\}|$ (by orthogonality)

  (and similarly for 2nd-order universal quantification)

- **Denotation of implication :**

  Falsity value : $\|A \Rightarrow B\| \;=\; |A| \cdot \|B\|$ (by definition)

  Truth value : $|A \Rightarrow B| \;\subseteq\; |A| \to |B|$ (by orthogonality)

  writing $|A| \to |B| \;=\; \{t \in \Lambda \;:\; \forall u \in |A| \;\; tu \in |B|\}$ (realizability arrow)

  1. Converse inclusion does not hold in general, unless $\perp\!\!\!\perp$ closed under **Push**
  2. In all cases : If $t \in |A| \to |B|$, then $\lambda x\,.\,tx \in |A \Rightarrow B|$ (η-expansion)

Anatomy of the model (2/2)

- Falsity value $\|A\|$ and truth value $|A|$ depend on the pole $\bot\!\!\!\bot$
  $\rightsquigarrow$ write them $\|A\|_{\bot\!\!\!\bot}$ and $|A|_{\bot\!\!\!\bot}$ to recall the dependecy
- **Degenerate case :** $\bot\!\!\!\bot = \varnothing$
  - Truth values can take only two values : $\varnothing$ and Λ
  - Classical realizability simply mimics the Tarski interpretation :

Degenerated interpretation

In the case where $\bot\!\!\!\bot = \varnothing$, for every closed formula A :

$$|A| = \begin{cases} \Lambda & \text{if } \mathscr{M} \models A \\ \varnothing & \text{if } \mathscr{M} \not\models A \end{cases}$$

- **Non degenerate cases :** $\bot\!\!\!\bot \neq \varnothing$
  - Every truth value $|A|$ is inhabited :
    If $t_0 \star \pi_0 \in \bot\!\!\!\bot$, then $\mathsf{k}_{\pi_0} t_0 \in |A|$ for all A (paraproof)
  - We shall only consider realizers that are proof-like terms

The realizability relation

- Realizability relations :

$$t \Vdash A \quad \equiv \quad t \in |A|_{\perp\!\!\!\perp} \qquad \text{(Realizability w.r.t. } \perp\!\!\!\perp\text{)}$$
$$t \Vdash\!\!\!\Vdash A \quad \equiv \quad \forall \perp\!\!\!\perp \;\; t \in |A|_{\perp\!\!\!\perp} \qquad \text{(Universal realizability)}$$

- The computational behavior of a closed λ_c-term determines the formulæ it realizes :

Example :
$$\text{cc} \star t \cdot \pi \succ t \star \text{k}_\pi \cdot \pi$$
$$\text{k}_\pi \star t \cdot \pi' \succ t \star \pi$$

Realizing Peirce's law

1. If $\pi \in \|A\|$, then $\text{k}_\pi \Vdash A \Rightarrow B$ (relatively to a pole $\perp\!\!\!\perp$)
2. $\text{cc} \Vdash\!\!\!\Vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$

Proof : Exercise !

Adequacy

- **Valuation** = function ρ such that :
  - $\rho(x) \in \mathbb{N}$ for every 1st-order variable x
  - $\rho(X) : \mathbb{N}^k \to \mathfrak{P}(\Pi)$ for every k-ary 2nd-order variable X

- Closing a formula A with a valuation ρ : $A[\rho]$ (with parameters)

- A judgment $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ is adequate (w.r.t. $\perp\!\!\!\perp$)
  if for all valuations ρ, for all $u_1 \Vdash A_1[\rho]$, ..., $u_1 \Vdash A_n[\rho]$:

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

Theorem (Adequacy)

All derivable judgments of λNK2 are adequate (w.r.t. any pole $\perp\!\!\!\perp$)

Corollary : If $\vdash t : A$ (A closed), then $t \Vdash\!\vdash A$

Subtyping

- Judgment $A \leq B$ ('A is a subtype of B') adequate when
$$\|B[\rho]\| \subseteq \|A[\rho]\| \qquad \text{for all valuations } \rho$$

(Implies $|A[\rho]| \subseteq |B[\rho]|$)

More adequate typing/subtyping rules

$$\frac{}{A \leq A} \qquad \frac{A \leq B \qquad B \leq C}{A \leq C} \qquad \frac{\Gamma \vdash t : A \qquad A \leq B}{\Gamma \vdash t : B}$$

$$\frac{}{\forall x\, A \;\leq\; A\{x := e\}} \qquad \frac{}{\forall X\, A \;\leq\; A\{X := P\}}$$

$$\frac{A \leq B}{A \leq \forall x\, B}\ {\scriptstyle x \notin FV(A)} \qquad \frac{A \leq B}{A \leq \forall X\, B}\ {\scriptstyle X \notin FV(A)} \qquad \frac{A' \leq A \qquad B \leq B'}{A \Rightarrow B \;\leq\; A' \Rightarrow B'}$$

$$\frac{}{\forall x\,(A \Rightarrow B) \;\leq\; A \Rightarrow \forall x\, B}\ {\scriptstyle x \notin FV(A)} \qquad \frac{}{\forall X\,(A \Rightarrow B) \;\leq\; A \Rightarrow \forall X\, B}\ {\scriptstyle X \notin FV(A)}$$

- Example : $\forall X\, \forall Y\, (((X \Rightarrow Y) \Rightarrow X) \Rightarrow X) \;\leq\; \forall X\, (\neg\neg X \Rightarrow X)$

Realizing equalities

- Equality between individuals defined by

$$e_1 = e_2 \;\equiv\; \forall Z\,(Z(e_1) \Rightarrow Z(e_2)) \qquad \text{(Leibniz equality)}$$

Denotation of Leibniz equality

Given two closed first-order terms e_1, e_2 (and a pole $\perp\!\!\!\perp$)

$$\|e_1 = e_2\| \;=\; \begin{cases} \|\mathbf{1}\| \;=\; \{t\cdot\pi \;:\; (t\star\pi)\in\perp\!\!\!\perp\} & \text{if } [\![e_1]\!] = [\![e_2]\!] \\ \|\top\Rightarrow\perp\| \;=\; \Lambda\cdot\Pi & \text{if } [\![e_1]\!] \neq [\![e_2]\!] \end{cases}$$

writing $\mathbf{1} \equiv \forall Z\,(Z \Rightarrow Z)$ and $\top \equiv \dot{\varnothing}$

Corollary (Realizing true equations)

If $\mathscr{M} \models \forall\vec{x}\,(e_1(\vec{x}) = e_2(\vec{x}))$ (truth in the ground model)

then $\mathbf{I} \equiv \lambda z\,.\,z \Vdash \forall\vec{x}\,(e_1(\vec{x}) = e_2(\vec{x}))$ (universal realizability)

$\Rightarrow$ Defining equations of prim. rec. functions (universally) realized by $\mathbf{I}$

More connectives (1/2)

- Add binary intersection type :

Formulas $\quad A, B \quad ::= \quad \cdots \quad | \quad A \cap B \quad | \quad \top$

letting $\quad \|A \cap B\| = \|A\| \cup \|B\| \quad$ and $\quad \|\top\| = \varnothing$

so that $\quad |A \cap B| = |A| \cap |B| \quad$ and $\quad |\top| = \Lambda$

- Intersection type is a strong form of conjunction :

$$\lambda xz\,.\,z\,x\,x \;\Vdash\; A \cap B \Rightarrow A \wedge B$$

(but converse implication not realized in general)

- More typing / subtyping rules :

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash t : B}{\Gamma \vdash t : A \cap B} \qquad \frac{}{\Gamma \vdash t : \top}\ (FV(t) \subseteq \mathrm{dom}(\Gamma))$$

$$\frac{C \leq A \qquad C \leq B}{C \leq A \cap B} \qquad \frac{}{A \cap B \leq A} \qquad \frac{}{A \cap B \leq B} \qquad \frac{}{A \leq \top}$$

$$\frac{}{(A \Rightarrow B) \cap (A \Rightarrow C) \;\leq\; A \Rightarrow (B \cap C)} \qquad \frac{}{\top \;\leq\; A \Rightarrow \top}$$

More connectives (2/2)

- Add equational implication :

Formulas $A, B \quad ::= \quad \cdots \quad | \quad e_1 = e_2 \mapsto A$

Let $$\|e_1 = e_2 \mapsto A\| \;=\; \begin{cases} \|A\| & \text{if } [\![e_1]\!] = [\![e_2]\!] \\ \varnothing & \text{if } [\![e_1]\!] \neq [\![e_2]\!] \end{cases}$$

Proposition (Equivalence of $e_1 = e_2 \mapsto A$ and $e_1 = e_2 \Rightarrow A$)

$$\lambda xy\,.\,y\,x \quad \Vdash \quad (e_1 = e_2 \mapsto A) \;\Rightarrow\; (e_1 = e_2 \Rightarrow A)$$
$$\lambda x\,.\,x\,\mathbf{I} \quad \Vdash \quad (e_1 = e_2 \Rightarrow A) \;\Rightarrow\; (e_1 = e_2 \mapsto A)$$

- **Example :** $e_1 \neq e_2 \;\equiv\; (e_1 = e_2 \mapsto \bot)$ (disequality)
  - Denotation of $e_1 \neq e_2$ much simpler than $\neg(e_1 = e_2)$
  - But $e_1 \neq e_2$ equivalent to $\neg(e_1 = e_2)$ (in the sense of realizability)

Realizing true Horn formulas

- Horn formula with $p+1$ literals :

$$H \equiv \forall \vec{x}\,(E_1(\vec{x}) \Rightarrow \cdots \Rightarrow E_p(\vec{x}) \Rightarrow F(\vec{x})) \qquad (p \geq 0)$$

where $E_i(\vec{x}) \equiv e_{i,1}(\vec{x}) = e_{i,2}(\vec{x})$ (for all $i \in [1..p]$)
and either

- $F(\vec{x}) \equiv e_1(\vec{x}) = e_2(\vec{x})$, or (positive Horn formula)
- $F(\vec{x}) \equiv e_1(\vec{x}) \neq e_2(\vec{x})$ (negative Horn formula)

Theorem (Realizing true Horn formulas)

If $\mathscr{M} \models H$, then

$$\mathbf{I} \equiv \lambda z\,.\,z \Vdash H \quad \text{(if } H \text{ positive)}$$
$$\lambda x_1 \cdots x_p\,.\,x_1\,(\cdots(x_p\,\mathbf{I})\cdots) \Vdash H \quad \text{(if } H \text{ negative)}$$

Corollary (Realizing Peano 3rd and 4th axioms)

$$\mathbf{I} \Vdash \forall x\,\forall y\,(s(x) = s(y) \Rightarrow x = y)$$
$$\mathbf{I} \Vdash \forall x\,(s(x) \neq 0)$$

Program extraction

- All Peano axioms are universally realizable, but induction :

$$\not\Vdash \ \forall x\,(x \in \mathbb{N}) \qquad \text{(cf next lecture)}$$

Extracting programs from proofs in PA2

If $\mathrm{PA2} \vdash A$, then there is $t \in \mathrm{PL}$ such that $t \Vdash A^{\mathbb{N}}$

($A^{\mathbb{N}}$ obtained from A by relativizing all 1st-order quantifications to $\mathbb{N}$)

- **Practical extraction :**
  - Only apply adequacy theorem to computationally relevant parts of the proof, use 'default' realizers everywhere else (i.e. Horn formulas)

    $\rightsquigarrow$ Realizer optimization

- **Example :**

$$\lambda xyznuvw\,.\,u\,(v\,(w\,\mathbf{I})) \ \Vdash$$
$$(\forall x, y, z, n \in \mathbb{N})\ (x \geq 1 \Rightarrow y \geq 1 \Rightarrow n \geq 3 \Rightarrow x^n + y^n \neq z^n)$$

Plan

Some problems of classical realizability

1. **The specification problem**

   Given a formula A, characterize its universal realizers
   from their computational behavior

   $\rightsquigarrow$ cf Mauricio Guillermo's talk

2. **Extracting witnesses from classical realizers**

   $\rightsquigarrow$ cf next slide

3. **Realizability algebras + Cohen forcing**

   Realizability algebras : a program to well-order $\mathbb{R}$ [Krivine 2011]
   Forcing as a program transformation [Miquel 2011]

   $\rightsquigarrow$ cf Thomas Streicher's talk

4. **Studying the models induced by classical realizability**

   Which interesting formulas are realized in $\mathscr{M}_{\perp\!\!\!\perp}$
   that are not already true in the ground model $\mathscr{M}$?

   Realizability algebras II : new models of ZF + DC [Krivine 2011]

   $\rightsquigarrow$ cf next lecture (tomorrow)

The problem of witness extraction

- **Problem :** Extract a witness from a universal realizer (or a proof)

$$t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$$

i.e. some $n \in \mathbb{N}$ such that $A(n)$ is true

- This should not be always possible !

$$t_0 \Vdash (\exists x \in \mathbb{N})\, ((x = 1 \wedge C) \vee (x = 0 \wedge \neg C))$$

(C = Continuum hypothesis, Goldbach's conjecture, etc.)

- Possible tradeoffs :
  - Intuitionistic logic : restrict the shape of the realizer / proof t_0
  - Classical logic : restrict the shape of the formula $A(x)$

Storage operators

- Add call-by-value implication :

Formulas $A, B \quad ::= \quad \cdots \quad | \quad \{e\} \Rightarrow A$

letting $\quad \|\{e\} \Rightarrow A\| \;=\; |\overline{[\![e]\!]}| \cdot \|A\|$
$\qquad\qquad\qquad\qquad\quad\; = \; \{\overline{n} \cdot \pi \;:\; n = [\![e]\!],\ \pi \in \|A\|\}$

- From the definition : $\quad e \in \mathbb{N} \Rightarrow A \;\leq\; \{e\} \Rightarrow A$

  so that : $\quad \mathbf{I} \Vdash \forall x \forall Z\,((x \in \mathbb{N} \Rightarrow Z) \Rightarrow (\{x\} \Rightarrow Z))$ (direct coercion)

- Storage operator = proof-like term M such that :

$$M \Vdash \forall x \forall Z\,((\{x\} \Rightarrow Z) \Rightarrow (x \in \mathbb{N} \Rightarrow Z)) \qquad \text{(converse coercion)}$$

Theorem : Storage operators exist, e.g. $\quad M \equiv \lambda fn\,.\,n\,f\,(\lambda hx\,.\,h\,(\overline{s}\,x))\,\overline{0}$

- **Conclusion :** $\quad e \in \mathbb{N} \Rightarrow A \quad$ and $\quad \{e\} \Rightarrow A \quad$ interchangeable

Computing using storage operators : an example

- Given a k-ary function f, write :

$$\begin{array}{lcl} \mathrm{Total}(f) & \equiv & (\forall x_1 \in \mathbb{N}) \cdots (\forall x_k \in \mathbb{N})\, (f(x_1, \ldots, x_k) \in \mathbb{N}) \\ \mathrm{Comput}(f) & \equiv & \forall x_1 \cdots \forall x_k \, \forall Z \quad [\{x_1\} \Rightarrow \cdots \Rightarrow \{x_k\} \Rightarrow \\ & & \qquad (\{f(x_1, \ldots, x_k)\} \Rightarrow Z) \Rightarrow Z] \end{array}$$

Theorem (Specification of the formula Comput(f))

For every term $t \in \Lambda$ the following are equivalent :

1. $t \Vdash \mathrm{Comput}(f)$
2. t computes f : for all $(n_1, \ldots, n_k) \in \mathbb{N}^k$, $u \in \Lambda$, $\pi \in \Pi$:

$$t \star \overline{n}_1 \cdots \overline{n}_k \cdot u \cdot \pi \quad \succ \quad u \star \overline{f(n_1, \ldots, n_k)} \cdot \pi$$

- Using a storage operator M, one can build proof-like terms

$$\begin{array}{lcll} \xi_k & \Vdash & \mathrm{Total}(f) & \Rightarrow \quad \mathrm{Comput}(f) \\ \xi'_k & \Vdash & \mathrm{Comput}(f) & \Rightarrow \quad \mathrm{Total}(f) \end{array}$$

Exercise : Implement ξ_k and ξ'_k from M

Naive extraction

- A classical realizer $t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$ always evaluates to a pair witness/justification

Naive extraction

If $t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$, then :

$$t_0 \star M\,(\lambda xy\,.\,\mathrm{stop}\,x\,y)\cdot\pi \quad \succ \quad \mathrm{stop} \star \overline{n}\cdot u\cdot\pi$$

for some $n \in \mathbb{N}$ and $u \Vdash A(n)$ (w.r.t. a particular pole $\perp\!\!\!\perp$)

- But $n \in \mathbb{N}$ might be a false witness because the justification $u \Vdash A(n)$ is cheating ! (i.e. u might contain hidden continuations)
- In the case where t_0 comes from an intuitionistic proof, extracted witness $n \in \mathbb{N}$ is always correct

Extraction in the Σ^0_1-case

Extraction in the Σ^0_1-case

If $t_0 \Vdash (\exists x \in \mathbb{N})\,(f(x) = 0)$, then

$$t_0 \star M\,(\lambda xy\,.\,y\,(\mathsf{stop}\,x)) \cdot \pi \quad \succ \quad \mathsf{stop} \star \overline{n} \cdot \pi$$

for some $n \in \mathbb{N}$ such that $f(n) = 0$

- Storage operator M used to evaluate 1st component
- 2nd component (y) used as a breakpoint
  (Relies on the particular structure of equality realizers)
- Holds independently from the instruction set
- Supports any representation of numerals
  (one have to implement the storage operator M accordingly)

Extraction in the Σ^0_n-case

- Assume given a conditional refutation of $A(x)$
  i.e. a term $r_A \in \Lambda$ such that for all $n \in \mathbb{N}$:

$$\text{If} \quad \mathscr{M} \not\models A(n), \quad \text{then} \quad r_A \overline{n} \Vdash \neg A(n)$$

  - Exist for all formulas of 1st-order arithmetic [Krivine]

- Consider an instruction print with the rule

$$\text{print} \star \overline{n} \cdot u \cdot \pi \quad \succ \quad u \star \pi \qquad \text{(formal specification)}$$

  while printing n on some device (informal specification)

Kamikaze extraction

If $t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$, then :

1. The process $t_0 \star M\,(\lambda xy\,.\,\text{print}\, x\,(r_A\, x\, y)) \cdot \pi$ prints a correct witness after finitely many evaluation steps
2. May do anything after having found the correct witness...
   (crash / loop / print dummy witnesses, ...)

Krivine's realizability *vs* Friedman's *A*-translation

- Krivine's realizability can seen as the composition of Friedman's *A*-translation with Kleene's realizability

$$\text{CPS} \circ \text{Krivine} \quad = \quad \text{Kleene} \circ \text{Friedman} \qquad \text{[Oliva-Streicher'08]}$$

The dictionary

Classical realizability	**Friedman's *A*-translation**
Pole $\perp\!\!\!\perp$	Return formula R
Falsity value $\|\|A\|\|$	Negative translation $A^\perp$
$\|\|A \Rightarrow B\|\| = \|A\| \cdot \|\|B\|\|$	$(A \Rightarrow B)^\perp \equiv A^{\neg\neg} \wedge B^\perp$
Truth value $\|A\| = \|\|A\|\|^{\perp\!\!\!\perp}$	$A^{\neg\neg} \equiv A^\perp \Rightarrow R$

- Through the CPS translation, Krivine's extraction method in the Σ^0_1-case is exactly Friedman's trick [Miquel'11]