Classical realizability for the Calculus of Constructions

Alexandre Miquel
Plume team – LIP/ENS Lyon

February 14th, 2011
$\pi\mathbf{r}^2$ team – INRIA Paris-Rocquencourt

Motivation

Bridging different conceptions of the Curry-Howard correspondence:

The calculi of constructions (1985–)

- Type theory, intuitionistic logic, proof-theoretically very strong
- Strong evaluation, relies on confluence (conversion)
- Automated extraction (intuitionistic only)
- Long-term collaborative effort, successful implementation (Coq)
- Big machinery, difficult to extend

Krivine's classical realizability (1990–)

- Realizability, classical logic, defined in 2nd order arithmetic (PA2)
- Weak head evaluation, no need for confluence
- Supports the axiom of dependent choices
- Creation of a single mathematician (elitist conception)
- Light machinery, easy to extend (for those who understand it)

Plan

Plan

Classical realizability: the language of realizers

The language λ_c (Krivine)

Terms $t, u ::= \underbrace{x \mid \lambda x\,.\,t \mid tu \mid \overbrace{\text{cc} \mid \cdots}^{\text{instructions}}}_{\text{quasi-proofs}} \mid \mathsf{k}_\pi$

Stacks $\pi ::= \alpha \mid u \cdot \pi$ (u, π closed)

Processes $p, q ::= t \star \pi$ (t, π closed)

Evaluation rules (Krivine's Abstract Machine)

(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)	cc	$\star$	$t \cdot \pi$	$\succ$	t	$\star$	$\mathsf{k}_\pi \cdot \pi$
(Restore)	k_π	$\star$	$t \cdot \pi'$	$\succ$	t	$\star$	π
		$\cdots$				$\cdots$	

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
- 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:

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

Semantics of $\Rightarrow$ and $\forall$

- The point of view of stacks (opponents):

$$\|A \Rightarrow B\|_\rho \quad = \quad |A|_\rho \cdot \|B\|_\rho \quad = \quad \{t \cdot \pi \ : \ t \in |A|_\rho, \quad \pi \in \|B\|_\rho\}$$

$$\|\forall x \ A\|_\rho \quad = \quad \bigcup_{v \in D} \|A\|_{\rho, x \leftarrow v} \qquad (D = \text{domain of quantification})$$

- The point of view of terms (truth value):

$$\text{Def:} \qquad |A|_\rho \quad = \quad \{t \in \Lambda \ : \ \forall \pi \in \|A\| \quad t \star \pi \in \perp\!\!\!\perp\}$$

$$|A \Rightarrow B|_\rho \quad \subseteq \quad |A|_\rho \to |B|_\rho$$

$$|\forall x \ A|_\rho \quad = \quad \bigcap_{v \in D} |A|_{\rho, x \leftarrow v}$$

- Notation: $t \Vdash A \quad \equiv \quad t \in |A|$ (w.r.t. a fixed pole $\perp\!\!\!\perp$)

Properties of realizability

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

Example:

$$\begin{array}{lllll} cc & \star & t \cdot \pi & \succ & t \star k_\pi \cdot \pi \\ k_\pi & \star & t \cdot \pi' & \succ & t \star \pi \end{array}$$

1. If $\pi \in \|A\|$, then $k_\pi \in |A \Rightarrow B|$
2. $cc \in |((A \Rightarrow B) \Rightarrow A) \Rightarrow A|$

- As for BHK semantics (Kleene's realizability), Krivine's semantics is compatible with the typing rules of AF2:

Proposition (Adequacy)

If $x_1 : A_1, \ldots, x_n : A_n \vdash t : B$ (in AF2)
then for all poles $\perp\!\!\!\perp$ and for all realizers $u_1 \Vdash A_1$, ..., $u_n \Vdash A_n$
we have

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

Provability and realizability

- In 2nd-order arithmetic:
  - All provable formulæ of 2nd-order logic are realized (adequacy), including the (relativized) induction principle
  - Other Peano axioms have (very) simple realizers
  - We can realize the axiom of dependent choices [Krivine'03] using a suitable extra instruction ('quote' or 'clock')
  - Witness extraction techniques for Σ^0_1/Π^0_2 formulas [Miquel'10]
- Classical realizability model of PA2 extends to:
  - Higher-order arithmetic (PAω) [Raffalli, Miquel]
  - Zermelo-Fraenkel set theory (ZF) [Krivine'01]
  - The calculus of constructions with universes [Miquel'07]
- Classical realizability is compatible with forcing [Krivine'08,'09,'10]

Plan

From ω-sets to Π-sets (1/2)

- Intuitionistic realizability model of CC based on Hyland's notion of a ω-set: [Longo-Moggi, Luo]

ω-sets

An ω-set is a pair $A = \langle |A|, \Vdash_A \rangle$ formed by

- A carrier $|A|$ (carrier)
- A binary relation $n \Vdash_A x$ $(n \in \omega,\ x \in |A|)$

- Generalized to:
  - Arbitrary PCAs (D-sets) [Streicher]
  - Saturated sets (Λ-sets) [Altenkirch]
- Compatible with Prop/Set impredicative

From ω-sets to Π-sets (2/2)

Π-sets

A Π-set is a pair $A = \langle |A|, \perp_A \rangle$ formed by

- A set $|A|$ (carrier)
- A binary relation $x \perp_A \pi$ $(x \in |A|, \pi \in \Pi)$

Notation: $A(x) = \{\pi \in \Pi \ : \ x \perp_A \pi\}$

- Realizability parameterized by a pole $\mathbb{\perp}$:

$$\begin{array}{rcl} t \Vdash x \in A & \equiv & t \in (A(x))^{\mathbb{\perp}} \\ & \equiv & \forall \pi \ (x \perp_A \pi \quad \Rightarrow \quad t \star \pi \in \mathbb{\perp}) \end{array}$$

- A Π-set A is coarse when $\perp_A = \varnothing$ (computational irrelevance)
  - By orthogonality: $t \Vdash x \in A$ for all $t \in \Lambda$
  - Given a set E, we write: $\mathrm{coarse}(E) = \langle E, \varnothing \rangle$

Interpreting dependent products

- Given
  - a Π-set A
  - a family of Π-sets $(B_x)_{x\in|A|}$,

  the dependent product $\prod_{x\in A} B_x$ is the Π-set defined by

$$\left|\prod_{x\in A} B_x\right| = \prod_{x\in|A|} |B_x| \qquad \text{(set-theoretic, no restriction)}$$

$$\left(\prod_{x\in A} B_x\right)(f) = \{t\cdot\pi \ : \ \exists x\in|A| \ (t \Vdash x\in A \ \wedge \ f(x)\perp_{B_x}\pi)\}$$

- Remarks:
  - Mixture of $\forall$ $(\exists x\in|A|\ldots)$ and $\Rightarrow$ $(t\Vdash\cdots$ and $\cdots\perp_{B_x}\pi)$
  - The definition of $\prod_{x\in A} B_x$ depends on the pole $\perp\!\!\!\perp$

Interpreting propositions

- A Π-set A is degenerated if $|A| = \{\bullet\}$ (proof-irrelevance)
  - A is fully determined by $A(\bullet) \subseteq \Pi$ (falsity value)
  - The set of all degenerated Π-sets is isomorphic to $\mathfrak{P}(\Pi)$

- Using Aczel's encoding of functions, we can identify every constant function $(x \in E \mapsto \bullet)$ with the proof-object $\bullet$

Impredicativity of degenerated Π-set

The product of a family of degenerated Π-sets $(B_x)_{x \in |A|}$ indexed by an arbitrary Π-set A is a degenerated Π-set

- **Particular case:** implication $A \Rightarrow B$ (A, B degenerated)

$$(A \Rightarrow B)(\bullet) \;=\; (A(\bullet))^{\perp\!\!\!\perp} \cdot B(\bullet)$$

- We let $\llbracket \mathrm{Prop} \rrbracket \;=\; \mathrm{coarse}\big\{\langle \{\bullet\},\ \{\bullet\} \times S\rangle \;:\; S \in \mathfrak{P}(\Pi)\big\}$

Interpreting the hierarchy of predicative universes

- Given a set of sets $\mathcal{U}$, we write

$$\mathcal{U}^{(\Pi)} \;=\; \{A \in \Pi\text{-set} \;:\; |A| \in \mathcal{U}\}$$

(set of all Π-sets whose carrier is in $\mathcal{U}$)

- Given an increasing sequence of inaccessible cardinals $(\mu_i)_{i\geq 1}$ we let

$$[\![\mathsf{Type}_i]\!] \;=\; \mathsf{coarse}((V^*_{\mu_i})^{(\Pi)})$$

writing $V^*_{\mu_i} = V_{\mu_i} \setminus \{\varnothing\}$ (using Veblen's hierarchy $(V_\alpha)_{\alpha\in On}$)

- **Remark:** To keep consistent w.r.t. Krivine realizability, we only allow in the model Π-sets with a nonempty carrier

Building the model

- The interpretation $[\![M]\!]_{\perp\!\!\!\perp,\rho}$ of a term M of CC_ω depends on a fixed pole $\perp\!\!\!\perp$ and on a valuation ρ
- Propositions are interpreted as degenerated Π-sets

  Proof-terms are interpreted by $\bullet$
- Predicative universes are interpreted using large cardinals
- Abstraction/application interpreted set-theoretically

  + Aczel's trick to identify $(v \in |X| \mapsto \bullet)$ with $\bullet$

Proposition (Soundness)

If $\Gamma \vdash M : T$, then for all valuations $\rho \in [\![\Gamma]\!]_{\perp\!\!\!\perp}$:

1. $[\![T]\!]_{\perp\!\!\!\perp,\rho}$ is a Π-set
2. $[\![M]\!]_{\perp\!\!\!\perp,\rho} \in |[\![T]\!]_{\perp\!\!\!\perp,\rho}|$

Extraction and adequacy

- We define an extraction function $M \mapsto M^*$ from CC_ω to λ_c:

$$\begin{array}{rcl@{\qquad}rcl} x^* & = & x & \mathrm{Prop} & = & \text{any } \lambda_c\text{-term} \\ (\lambda x : T\,.\,M)^* & = & \lambda x\,.\,M^* & \mathrm{Type}_i^* & = & \text{any } \lambda_c\text{-term} \\ (MN)^* & = & M^* N^* & (\Pi x : T\,.\,U)^* & = & \text{any } \lambda_c\text{-term} \end{array}$$

Proposition (Adequacy)

If $x_1 : T_1, \ldots, x_n : T_n \vdash M : U$ (in $\mathrm{CC}_\omega^{\mathrm{irr}}$),

then for all $\rho \in [\![\Gamma]\!]$, for all $v_1 \in |[\![T_1]\!]_\rho|, \ldots, v_n \in |[\![T_n]\!]_\rho|$
and for all realizers $u_1 \Vdash v_1 \in [\![T_1]\!]_\rho, \ldots, u_n \Vdash v_n \in [\![T_n]\!]_\rho$

$$M^*\{x_1 := u_1; \ldots; x_n := u_n\} \Vdash [\![M]\!]_\rho \in [\![U]\!]_\rho$$

(independently from the choice of $\perp\!\!\!\perp$)

Proof irrelevance

- In the classical realizability model of CC_ω, we can interpret the typed equality judgment

$$\Gamma \vdash M_1 = M_2 : T$$

by $[\![M_1]\!] = [\![M_2]\!]$ (equality of denotations)

- The usual inference rules for equality are sound and adequate, including the rule of proof-irrelevance:

$$\frac{\Gamma \vdash M_1 : T \qquad \Gamma \vdash M_2 : T \qquad \Gamma \vdash T : \mathrm{Prop}}{\Gamma \vdash M_1 = M_2 : T}$$

- In this system, we can give a proof-term for

$$\Pi A : \mathrm{Prop} \,.\, \Pi x, y : A \,.\, x =_A y$$

(where $=_A$ stands for Leibniz equality on A)

What we **cannot** (?) realize

- The law of Peirce in Type_i

$$\not\Vdash \quad \Pi A, B : \mathrm{Type}_i \,.\, ((A \to B) \to A) \to A$$

  - Similarly for the excluded middle in Type_i

- The equivalence between total functional relations and functions:

$$\not\Vdash \quad \Pi A, B : \mathrm{Type}_i \,.\, \Pi R : (A \to B \to \mathrm{Prop}) \,. \\ \mathrm{Functional}\, R \to \mathrm{Total}\, R \to \\ \Sigma f : (A \to B) \,.\, \Pi x : A \,.\, R\, x\, (f\, x)$$

  - Intuitions:
    - Functions $\subsetneq$ FuncRel $\cap$ TotalRel
    - Functions $f : A \to B$ remain computable, at least in some sense
    - Total functional relations represent the classical notion of function, including (for instance) the Halting function

Plan

Example: adding a constant for Peirce's law

- Let us add a constant Peirce with the typing rule:

$$\text{Peirce} \ : \ \Pi A, B : \text{Prop} \,.\, ((A \to B) \to A) \to A$$

- In the model, we interpret the constant Peirce by $[\![\text{Peirce}]\!] = \bullet$ and we extend the extraction function $M \mapsto M^*$ by

$$(\text{Peirce})^* \ = \ \lambda\_, \_ \,.\, \mathfrak{cc}$$

- This extension is both sound and adequate

- **Remark:** The constant Peirce is given no computation rule!
  - This is not necessary, since the system is proof irrelevant
  - The computational strengh of the law of Peirce is only activated through the extraction function $M \mapsto M^*$ (in λ_c)

Enriching CC_ω with new constants (1/2)

- To interpret a new constant $c : T$ (with an equational theory), we must define in the model:
  - A denotation $[\![c]\!] \in |[\![T]\!]|$ (satisfying the equational theory)
  - A realizer $c^* \Vdash [\![c]\!] \in [\![T]\!]$ (i.e. $c^* \in ([\![T]\!]([\![c]\!]))^{\perp\!\!\perp}$)
  - No other constraint on c^*: computational transparency
- **Example:** If $c : T$ is a type ($T \equiv \mathrm{Prop}/\mathrm{Type}_i$):
  - $[\![c]\!]$ must be a Π-set
    - whose carrier is the set-theoretic equivalent of c in the model (For instance: $|[\![\mathtt{nat}]\!]| = \mathbb{N}$)
    - whose local refutation relation defines the representation of data of type c in the extracted code (unary of binary natural numbers?)
  - $c^* =$ any closed λ_c-term (since $[\![T]\!]$ is a coarse Π-set)

Enriching CC_ω with new constants (2/2)

- If c is a function, for instance: $\texttt{plus} : \texttt{nat} \to \texttt{nat} \to \texttt{nat}$
  - The choice of $[\![c]\!]$ is (in general) dictated by the equations de c, here: $[\![\texttt{plus}]\!] = +_{\mathbb{N}}$
  - But we usually have several possible choices for c^*...

    (In the case of unary integers: do we define c^* by recursion on the first argument or by recursion on the second argument?)

- If $c : T$ is an axiom (T : Prop):
  - $[\![c]\!] = \bullet$ (no other possible choice)
  - $c^* =$ any quasi-proof $t \Vdash \bullet \in [\![T]\!]$ (if there is some)

- The same holds if c is a theorem ($c := M : T$: Prop)
  - We can choose: $c^* = M^*$... (default realizer)
  - Or we can take any other quasi-proof $c^* \Vdash \bullet \in [\![T]\!]$

    $\rightsquigarrow$ Allows to introduce optimized realizers

Example: unary natural numbers

We extend $\mathrm{CC}^{\mathrm{irr}}_{\omega} + \mathrm{Peirce}$ with the constants

$$\begin{array}{rcl} \mathsf{nat} & : & \mathsf{Type}_1 \qquad\qquad 0 \;:\; \mathsf{nat} \qquad\qquad \mathsf{S} \;:\; \mathsf{nat} \to \mathsf{nat} \\ \mathsf{nat\_ind} & : & \Pi X : \mathsf{nat} \to \mathsf{Prop} \,.\, (X0 \to \Pi y : \mathsf{nat} \,.\, (Xy \to X(\mathsf{S}\, y)) \to \Pi x : \mathsf{nat} \,.\, Xx) \\ \mathsf{nat\_rec}_i & : & \Pi X : \mathsf{nat} \to \mathsf{Type}_i \,.\, (X0 \to \Pi y : \mathsf{nat} \,.\, (Xy \to X(\mathsf{S}\, y)) \to \Pi x : \mathsf{nat} \,.\, Xx) \end{array}$$

interpreted and realized by:

$$[\![\mathsf{nat}]\!] = \langle \mathbb{N},\ \bot\!\!\!\bot_{\mathbb{N}} \rangle \quad \text{where} \quad n \bot\!\!\!\bot_{\mathbb{N}} \pi \ \text{ iff } \ \pi \in \|\mathsf{Nat}(n)\| \quad \text{(2nd-order encoding)}$$

$$[\![0]\!] = 0 \qquad [\![\mathsf{S}]\!] \;=\; (n \mapsto n+1) \qquad [\![\mathsf{nat\_rec}_i]\!] \;=\; \text{set-theoretic recursor}$$

$$\begin{array}{rcl} \mathsf{nat}^* & = & \text{any closed quasi-proof} \\ 0^* & = & \lambda xf \,.\, x \qquad\qquad \mathsf{S}^* \;=\; \lambda nxf \,.\, f(nxf) \\ \mathsf{nat\_ind}^* & = & \lambda \_xfn \,.\, n\,(\lambda z \,.\, z\, 0^* x)\,(\lambda p \,.\, p\ (\lambda myz \,.\, z\,(s^* m)\,(fmy)))\,(\lambda xy \,.\, y) \\ \mathsf{nat\_rec}_i^* & = & \mathsf{nat\_ind}^* \end{array}$$

(Expected equations for $\mathsf{nat\_rec}_i$ are true in the model)

Relating classical realizability in $\mathrm{CC}_\omega^{\mathrm{irr}}$ and in PA2

The common fragment

PA2
$$A, B \quad ::= \quad X(t_1, \ldots, t_n) \quad | \quad A \Rightarrow B \quad | \quad \forall x\,(\mathsf{Nat}(x) \Rightarrow A) \quad | \quad \forall X\,(\top \Rightarrow A)$$

$\mathrm{CC}_\omega^{\mathrm{irr}}$
$$A, B \quad ::= \quad X\ t_1\ \cdots\ t_n \quad | \quad A \to B \quad | \quad \Pi x : \mathsf{nat}\,.\,A \quad | \quad \Pi X : \mathsf{Prop}\,.\,A$$

Classical realizability in $\mathrm{CC}_\omega^{\mathrm{irr}}$ coincides with Krivine's realizability in PA2 on the common fragment:

Proposition

If A is a formula/proposition of the common fragment, then:

$$[\![A]\!]^{\mathrm{CC}_\omega^{\mathrm{irr}}} \quad = \quad \langle \{\bullet\},\ \{\bullet\} \times \|A\|^{\mathrm{PA2}} \rangle$$

$\equiv$ A has the same realizers in the two realizability models

In [CSL'07], we enrich the syntax of $\mathrm{CC}_\omega^{\mathrm{irr}}$ to get $\mathrm{PA2} \subset \mathrm{CC}_\omega^{\mathrm{irr}}$

Plan

Introducing primitive numerals in λ_c (1/2)

- We enrich the language λ_c with the following instructions:
  - For every $n \in \mathbb{N}$, an instruction $\hat{n}$ with no evaluation rule

    Intuition: $\hat{n} \star \pi \succ \texttt{segfault}$
  - For every primitive recursive function $f : \mathbb{N}^k \to \mathbb{N}$, an instruction $\check{f}$ with the evaluation rule

    $$\check{f} \star \hat{n}_1 \cdots \hat{n}_k \cdot u \cdot \pi \succ u \star \hat{m} \cdot \pi \qquad \text{where } m = f(n_1, \ldots, n_k)$$

    (We can do the same for other total or partial recursive functions)
  - An instruction null with the evaluation rule

    $$\text{null} \star \hat{n} \cdot u_0 \cdot u_1 \cdot \pi \succ \begin{cases} u_0 \star \pi & \text{if } n = 0 \\ u_1 \star \pi & \text{otherwise} \end{cases}$$

    (We can add similar instructions for other tests)

Introducing primitive numerals in λ_c (2/2)

- We enrich the language of formulas of PA2 with a new connnective $\{e\} \Rightarrow B$ interpreted in the classical realizability model (of PA2) by

$$\|\{e\} \Rightarrow B\| \;=\; \{\hat{n} \cdot \pi \;:\; n = [\![e]\!],\ \pi \in \|B\|\}$$

  - Intuition: $\{e\} \Rightarrow A$ is the type of all functions mapping the value of e (as a primitive numeral $\hat{n}$) to an object of type B

- Let $\mathrm{nat}'(x) \;\equiv\; \forall Z\,((\{x\} \Rightarrow Z) \Rightarrow Z)$
  - Intuitively: the type of lazy numerals of value x
  - For all $n \in \mathbb{N}$: $\lambda z\,.\,z\,\hat{n} \;\Vdash\; \mathrm{nat}'(n)$
  - We can realize the equivalence

$$\forall x\,(\mathrm{nat}(x) \;\Leftrightarrow\; \mathrm{nat}'(x))$$

  (using coercions between the two representations)

Introducing primitive numerals in the model

- We change the interpretation of natural numbers as follows:

$$[\![\mathsf{nat}]\!] = \langle \mathbb{N},\ \bot\!\!\!\bot_{\mathbb{N}} \rangle \qquad \text{where} \quad n \bot\!\!\!\bot_{\mathbb{N}} \pi \ \text{ iff } \ \pi \in \|\mathsf{Nat}'(n)\| \quad \text{(Lazy numerals)}$$

$$[\![0]\!] = 0 \qquad [\![\mathsf{S}]\!] = (n \mapsto n+1) \qquad [\![\mathsf{nat\_rec}_i]\!] = \text{set-theoretic recursor}$$

and update the corresponding realizers:

$$\mathsf{nat}^* = \text{any closed quasi-proof}$$

$$0^* = \lambda z\,.\,z\,\hat{0} \qquad \mathsf{S}^* = \lambda nz\,.\,n\,(\lambda n'\,.\,\check{s}\,n'\,z)$$

$$\mathsf{nat\_ind}^* = \lambda\_xfn\,.\,n\,(\lambda n'\,.\,\mathsf{null}\,n'\,x\,(\check{\mathsf{pred}}\,n'\,(\lambda p\,.\,f\,(\lambda z\,.\,zp)\,(\mathsf{nat\_ind}^*\,x\,f\,(\lambda z\,.\,zp)))))$$

$$\mathsf{nat\_rec}_i^* = \mathsf{nat\_ind}^*$$

Computational transparency

- Once the denotation $[\![c]\!] \in |[\![T]\!]|$ has been defined (and fulfils the accompanying equational theory), we can take for c^* any realizer $t \Vdash [\![c]\!] \in [\![T]\!]$.

- **Crucial point:** The realizer c^* does not necessarily have to compute (in λ_c) using the same rules as c (in CC_ω)

- **Example:** In Coq, addition and multiplication are defined by induction on the unary representation of natural numbers.

  But through the extraction function, we can let instead:

$$\begin{aligned} \mathsf{plus}^* &= \lambda nmz\,.\,n\,(\lambda n\,.\,m\,(\lambda m\,.\,\check{+}\,n\,m\,z)) \\ \mathsf{mult}^* &= \lambda nmz\,.\,n\,(\lambda n\,.\,m\,(\lambda m\,.\,\check{\times}\,n\,m\,z)) \end{aligned}$$

- The same can be done for lemmas/theorems of stdlib

Example: commutativity of $+_{\mathsf{nat}}$ (default realizer)

```
Coq.Init.Datatypes.nat_rect =
  \P\f\f0 .fix_1_1 (\F\n Coq.Init.Datatypes.nat%case n f (\n f0 n (F n)))

Coq.Init.Datatypes.nat_ind =
  \P Coq.Init.Datatypes.nat_rect P

Coq.Init.Peano.plus_n_O =
  \n
    Coq.Init.Datatypes.nat_ind
      .type (Coq.Init.Logic.refl_equal .type (nat 0))
      (\n\IHn
         Coq.Init.Logic.f_equal
           .type .type Coq.Init.Datatypes.S n (Coq.Init.Peano.plus n (nat 0))
           IHn) n

Coq.Init.Peano.plus_n_Sm =
  \n\m
    Coq.Init.Datatypes.nat_ind
      .type (Coq.Init.Logic.refl_equal .type (Coq.Init.Datatypes.S m))
      (\n\IHn
         Coq.Init.Logic.f_equal
           .type .type Coq.Init.Datatypes.S
           (Coq.Init.Datatypes.S (Coq.Init.Peano.plus n m))
           (Coq.Init.Peano.plus n (Coq.Init.Datatypes.S m)) IHn) n

Coq.Arith.Plus.plus_comm =
  \n\m
    Coq.Init.Datatypes.nat_ind
      .type (Coq.Init.Peano.plus_n_O m)
      (\y\H
         Coq.Init.Logic.eq_ind
           .type (Coq.Init.Datatypes.S (Coq.Init.Peano.plus m y)) .type
           (Coq.Init.Logic.f_equal
              .type .type Coq.Init.Datatypes.S (Coq.Init.Peano.plus y m)
              (Coq.Init.Peano.plus m y) H)
           (Coq.Init.Peano.plus m (Coq.Init.Datatypes.S y))
           (Coq.Init.Peano.plus_n_Sm m y)) n
```

Example: commutativity of $+_{\mathsf{nat}}$ (optimized realizer)

```
Coq.Arith.Plus.plus_comm =
  \n\m\z z
```

Why optimizing realizers in Prop?

- In intuitionistic realizability, proof terms (sort Prop) can be completely dropped during the extraction process... [Letouzey'02]

  ... but this is no more possible in classical realizability!

- **Reason**: As in intuitionistic realizability, a classical realizer of a closed existential formula $\exists x : \mathsf{nat},\ A(x)$ contains both:
  - A natural number $n \in \mathbb{N}$ (witness)
  - A realizer $t \Vdash A(n)$ (justification)

- But the witness n may be wrong, in which case we need t to initiate backtracking (never trust a classical realizer!)

- Keeping all the information in Prop, we can implement several witness extraction techniques [Miquel'10]

Conclusion

- Krivine's classical realizability model extends to CC_ω
  - Realizability model based on Π-sets rather than on ω-sets
  - Incompatible with Set impredicative
- The classical realizability model of CC_ω coincides with Krivine's on the common fragment ($\approx$ PA2)
  - We can thus import classical realizability results from PA2: classical logic, axiom of dependent choices, witness extraction techniques, ...
- Classical reasoning confined in Prop
  - The predicative hierarchy remains intuitionistic
  - All the information in Prop is relevant! ($\neq$ Letouzey's extraction)
  - Allows witness extraction from classical existence proofs in Prop