Classical realizability for the Calculus of Constructions

Alexandre Miquel
Plume team – LIP/ENS Lyon

February 14th, 2011
πr² 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

1. Classical realizability

2. The Π-set model

3. Extensions

4. Optimizing realizers (or why realizability is useful for the hacker)
Plan

1. Classical realizability
2. The Π-set model
3. Extensions
4. Optimizing realizers (or why realizability is useful for the hacker)
Classical realizability: the language of realizers

The language $\lambda_c$ (Krivine)

| Terms          | $t, u ::= x | \lambda x . t | tu | \alpha | \cdots | k_\pi$ |
|----------------|--------------------------------------------------|
| Quasi-proofs   | $\forall u, \pi$ closed                         |
| Stacks         | $\pi ::= \alpha | u \cdot \pi$ (u, $\pi$ closed)         |
| Processes      | $p, q ::= t \star \pi$ (t, $\pi$ closed)       |

Evaluation rules (Krivine’s Abstract Machine)

<table>
<thead>
<tr>
<th>Rule</th>
<th>$tu \star \pi \Rightarrow t \star u \cdot \pi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Push)</td>
<td></td>
</tr>
<tr>
<td>(Grab)</td>
<td>$\lambda x . t \star u \cdot \pi \Rightarrow t{x:=u} \star \pi$</td>
</tr>
<tr>
<td>(Save)</td>
<td>$\alpha \star t \cdot \pi \Rightarrow t \star k_\pi \cdot \pi$</td>
</tr>
<tr>
<td>(Restore)</td>
<td>$k_\pi \star t \cdot \pi' \Rightarrow t \star \pi$</td>
</tr>
</tbody>
</table>

...
Classical realizability: principles

- **Intuitions:**
  - term = “proof” / stack = “counter-proof”
  - process = “contradiction” (slogan: never trust a classical realizer!)

- Classical realizability model parameterized by a pole \( \bot \)
  = 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:
  \[
  |A| = \|A\|^{\bot} = \{ t \in \Lambda : \forall \pi \in \|A\| \quad t \ast \pi \in \bot \} \]
Semantics of \( \Rightarrow \) and \( \forall \)

1. The point of view of stacks (opponents):

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

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

2. The point of view of terms (truth value):

**Def:**

\[
|A|_\rho = \{ t \in \Lambda : \forall \pi \in |A| \quad t \star \pi \in \bot \}
\]

\[
|A \Rightarrow B|_\rho \subseteq |A|_\rho \to |B|_\rho
\]

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

**Notation:**

\[
t \vdash A \iff t \in |A|\quad \text{(w.r.t. a fixed pole } \bot)\]
Properties of realizability

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

Example:
\[ \alpha \star t \cdot \pi \supset t \star k_\pi \cdot \pi \]
\[ k_\pi \star t \cdot \pi' \supset t \star \pi \]

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

- 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 \( \bot \) and for all realizers \( u_1 \vdash A_1, \ldots, u_n \vdash A_n \) 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 using a suitable extra instruction (‘quote’ or ‘clock’) [Krivine’03]
  - Witness extraction techniques for $\Sigma^0_1/\Pi^0_2$ formulas [Miquel’10]

- Classical realizability model of PA2 extends to:
  - Higher-order arithmetic (PA$\omega$) [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]
| 1 | Classical realizability |
| 2 | The Π-set model |
| 3 | Extensions |
| 4 | Optimizing realizers (or why realizability is useful for the hacker) |
From $\omega$-sets to $\Pi$-sets  (1/2)

- Intuitionistic realizability model of CC
  based on Hyland’s notion of a $\omega$-set:

  $\omega$-sets

  An $\omega$-set is a pair $A = \langle |A|, \models_A \rangle$ formed by:
  - A carrier $|A|$ (carrier)
  - A binary relation $n \models_A x$ ($n \in \omega, x \in |A|$)

- Generalized to:
  - Arbitrary PCAs ($D$-sets)
  - Saturated sets ($\Lambda$-sets)

- Compatible with Prop/Set impredicative

[Longo-Moggi, Luo]

[Streicher]

[Altenkirch]
From $\omega$-sets to $\Pi$-sets  \hspace{1cm} (2/2)

$\Pi$-sets

A $\Pi$-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 $\perp$:

  $t \vdash x \in A \equiv t \in (A(x))^\perp$

  $\equiv \forall \pi \ (x \perp_A \pi \Rightarrow t \star \pi \in \perp)$

- A $\Pi$-set $A$ is coarse when $\perp_A = \emptyset$ (computational irrelevance)

  - By orthogonality: $t \vdash x \in A$ for all $t \in \Lambda$
  - Given a set $E$, we write: coarse($E$) = $\langle E, \emptyset \rangle$
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

$$\left| \prod_{x \in A} B_x \right| = \prod_{x \in |A|} |B_x|$$

(set-theoretic, no restriction)

$$(\prod_{x \in A} B_x)(f) = \{ t \cdot \pi : \exists x \in |A| (t \vdash x \in A \land f(x) \perp_{B_x} \pi) \}$$

- 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 $\perp$
Interpreting propositions

- A $\Pi$-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 $\Pi$-sets is isomorphic to $\mathcal{P}(\Pi)$

- 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 $\Pi$-set**

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

- **Particular case:** implication $A \Rightarrow B$ ($A, B$ degenerated)
  \[(A \Rightarrow B)(\bullet) = (A(\bullet))^{\perp} \cdot B(\bullet)\]

- We let $\sem{\text{Prop}} = \text{coarse}\{\langle\{\bullet\}, \{\bullet\} \times S\} : S \in \mathcal{P}(\Pi)\}$
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 $\Pi$-sets whose carrier is in $\mathcal{U}$)

- Given an increasing sequence of inaccessible cardinals $(\mu_i)_{i \geq 1}$ we let
  \[ \llbracket \text{Type}_i \rrbracket = \text{coarse}((V_{\mu_i}^*)^{(\Pi)}) \]
  writing
  \[ V_{\mu_i}^* = V_{\mu_i} \setminus \{ \emptyset \} \]
  (using Veblen's hierarchy $(V_\alpha)_{\alpha \in \text{On}}$)

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

- The interpretation $\llbracket M \rrbracket_{\bot, \rho}$ of a term $M$ of $\mathbb{CC}_\omega$ depends on a fixed pole $\bot$ and on a valuation $\rho$.

- Propositions are interpreted as degenerated $\Pi$-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 $\rho \in \llbracket \Gamma \rrbracket_{\bot}$:

1. $\llbracket T \rrbracket_{\bot, \rho}$ is a $\Pi$-set
2. $\llbracket M \rrbracket_{\bot, \rho} \in |\llbracket T \rrbracket_{\bot, \rho}|$
We define an extraction function $M \mapsto M^*$ from $\text{CC}_\omega$ to $\lambda_c$:

- $x^* = x$
- $(\lambda x : T . M)^* = \lambda x . M^*$
- $(MN)^* = M^* N^*$
- $\text{Prop}^* = \text{any } \lambda_c\text{-term}$
- $\text{Type}_i^* = \text{any } \lambda_c\text{-term}$
- $(\Pi x : T . U)^* = \text{any } \lambda_c\text{-term}$

**Proposition (Adequacy)**

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

then for all $\rho \in \mathbb{G}$, for all $v_1 \in \llbracket T_1 \rrbracket_\rho, \ldots, v_n \in \llbracket T_n \rrbracket_\rho$

and for all realizers $u_1 \vdash v_1 \in \llbracket T_1 \rrbracket_\rho, \ldots, u_n \vdash v_n \in \llbracket T_n \rrbracket_\rho$

$$M^*\{x_1 := u_1; \ldots; x_n := u_n\} \vdash \llbracket M \rrbracket_\rho \in \llbracket U \rrbracket_\rho$$

(independently from the choice of $\bot$)
In the classical realizability model of $CC_\omega$, 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:

$$\Gamma \vdash M_1 : T \quad \Gamma \vdash M_2 : T \quad \Gamma \vdash T : Prop$$

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

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

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

(where $=_A$ stands for Leibniz equality on $A$)
What we **cannot** (?) realize

- The law of Peirce in Type\textsubscript{i}
  \[ \not\mathcal{V} \quad \Pi A, B : \text{Type}_i . ((A \to B) \to A) \to A \]

- Similarly for the excluded middle in Type\textsubscript{i}

- The equivalence between total functional relations and functions:
  \[ \not\mathcal{V} \quad \Pi A, B : \text{Type}_i . \Pi R : (A \to B \to \text{Prop}) . \\
  \text{Functional } R \to \text{Total } R \to \\
  \Sigma f : (A \to B) . \Pi x : A . R x (f x) \]

- 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

1. Classical realizability
2. The Π-set model
3. Extensions
4. Optimizing realizers (or why realizability is useful for the hacker)
Example: adding a constant for Peirce’s law

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

\[ \text{Peirce} : \prod A, B : \text{Prop} . ((A \rightarrow B) \rightarrow A) \rightarrow A \]

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

\[ (\text{Peirce})^* = \lambda_\_, \_. \alpha \]

- 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 strength of the law of Peirce is only activated through the extraction function \(M \mapsto M^*\) (in \(\lambda_c\))
Enriching $\mathbb{CC}_\omega$ with new constants  (1/2)

- To interpret a new constant $c : T$ (with an equational theory), we must define in the model:
  - A denotation $\llbracket c \rrbracket \in \llbracket T \rrbracket$ (satisfying the equational theory)
  - A realizer $c^* \vdash \llbracket c \rrbracket \in \llbracket T \rrbracket$ (i.e. $c^* \in (\llbracket T \rrbracket(\llbracket c \rrbracket))^\perp$)
  - No other constraint on $c^*$: computational transparency

**Example:** If $c : T$ is a type ($T \equiv \text{Prop}/\text{Type}_i$):

- $\llbracket c \rrbracket$ must be a $\Pi$-set
  - whose carrier is the set-theoretic equivalent of $c$ in the model (For instance: $\llbracket \text{nat} \rrbracket = \mathbb{N}$)
  - whose local refutation relation defines the representation of data of type $c$ in the extracted code (unary of binary natural numbers?)
- $c^* = \text{any closed } \lambda_c\text{-term}$ (since $\llbracket T \rrbracket$ is a coarse $\Pi$-set)
Enriching $\text{CC}_\omega$ with new constants (2/2)

- If $c$ is a function, for instance: $\text{plus} : \text{nat} \to \text{nat} \to \text{nat}$
  - The choice of $[c]$ is (in general) dictated by the equations de $c$, here: $[\text{plus}] = +_\mathbb{N}$
  - 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 ($T : \text{Prop}$):
  - $[c] = \bullet$ (no other possible choice)
  - $c^* = \text{any quasi-proof } t \vdash \bullet \in [T]$ (if there is some)

- The same holds if $c$ is a theorem ($c := M : T : \text{Prop}$)
  - 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 $CC^{irr}_\omega + \text{Peirce}$ with the constants

<table>
<thead>
<tr>
<th>nat</th>
<th>Type₁</th>
<th>0</th>
<th>nat</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>nat → nat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nat_ind</td>
<td>$\Pi X : \text{nat} → \text{Prop} \ (X 0 \rightarrow \Pi y : \text{nat} \ . \ (Xy \rightarrow X(Sy)) \rightarrow \Pi x : \text{nat} \ . \ Xx)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nat_recᵢ</td>
<td>$\Pi X : \text{nat} → \text{Typeᵢ} \ . \ (X 0 \rightarrow \Pi y : \text{nat} \ . \ (Xy \rightarrow X(Sy)) \rightarrow \Pi x : \text{nat} \ . \ Xx)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

interpreted and realized by:

$\lbrack\text{nat}\rbrack = \langle \mathbb{N}, \bot_{\mathbb{N}} \rangle$ where $n \downarrow_{\mathbb{N}} \pi$ iff $\pi \in \Vert \text{Nat}(n) \Vert$ (2nd-order encoding)

$\lbrack 0 \rbrack = 0 \quad \lbrack S \rbrack = (n \mapsto n + 1) \quad \lbrack \text{nat}_\text{rec}_i \rbrack = \text{set-theoretic recursor}$

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

$0^* = \lambda xf . x \quad S^* = \lambda n xf . f(n xf)$

$\text{nat}_\text{ind}^* = \lambda x fn . n (\lambda z . z 0^* x) (\lambda p . p (\lambda m y z . z (s^* m) (f m y))) (\lambda x y . y)$

$\text{nat}_\text{rec}_i^* = \text{nat}_\text{ind}^*$

(Expected equations for $\text{nat}_\text{rec}_i$ are true in the model)
Relating classical realizability in $\text{CC}^{\text{irr}}_\omega$ and in $\text{PA}_2$

**The common fragment**

\[
\begin{align*}
\text{PA}_2 & : A, B ::= X(t_1, \ldots, t_n) \mid A \Rightarrow B \\
& \quad \mid \forall x (\text{Nat}(x) \Rightarrow A) \mid \forall X (\top \Rightarrow A)
\end{align*}
\]

\[
\begin{align*}
\text{CC}^{\text{irr}}_\omega & : A, B ::= X \, t_1 \cdots t_n \mid A \rightarrow B \\
& \quad \mid \Pi x : \text{nat} . A \mid \Pi X : \text{Prop} . A
\end{align*}
\]

Classical realizability in $\text{CC}^{\text{irr}}_\omega$ coincides with Krivine’s realizability in $\text{PA}_2$ on the common fragment:

**Proposition**

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

\[
\begin{align*}
\llbracket A \rrbracket^{\text{CC}^{\text{irr}}_\omega} &= \langle \{ \bullet \}, \{ \bullet \} \rangle \times \| A \|^{\text{PA}_2} \\
\equiv & \quad A \text{ has the same realizers in the two realizability models}
\end{align*}
\]

In [CSL’07], we enrich the syntax of $\text{CC}^{\text{irr}}_\omega$ to get $\text{PA}_2 \subset \text{CC}^{\text{irr}}_\omega$
Plan

1. Classical realizability

2. The Π-set model

3. Extensions

4. Optimizing realizers (or why realizability is useful for the hacker)
Introducing primitive numerals in $\lambda_c$  (1/2)

- We enrich the language $\lambda_c$ with the following instructions:
  - For every $n \in \mathbb{N}$, an instruction $\hat{n}$ with no evaluation rule
    \[
    \hat{n} \star \pi \succ \text{segfault}
    \]
    Intuition:
    
  - For every primitive recursive function $f : \mathbb{N}^k \to \mathbb{N}$, an instruction $\tilde{f}$ with the evaluation rule
    \[
    \tilde{f} \star \hat{n}_1 \cdots \hat{n}_k \cdot u \cdot \pi \succ u \star \hat{m} \cdot \pi \quad \text{where } m = f(n_1, \ldots, n_k)
    \]
    (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 $\lambda_c$ (2/2)

- We enrich the language of formulas of PA2 with a new connective $\{e\} \Rightarrow B$ 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 $\text{nat}'(x) \equiv \forall Z ((\{x\} \Rightarrow Z) \Rightarrow Z)$

  - Intuitively: the type of lazy numerals of value $x$

  - For all $n \in \mathbb{N}$: $\lambda z. z \hat{n} \vdash \text{nat}'(n)$

  - We can realize the equivalence
    $$\forall x (\text{nat}(x) \Leftrightarrow \text{nat}'(x))$$

    (using coercions between the two representations)
Introducing primitive numerals in the model

- We change the interpretation of natural numbers as follows:

\[
\begin{align*}
\llbracket \text{nat} \rrbracket &= \langle \mathbb{N}, \bot_{\mathbb{N}} \rangle \\
\text{where } n \bot_{\mathbb{N}} \pi &\text{ iff } \pi \in \llbracket \text{Nat}'(n) \rrbracket \quad \text{(Lazy numerals)}
\end{align*}
\]

\[
\begin{align*}
\llbracket 0 \rrbracket &= 0 \\
\llbracket S \rrbracket &= (n \mapsto n + 1) \\
\llbracket \text{nat}_{\text{rec}} \rrbracket &= \text{set-theoretic recursor}
\end{align*}
\]

and update the corresponding realizers:

\[
\begin{align*}
\text{nat}^* &= \text{any closed quasi-proof} \\
0^* &= \lambda z. z \hat{0} \\
S^* &= \lambda n z. n (\lambda n'. \check{s} n' z) \\
\text{nat}_{\text{ind}}^* &= \lambda x f n (\lambda n' . \text{null} n' x (\check{p} \text{red} n' (\lambda p f (\lambda z z p) (\text{nat}_{\text{ind}}^* x f (\lambda z z p)))))) \\
\text{nat}_{\text{rec}}^*_i &= \text{nat}_{\text{ind}}^*
\end{align*}
\]
Computational transparency

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

- **Crucial point:** The realizer $c^*$ does not necessarily have to compute (in $\lambda_c$) using the same rules as $c$ (in $\text{CC}_\omega$).

- **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:

  $$\text{plus}^* = \lambda n m z. n (\lambda n . m (\lambda m . \hat{+} n m z))$$
  $$\text{mult}^* = \lambda n m z. n (\lambda n . m (\lambda m . \hat{\times} n m z))$$

- The same can be done for lemmas/theorems of stdlib.
Example: commutativity of $+_{\text{nat}}$

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_0 = \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.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

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

Coq.Init.Datatypes.nat_ind .type (Coq.Init.Peano.plus_n_0 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 $+_{\text{nat}}$  

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 $\exists x : \text{nat}, A(x)$ contains both:
  - A natural number $n \in \mathbb{N}$ (witness)
  - A realizer $t \models A(n)$ (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 $\mathbb{CC}_\omega$
  - Realizability model based on $\Pi$-sets rather than on $\omega$-sets
  - Incompatible with Set impredicative

- The classical realizability model of $\mathbb{CC}_\omega$ coincides with Krivine’s on the common fragment ($\approx \text{PA2}$)
  - 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