An introduction to Krivine realizability

Alexandre Miquel

UNIVERSIDAD DE LA REPUBLICA URUGUAY

FACULTAD DE INGENIERIA

EQUIPO DE LÓGICA UDELAR

November 3th, 10th, 17th & 24th, 2021

What is classical realizability?

- A complete reformulation of the principles of Kleene realizability to take into account classical reasoning [Krivine 1994, 2009, ...]
  - Based on Griffin's discovery about the connection between classical reasoning and control operators (call/cc) [Griffin 1990]

$$\text{call/cc} \;:\; ((A \Rightarrow B) \Rightarrow A) \Rightarrow A \qquad \text{(Peirce's law)}$$

  - Interprets the Axiom of Dependent Choices (DC) [K. 2003]

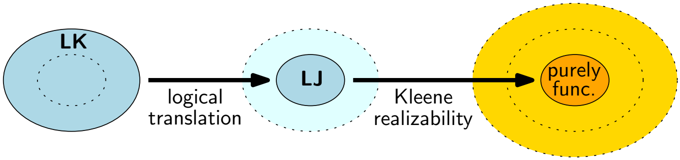
- Initially designed for PA2, but extends to:
  - Higher-order arithmetic (PAω)
  - Zermelo-Fraenkel set theory (ZF) [K. 2001, 2012]
  - The calculus of inductive constructions (CIC) [M. 2007]
    (with classical logic in Prop)

- Deep connections with Cohen forcing [K. 2011]

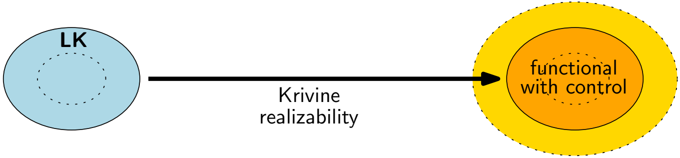
  $\rightsquigarrow$ can be used to define new models of PA2/ZF [K. 2012]

The design of classical realizability

- Traditionally, classical proofs are turned into intuitionistic proofs (via some translation/interpretation from LK into LJ) before being interpreted as purely functional programs



- Rather than restricting to LJ a priori, interpret classical proofs directly, using functional programs with control operators



Programming with continuations

- Control operators give to the programs the ability to capture their evaluation context (the "continuation"), so that they can backtrack when something goes wrong.

  $\rightsquigarrow$ Allow programs to use the method of trial and error.

- **Technically:** Extend the pure λ-calculus with a new binder $\mathcal{C}k\,.\,t$ that captures the current continuation in the bound variable k:

$$\frac{k : A \Rightarrow B \;\;\vdash\;\; t \;:\; A}{\vdash\;\; \mathcal{C}k\,.\,t \;:\; A}$$

  - The variable $k : A \Rightarrow B$ captures the current A-continuation, that is: the evaluation context asking for a value of type A.
  - When applied to an object of type A (the "new answer"), the A-continuation $k : A \Rightarrow B$ restores the evaluation context that was saved in k, with the new answer of type A. The current context is aborted, hence B can be any type (typically: $B \equiv \bot$).

Programming with continuations (2/2)

- In practice, the binder $\mathcal{C}k\,.\,t$ is implemented from the control operator œ ("call/cc"), letting $\mathcal{C}k\,.\,t \;\equiv\; \text{œ}\,(\lambda k\,.\,t)$.

  We have:
  $$\text{œ} \;:\; ((A \Rightarrow B) \Rightarrow A) \Rightarrow A \qquad \text{(Peirce's law)}$$
  $$\text{œ} \;:\; (\neg A \Rightarrow A) \Rightarrow A \qquad \text{(particular case: } B \equiv \bot)$$

- **Question:** $A \vee \neg A$?

- **Answer:** $$\mathbf{EM} \;\equiv\; \text{œ}\,(\lambda \overset{\neg(A\vee\neg A)}{k}\,.\,\mathbf{right}(\lambda \overset{A}{x}\,.\,k\,(\mathbf{left}\,x))) \;:\; A \vee \neg A$$

  where
  $$\mathbf{left} \;:\; \forall X\,\forall Y\,(X \Rightarrow X \vee Y)$$
  $$\mathbf{right} \;:\; \forall X\,\forall Y\,(Y \Rightarrow X \vee Y)$$

- Note that **EM** does not even need to know the formula A!

  It is actually polymorphic in A: $\mathbf{EM} \;:\; \forall X\,(X \vee \neg X)$

The role of paraproofs (metaphore)

Georges de la Tour. *Le tricheur à l'as de carreau* ($\sim$ 1636)

Plan

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

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

- Two kinds of variables
  - 1st-order vars: $x, y, z, \ldots$
  - 2nd-order vars: $X, Y, Z, \ldots$ 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:

  1. Each k-ary function symbol f is interpreted in $\mathbb{N}$ by a function
  $$f^{\mathbb{N}} \ : \ \mathbb{N}^k \to \mathbb{N}$$
  2. The signature Σ contains at least a function symbol for every primitive recursive function (0, s, pred, $+$, $-$, $\times$, $/$, mod, $\ldots$), each of them being interpreted the standard way

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

Formulas

- Formulas of minimal second-order logic

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

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

- Other connectives/quantifiers defined via second-order encodings:

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

Predicates

- Concrete relations are represented using predicates (syntactic sugar)

Predicates $\quad P, Q \quad ::= \quad \hat{x}_1 \cdots \hat{x}_k A_0 \quad$ (of arity $k \geq 0$)

Predicate = 2nd-order formula abstracted w.r.t. some 1st-order variables

Definition (Predicate application and 2nd-order substitution)

1. $P(e_1, \ldots, e_k)$ is the formula defined by

$$P(e_1, \ldots, e_k) \equiv A_0[x_1 := e_1, \ldots, x_k := e_k]$$

where $P \equiv \hat{x}_1 \cdots \hat{x}_k A_0$, and where $e_1, \ldots, e_k$ are k first-order terms

2. 2nd-order substitution $A[X := P]$ (where X and P are of the same arity k) consists to replace in the formula A every atomic subformula of the form

$$X(e_1, \ldots, e_k) \qquad \text{by the formula} \qquad P(e_1, \ldots, e_k)$$

- **Note:** Every k-ary 2nd-order variable X can be seen as a predicate:

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

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:

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

- Inclusion and extensional equality:

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

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

Natural deduction for classical 2nd-order logic (NK2)

Rules of system NK2

$$\frac{}{\Gamma \vdash A}\ {\scriptstyle A\in\Gamma} \qquad \frac{}{\Gamma \vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \qquad \frac{\Gamma \vdash A \Rightarrow B \qquad \Gamma \vdash A}{\Gamma \vdash B}$$

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

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

- From these rules, one can derive the introduction & elimination rules for $\bot$, $\wedge$, $\vee$, $\exists^1$, $\exists^2$, $=$ using their 2nd-order definition
- Classical logic obtained via Peirce's law: $((A \Rightarrow B) \Rightarrow A) \Rightarrow A$
- Elimination rule for 2nd-order $\forall$ implies all comprehension axioms:

$$\forall \vec{z}\ \forall \vec{Z}\ \exists X\ \forall \vec{x}\ [X(\vec{x}) \ \Leftrightarrow\ A(\vec{x}, \vec{z}, \vec{Z})]$$

A type system for classical 2nd-order logic (λNK2)

- Represent the computational contents of classical proofs using Curry-style proof terms, with call/cc for classical logic:

$$t, u \quad ::= \quad x \quad | \quad \lambda x\,.\,t \quad | \quad tu \quad | \quad \text{œ}$$

- **Typing judgement:** $\underbrace{x_1 : A_1, \ldots, x_n : A_n}_{\text{typing context } \Gamma} \vdash t : B$

Typing rules

$$\frac{}{\Gamma \vdash x : A}\;{\scriptstyle (x:A)\in\Gamma} \qquad \frac{}{\Gamma \vdash \text{œ} : ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}$$

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

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

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

Note: $\forall$ interpreted uniformly $\Rightarrow$ type checking & inference are undecidable

From the derivation to the proof term

- Deduction system NK2 and type system λNK2 are equivalent:

Theorem

$A_1, \ldots, A_n \vdash_{\mathrm{NK2}} A$ iff $x_1 : A_1, \ldots, x_n : A_n \vdash_{\lambda\mathrm{NK2}} t : A$ for some t

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{[\forall x\,(B(x) \Rightarrow C(x))]}\,g}{B(x) \Rightarrow C(x)} \qquad \dfrac{\dfrac{\dfrac{}{[\forall x\,(A(x) \Rightarrow B(x))]}\,f}{A(x) \Rightarrow B(x)} \qquad \dfrac{}{[A(x)]}\,u}{B(x)}\,@}{C(x)}\,@}{A(x) \Rightarrow C(x)}\,\lambda u}{\forall x\,(A(x) \Rightarrow C(x))}}{\forall x\,(B(x) \Rightarrow C(x)) \;\Rightarrow\; \forall x\,(A(x) \Rightarrow C(x))}\,\lambda g}{\forall x\,(A(x) \Rightarrow B(x)) \;\Rightarrow\; \forall x\,(B(x) \Rightarrow C(x)) \;\Rightarrow\; \forall x\,(A(x) \Rightarrow C(x))}\,\lambda f$$

$$\lambda f\,.\,\lambda g\,.\,\lambda u\,.\,g\,(f\,u)$$

Typing examples

- Intuitionistic principles:

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

- Excluded middle, double negation elimination:

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

- De Morgan laws:

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

Extensional equality

Recall that in (intuitionistic or classical) second-order logic:

- Equality between individuals (i.e. 1st-order objects) is defined by
$$e = e' \quad :\equiv \quad \forall Z\,(Z(e) \Rightarrow Z(e')) \qquad \text{(Leibniz equality)}$$
- Equality between predicates (i.e. 2st-order objects) is defined by:
$$P = Q \quad :\equiv \quad \forall \vec{x}\,(P(\vec{x}) \Leftrightarrow Q(\vec{x})) \qquad \text{(Extensional equality)}$$

Proposition (Extensionality in 2nd-order logic)

For each 2nd-order formula $A(X, \vec{z}, \vec{Z})$ depending on $X, \vec{z}, \vec{Z}$, we have:
$$\mathrm{NJ2} \vdash \forall \vec{z}\,\forall \vec{Z}\,\forall X\,\forall Y\,\left[X = Y \Rightarrow (A(X, \vec{z}, \vec{Z}) \Leftrightarrow A(Y, \vec{z}, \vec{Z}))\right]$$

Proof: By induction on the size of the formula $A(X, \vec{z}, \vec{Z})$ – Exercise!

Remark: The proposition holds because $X(e_1, \ldots, e_k)$ (predicate application) is the only construction of the language that involves 2nd-order variables X...

... but it does not hold anymore in higher-order formalisms: NK3, ..., NKω

Classical second-order arithmetic (PA2)

Classical 2nd-order arithmetic (PA2) is the (classical) 2nd-order theory whose axioms are:

- **Defining equations of all primitive recursive functions:**

$$\forall x\,(x+0=x) \qquad \forall x\,(x\times 0=0)$$
$$\forall x\,\forall y\,(x+s(y)=s(x+y)) \qquad \forall x\,\forall y\,(x\times s(y)=x\times y+x)$$

$$\forall x\,(\mathrm{pred}(0)=0) \qquad \forall x\,(x-0=0)$$
$$\forall x\,(\mathrm{pred}(s(x))=x) \qquad \forall x\,\forall y\,(x-s(y))=\mathrm{pred}(x-y) \qquad \text{etc.}$$

- **Peano axioms:**

(P3) $\forall x\,\forall y\,(s(x)=s(y)\Rightarrow x=y)$

(P4) $\forall x\,\neg(s(x)=0)$

(P5) $\forall Z\,[0\in Z\Rightarrow\forall y\,(y\in Z\Rightarrow s(y)\in Z)\Rightarrow\forall x\,(x\in Z)]$

Remark: Thanks to 2nd-order $\forall$, induction is now a single axiom:

$$\begin{aligned}\mathrm{Ind}\ &\equiv\ \forall Z\,[0\in Z\Rightarrow\forall y\,(y\in Z\Rightarrow s(y)\in Z)\Rightarrow\forall x\,(x\in Z)]\\ &\Leftrightarrow\ \forall x\,(x\in\mathbb{N}) \qquad \text{(“every individual is a natural number”)}\end{aligned}$$

The problem of induction (1/2)

- **Problem:** Induction axiom Ind $(\Leftrightarrow\ \forall x\,(x \in \mathbb{N}))$ is not realizable!
  (Due to uniform interpretation of $\forall$)
- Nevertheless, we observe that:

Proposition

The following formulas are derivable in $\mathrm{HA2}^-$ $(:=\ \mathrm{HA2} - \mathrm{Ind})$

$$\begin{array}{rcll}
\mathrm{NJ2} & \vdash & 0 \in \mathbb{N} & \\
\mathrm{NJ2} & \vdash & (\forall x \in \mathbb{N})(s(x) \in \mathbb{N}) & \\
\mathrm{NJ2} + \text{def. of } + & \vdash & (\forall x, y \in \mathbb{N})(x + y \in \mathbb{N}) & \\
\mathrm{NJ2} + \text{def. of } \times & \vdash & (\forall x, y \in \mathbb{N})(x \times y \in \mathbb{N}) & \text{(etc.)} \\
\mathrm{NJ2} & \vdash & \forall Z\,[0 \in Z \Rightarrow & \\
 & & \quad (\forall y \in \mathbb{N})\,(y \in Z \Rightarrow s(y) \in Z) \Rightarrow & \\
 & & \quad (\forall x \in \mathbb{N})\,(x \in Z)] &
\end{array}$$

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

Exercise: Write the corresponding proof-terms (in system λNK2)

The problem of induction (2/2)

- **Problem:** Induction axiom Ind $(\Leftrightarrow \forall x\,(x \in \mathbb{N}))$ is not realizable!
(Due to uniform interpretation of $\forall$)

- **Solution:** Restrict to $\mathrm{PA2}^- := \mathrm{PA2} - \mathrm{Ind}$
and relativize all 1st-order quantifications to $\mathbb{N}$:

Definition of the operation of relativization $A \mapsto A^{\mathbb{N}}$

$$(X(e_1,\dots,e_k))^{\mathbb{N}} :\equiv X(e_1,\dots,e_k) \qquad (\forall x\, A)^{\mathbb{N}} :\equiv (\forall x \in \mathbb{N}) A^{\mathbb{N}}$$
$$(A \Rightarrow B)^{\mathbb{N}} :\equiv A^{\mathbb{N}} \Rightarrow B^{\mathbb{N}} \qquad (\forall X\, A)^{\mathbb{N}} :\equiv \forall X\, A^{\mathbb{N}}$$

Theorem

If $\mathrm{PA2} \vdash A$, then $\mathrm{PA2}^- \vdash A^{\mathbb{N}}$

Proof: Exercise.

- **Conclusion:** In what follows, we shall work in $\mathrm{PA2}^- := \mathrm{PA2} - \mathrm{Ind}$,
relativizing 1st-order quantifications to $\mathbb{N}$ whenever needed

Two semantics for classical 2nd-order logic (1/3)

There are two semantics for (classical) 2nd-order logic:

Full semantics vs. **Henkin semantics**

Moreover, **full semantics** is a particular case of **Henkin semantics**

Full semantics: A full model $\mathscr{M}$ of LK2 is given by:

- A nonempty set $|\mathscr{M}|$ (domain of 1st-order objects)
- A function $f^{\mathscr{M}} : |\mathscr{M}|^k \to |\mathscr{M}|$ for each k-ary function symbol f
- A relation $f^{\mathscr{M}} \subseteq |\mathscr{M}|^k$ for each k-ary predicate symbol p

As usual, the interpretation of a 1st-order term (or a 2nd-order formula) is parameterized by a valuation (in $\mathscr{M}$), that is: a function ρ mapping

- each 1st-order variable x to an element $\rho(x) \in |\mathscr{M}|$, and
- each k-ary 2nd-order variable X to a relation $\rho(X) \in \mathfrak{P}(|\mathscr{M}|^k)$

The denotation of a 1st-order term e in a valuation ρ (notation: $e[\rho]^{\mathscr{M}}$) is defined as usual (i.e. Tarski semantics of 1st-order terms)

Two semantics for classical 2nd-order logic (2/3)

Full semantics (continued):

Definition of the satisfaction predicate $\mathscr{M} \models A[\rho]$

$\mathscr{M} \models \bot[\rho]$	never holds	
$\mathscr{M} \models p(e_1, \ldots, e_k)[\rho]$	iff	$(e_1[\rho]^{\mathscr{M}}, \ldots, e_k[\rho]^{\mathscr{M}}) \in p^{\mathscr{M}}$
$\mathscr{M} \models X(e_1, \ldots, e_k)[\rho]$	iff	$(e_1[\rho]^{\mathscr{M}}, \ldots, e_k[\rho]^{\mathscr{M}}) \in \rho(X)$
$\mathscr{M} \models (A \Rightarrow B)(\rho)$	iff	$\mathscr{M} \models A[\rho]$ implies $\mathscr{M} \models B[\rho]$
$\mathscr{M} \models (\forall x\, A)[\rho]$	iff	$\mathscr{M} \models A[\rho, x \leftarrow a]$ for all $a \in \lvert\mathscr{M}\rvert$
$\mathscr{M} \models (\forall X\, A)[\rho]$	iff	$\mathscr{M} \models A[\rho, X \leftarrow R]$ for all $R \in \mathfrak{P}(\lvert\mathscr{M}\rvert^k)$

Note that in the model, 2nd-order objects are all the possible relations $R \in \mathfrak{P}(|\mathscr{M}|^k)$.
So that when $|\mathscr{M}|$ is infinite, the model (1st- and 2nd-order objects) is uncountable.

- A full model of a 2nd-order (classical) theory $\mathscr{T}$ (for example: PA2) is a full model of LK2 that satisfies all the axioms of $\mathscr{T}$
- **Example:** The (full) standard model of PA2:

  $|\mathscr{M}| := \mathbb{N}, \quad s^{\mathscr{M}} := (n \mapsto n+1), \quad (+)^{\mathscr{M}} := (n, m \mapsto n+m) \quad$ (etc.)

Two semantics for classical 2nd-order logic (3/3)

Henkin semantics: A pre-Henkin model $\mathscr{M}$ of LK2 is given by:

- The same ingredients $(|\mathscr{M}|,\ f^k\ldots,\ p^k\ldots)$ as before, plus:
- A set of relations $\mathrm{Rel}_k(\mathscr{M}) \subseteq \mathfrak{P}(|\mathscr{M}|^k)$ (domain of 2nd-order objects)

Definition of the satisfaction predicate $\mathscr{M} \models A[\rho]$

$$\mathscr{M} \models (\forall X\, A)[\rho] \quad \text{iff} \quad \mathscr{M} \models A[\rho, X \leftarrow R] \quad \text{for all } R \in \mathrm{Rel}_k(|\mathscr{M}|)$$

(Other clauses of the definition remain unchanged)

Note that in the model, 2nd-order objects are only the relations $R \in \mathrm{Rel}_k(\mathscr{M})$. So that even when $|\mathscr{M}|$ is infinite, the model (1st- and 2nd-order objects) may be countable.

- A Henkin model of LK2 is a pre-Henkin model $\mathscr{M}$ that satisfies all comprehension axioms:

$$\mathscr{M} \models \forall\vec{z}\,\forall\vec{Z}\,\exists X\,\forall\vec{x}\,[X(\vec{x}) \Leftrightarrow A(\vec{x},\vec{z},\vec{Z})]$$

(where $A(x_1,\ldots,x_k,\vec{z},\vec{Z})$ is any formula with free vars. $\vec{z},\vec{Z},x_1,\ldots,x_k$)

- As before, a Henkin model of a 2nd-order theory $\mathscr{T}$ (for example: PA2) is a Henkin model of LK2 that satisfies all the axioms of $\mathscr{T}$

Full semantics vs Henkin semantics

- **Clearly:** Full semantics = Henkin semantics where

$$\mathrm{Rel}_k(\mathscr{M}) \;=\; \mathfrak{P}(|\mathscr{M}|^k) \qquad \text{(for all } k \geq 0\text{)}$$

Note: $\mathrm{Rel}_k(\mathscr{M}) = \mathfrak{P}(|\mathscr{M}|^k) \;\Rightarrow\; \mathscr{M}$ satisfies all (k-ary) comprehension axioms

- When designing a notion of model, we are in general interested in the properties of soundness, completenes and compactness.

  Regarding full and Henkin models, the situation is the following:

	Soundness	Completeness	Compactness
Full semantics	Yes	No	No
Henkin semantics	Yes	Yes	Yes

Intuition: The notion of full model is too restrictive, hence we lose the properties of completeness and compactness.

- The fact that Henkin models preserve completeness/compactness is due to the possibility of presenting 2nd-order logic as a 1st-order *theory*

Second-order logic as a first-order *theory* (1/2)

2nd-order logic over a 1st-order language $\mathcal{L}$ ($\mathrm{NK2}_{\mathcal{L}}$) can be presented as a multi-sorted 1st-order theory $\mathscr{T}_{\mathrm{LK2},\mathcal{L}}$ that is defined as follows:

- The language of $\mathscr{T}_{\mathrm{LK2},\mathcal{L}}$ has infinitely many sorts:
  - a sort ι of individuals, and
  - a sort o_k of k-ary relations, for each $k \geq 0$
- The function symbols of $\mathscr{T}_{\mathrm{LK2},\mathcal{L}}$ are all the (k-ary) function symbols of the language $\mathcal{L}$, now seen as function symbols of arity $\iota^k \to \iota$

  So that the terms of sort ι in $\mathscr{T}_{\mathrm{LK2},\mathcal{L}}$ are exactly the 1st-order terms of $\mathcal{L}$.
  On the other hand, the only terms of sort o_k are the variables $X : o_k$.
- A predicate symbol $@_k$ of arity $o_k \times \iota^k \to \mathrm{Prop}$
  (application of a k-ary predicate symbol to k individuals)
- The axioms of $\mathscr{T}_{\mathrm{LK2},\mathcal{L}}$ are all the comprehension axioms:

$$(\forall \vec{z} : \iota)(\forall \vec{Z} : o_*)(\exists X : o_k)(\forall \vec{x} : \iota)[@(X, \vec{x}) \ \Leftrightarrow \ A(\vec{x}, \vec{z}, \vec{Z})]$$

Second-order logic as a first-order *theory* (2/2)

There is an almost$^{(*)}$ one to one correspondence between the 2nd-order formulas of $\mathcal{L}$ and the 1st-order formulas of $\mathscr{T}_{\mathsf{LK2},\mathcal{L}}$:

Formulas of $\mathcal{L}$ (2nd-order)		**Formulas of $\mathscr{T}_{\mathsf{LK2},\mathcal{L}}$** (1st-order)
$X(e_1,\ldots,e_k)$	$\approx$	$@_k(X,e_1,\ldots,e_k)$
$\forall x\,A(x)$	$\approx$	$(\forall x:\iota)A(x)$
$\forall X\,A(X)$	$\approx$	$(\forall X:o_k)A(X)$

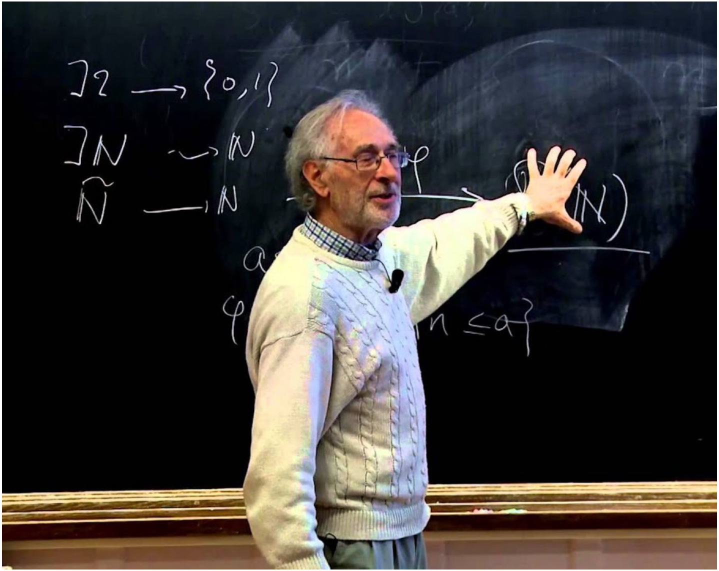
$(*)$ Up to the fact that $=$, $\top$, $\bot$, $\wedge$, $\vee$ and $\exists$ are defined in 2nd-order logic

Theorem

1. For each closed 2nd-order formula: $\underbrace{\mathsf{LK2}_{\mathcal{L}}\vdash A}_{\text{2nd-order}}$ iff $\underbrace{\mathscr{T}_{\mathsf{LK2},\mathcal{L}}\vdash A^{\text{1st}}}_{\text{1st-order}}$
2. The Henkin models of 2nd-order logic (in $\mathcal{L}$) are exactly the Tarski models of the 1st-order theory $\mathscr{T}_{\mathsf{LK2},\mathcal{L}}$ (up to insignificant changes of representation)

$\Rightarrow$ Henkin models enjoy soundness, completeness & compactness

Plan



Jean-Louis Krivine (1939–)

Terms, stacks and processes

- The syntax of the λ_c-calculus is parameterized by
  - A countable set $\mathcal{K} = \{\text{c}\!\!\!\text{c}, \ldots\}$ of instructions, containing at least the instruction $\text{c}\!\!\!\text{c}$ (call/cc)
  - A countable set Π_0 of stack constants (or stack bottoms)

Terms, stacks and processes

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

- A λ-calculus with two kinds of constants:
  - Instructions $\kappa \in \mathcal{K}$, including $\text{c}\!\!\!\text{c}$
  - Continuation constants k_π, one for every stack π (generated by $\text{c}\!\!\!\text{c}$)
- **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 $\quad t, u \quad ::= \quad x \mid \lambda x \,.\, t \mid tu \mid \kappa \quad (\kappa \in \mathcal{K})$

  - **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 are encoded as proof-like terms, letting:

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

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	œ	$\star$	$u \cdot \pi$	$\succ$	u	$\star$	$\mathrm{k}_\pi \cdot \pi$
Restore	k_π	$\star$	$u \cdot \pi'$	$\succ$	u	$\star$	π
		$\dots$				$\dots$	

(+ reflexivity & transitivity)

- Evaluation is not defined but axiomatized. The preorder $p \succ p'$ is just another parameter of the calculus, like the sets $\mathcal{K}$ and Π_0
- Extensible machinery: we 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:

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:
$$\begin{array}{rcl} (\lambda xy \,.\, t)\, u\, v \star \pi & \succ & \lambda xy \,.\, t \star u \cdot v \cdot \pi \\ & \succ & t[x := u][y := v] \star \pi \end{array}$$

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

Save	$\text{œ} \star u \cdot \pi$	$\succ$	$u \star \mathsf{k}_\pi \cdot \pi$
Restore	$\mathsf{k}_\pi \star u \cdot \pi'$	$\succ$	$u \star \pi$

  - The instruction œ is most often used in the pattern

$$\begin{array}{rcl} \text{œ}\, (\lambda k \,.\, t) \star \pi & \succ & \text{œ} \star (\lambda k \,.\, t) \cdot \pi \\ & \succ & (\lambda k \,.\, t) \star \mathsf{k}_\pi \cdot \pi \\ & \succ & t[k := \mathsf{k}_\pi] \star \pi \end{array}$$

Representing functions

Definition (function representation)

A partial function $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ is represented by a λ_c-term $\widehat{f} \in \Lambda$ if

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

for all $(n_1, \ldots, n_k) \in \mathrm{dom}(f)$ and for all $u \in \Lambda$, $\pi \in \Pi$

- Call by value encoding:
  - Consumes k values and returns 1 value on the stack
  - Control is given to the extra argument u (continuation, return block)
- Examples:

$$\begin{array}{rcl} \widehat{s} & := & \lambda xk\,.\,k\,(\bar{s}\,x) \\ \widehat{+} & := & \lambda xyk\,.\,y\,k\,(\lambda k'z\,.\,\widehat{s}\,z\,k)\,x \\ \widehat{\times} & := & \lambda xyk\,.\,y\,k\,(\lambda k'z\,.\,\widehat{+}\,z\,x\,k)\,\bar{0} \end{array}$$

Theorem (Representation of recursive functions)

All partial recursive functions are represented in the λ_c-calculus

Example of extra instructions (1/2)

- Numbering terms: the instruction quote:

$$\mathrm{quote} \star t \cdot u \cdot \pi \quad \succ \quad u \star \overline{\ulcorner t \urcorner} \cdot \pi$$

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

  - Useful to realize the axiom of dependent choices (DC) [K. 2003]

- Numbering stacks: the instruction quote′:

$$\mathrm{quote}' \star u \cdot \pi \quad \succ \quad u \star \overline{\ulcorner \pi \urcorner} \cdot \pi$$

where $\pi \mapsto \ulcorner \pi \urcorner$ is a fixed bijection from Π to $\mathbb{N}$

  - Can be implemented using quote
  - Useful to realize the axiom of dependent choices (DC) [K. 2003]

- Testing syntactic equality: the instruction eq:

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

  - Can be implemented using quote or quote′

Example of extra instructions (2/2)

- Non-deterministic choice operator: the instruction fork:

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

  - Useful for pedagogy – bad for realizability (collapses to forcing)

- The instruction stop:

$$\mathsf{stop} \star \pi \quad \not\succ$$

  - Stops execution. Final result returned on the stack π

- The instruction print:

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

and prints integer n on standard output (informal specification)

  - Useful to display intermediate results without stopping the machine

- The instruction hace_mate:

$$\mathsf{hace\_mate} \star u \cdot \pi \quad \succ \quad u \star \pi \quad + \quad \text{hace el mate}$$

On the determinism of evaluation

- A relation of evaluation $\succ$ (between processes) is deterministic when it is the reflexive-transitive closure of a relation $\succ^1$ of one step evaluation that is strictly deterministic, in the sense that:

$$p \succ^1 p' \quad \text{and} \quad p \succ^1 p'' \quad \text{implies} \quad p' \equiv p'' \qquad \text{(for all } p,\ p',\ p''\text{)}$$

- The relation of evaluation induced by the four basic rules (**Grab**, **Push**, **Save** and **Restore**) is clearly deterministic

  On the other hand β-reduction (in the λ-calculus) is not:

$$(\mathbf{I\,I})(\mathbf{I\,I}) \begin{cases} \to_\beta^1 & \mathbf{I}\,(\mathbf{I\,I}) \\ \to_\beta^1 & (\mathbf{I\,I})\,\mathbf{I} \end{cases}$$

- Instructions **quote**, **quote′**, **eq**, **stop**, **print** and **hace_mate** preserve the determinism of evaluation, while **fork** completely breaks it

- **Beware of non-determinism!** As soon as the calculus contains a term with the same evaluation rules as fork, the corresponding realizability model is equivalent to a forcing model (collapse)

Plan

Krivine the White

(Courtesy of Vincent Padovani)

Classical realizability: principles

- **Intuitions:**
  - term = "proof" / stack = "counter-proof"
  - process = "contradiction" (**Slogan:** Never trust a classical realizer!)

- Each classical realizability model is parameterized by a pole $\perp\!\!\!\perp$
  = set of processes ("contradictions") 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 $\|A\|$ is defined by induction on A (negative interp.)

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

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

More generally, given $S \subset \Pi$, we let $S^{\perp\!\!\!\perp} := \{t \in \Lambda \;:\; \forall \pi \in S \quad t \star \pi \in \perp\!\!\!\perp\}$ $(\subseteq \Lambda)$

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: the calculus of realizers
  - A saturated set of processes $\perp\!\!\!\perp \subseteq \Lambda \star \Pi$: the pole of the model
    (saturated = closed under anti-evaluation)
- Architecture:
  - First-order terms/variables are interpreted as natural numbers $n \in \mathbb{N}$
  - Formulas are interpreted as falsity values $S \in \mathfrak{P}(\Pi)$
  - k-ary second-order variables (and k-ary predicates) are 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 k-ary predicate constant $\dot{F}$ for every falsity function $F : \mathbb{N}^k \to \mathfrak{P}(\Pi)$

Interpreting closed formulas with parameters

Let A be a closed formula (with parameters)

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

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

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

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

Recall: For each $S\subseteq\Pi$ we write $S^{\perp\!\!\!\perp} := \{t\in\Lambda \;:\; \forall\pi\in S \quad t\star\pi\in\perp\!\!\!\perp\}$ $(\subseteq\Lambda)$

The realizability relation

Falsity value $\|A\|$ and truth value $|A|$ depend on the pole $\perp\!\!\!\perp$

$\rightsquigarrow$ write them (sometimes) $\|A\|_{\perp\!\!\!\perp}$ and $|A|_{\perp\!\!\!\perp}$ to recall the dependency

Realizability relations

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

$$t \Vvdash A \quad :\equiv \quad \forall\perp\!\!\!\perp \;\; t \in |A|_{\perp\!\!\!\perp} \qquad \text{(Universal realizability)}$$

From computation to realizability (1/4)

Fundamental idea: The computational behavior of a term determines the formulas it realizes:

Example 1: A closed term t is identity-like if:

$$t \star u \cdot \pi \quad \succ \quad u \star \pi \qquad\qquad \text{for all } u \in \Lambda,\ \pi \in \Pi$$

Proposition

If t is identity-like, then $t \Vdash \forall X\,(X \Rightarrow X)$

Proof: Exercise!

Remark: The converse implication also holds – Exercise!

- Examples of identity-like terms:
  - $\lambda x\,.\,x$, $(\lambda x\,.\,x)\,(\lambda x\,.\,x)$, etc.
  - $\lambda x\,.\,\text{æ}\,(\lambda k\,.\,x)$, $\lambda x\,.\,\text{æ}\,(\lambda k\,.\,k\,x)$, $\lambda x\,.\,\text{æ}\,(\lambda k\,.\,k\,x\,\omega)$, etc.
  - $\lambda x\,.\,\mathrm{quote}\,x\,(\lambda n\,.\,\mathrm{unquote}\,n\,(\lambda z\,.\,z))$
  - print $\overline{42}$, hace_mate

From computation to realizability (2/4)

Proof of: t identity-like iff $t \Vdash \forall X\,(X \Rightarrow X)$

($\Rightarrow$) Assume that t is identity-like, i.e.: $t \star u \cdot \pi \ \succ\ u \star \pi$ for all $t, u \in \Lambda$, $\pi \in \Pi$. Given a pole $\perp\!\!\!\perp$, we want to prove that $t \in |\forall X\,(X \Rightarrow X)|$ (w.r.t. the pole $\perp\!\!\!\perp$). For that, it suffices to prove that $t \star \pi \in \perp\!\!\!\perp$ for all $\pi \in \|\forall X\,(X \Rightarrow X)\|$.

Take an arbitrary $\pi \in \|\forall X\,(X \Rightarrow X)\|$. Since $\|\forall X\,(X \Rightarrow X)\| = \bigcup_{S \subseteq \Pi} \|\dot{S} \Rightarrow \dot{S}\|$,

we have $\pi \in \|\dot{S} \Rightarrow \dot{S}\|$ for some $S \subseteq \Pi$. And since $\|\dot{S} \Rightarrow \dot{S}\| = |\dot{S}| \cdot \|\dot{S}\|$, we also have $\pi \equiv u \cdot \pi'$ for some $u \in S^{\perp\!\!\!\perp}$ $(= |\dot{S}|)$ and $\pi' \in S$ $(= \|\dot{S}\|)$.

Now observe that $t \star \pi \ \equiv\ t \star u \cdot \pi' \ \succ\ u \star \pi' \ \in\ \perp\!\!\!\perp$ (since t is identity-like, and since $u \in S^{\perp\!\!\!\perp}$ and $\pi' \in S$), so that by anti-evaluation, we get $t \star \pi \in \perp\!\!\!\perp$ as desired.

($\Leftarrow$) Assume that $t \Vdash \forall X\,(X \Rightarrow X)$. Given $u \in \Lambda$ and $\pi \in \Pi$, we want to show that $t \star u \cdot \pi \ \succ\ u \star \pi$. For that, consider the pole $\perp\!\!\!\perp := \{p \in \Lambda \star \Pi \ :\ p \succ u \star \pi\}$ (closed under anti-evaluation) and the falsity value $S := \{\pi\}$ (with only one stack). Now observe that $u \star \pi \in \perp\!\!\!\perp$, hence $u \in S^{\perp\!\!\!\perp}$. Therefore we get

$$u \cdot \pi \ \in\ S^{\perp\!\!\!\perp} \cdot S \ =\ \|\dot{S} \Rightarrow \dot{S}\| \ \subseteq\ \|\forall X\,(X \Rightarrow X)\|,$$

from which we deduce that $t \star u \cdot \pi \in \perp\!\!\!\perp$ (since $t \in |\forall X\,(X \Rightarrow X)|$). From the definition of the pole $\perp\!\!\!\perp$, we conclude that $t \star u \cdot \pi \succ u \star \pi$. $\square$

From computation to realizability (3/4)

Example 2: Control operators:

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

- "Typing" k_π: $\quad k_\pi \star t \cdot \pi' \succ t \star \pi$

Lemma

If $\pi \in \|A\|$, then $\mathrm{k}_\pi \Vdash A \Rightarrow B$ (B any)

Proof: Exercise

- "Typing" cc: $\quad \text{cc} \star t \cdot \pi \succ t \star k_\pi \cdot \pi$

Proposition (Realizing Peirce's law)

$\text{cc} \Vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$

Proof: Exercise

From computation to realizability (4/4)

Proof of: $\pi \in \|A\|$ implies $\mathrm{k}_\pi \in |A \Rightarrow B|$ (w.rt. a fixed pole $\perp\!\!\!\perp$)

Assume that $\pi \in \|A\|$. We want to prove that $\mathrm{k}_\pi \in |A \Rightarrow B|$.
For that, it suffices to prove that $\mathrm{k}_\pi \star \pi' \in \perp\!\!\!\perp$ for all $\pi' \in \|A \Rightarrow B\|$.

Take an arbitrary $\pi' \in \|A \Rightarrow B\|$. Since $\|A \Rightarrow B\| = |A| \cdot \|B\|$, we have $\pi' \equiv u \cdot \pi''$ for some $u \in |A|$ and $\pi'' \in \|B\|$.

Now observe that $\mathrm{k}_\pi \star \pi' \equiv \mathrm{k}_\pi \star u \cdot \pi'' \succ u \star \pi \in \perp\!\!\!\perp$ (since $u \in |A|$ and $\pi \in \|A\|$), so that by anti-evaluation, we get $\mathrm{k}_\pi \star \pi' \in \perp\!\!\!\perp$ as desired. □

Proof of: $\text{œ} \Vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A$

Given a fixed pole $\perp\!\!\!\perp$, we want to prove that $\text{œ} \in |((A \Rightarrow B) \Rightarrow A) \Rightarrow A|$.
For that, it suffices to prove that $\text{œ} \star \pi \in \perp\!\!\!\perp$ for all $\pi \in \|((A \Rightarrow B) \Rightarrow A) \Rightarrow A\|$.

Take an arbitrary $\pi \in \|((A \Rightarrow B) \Rightarrow A) \Rightarrow A\|$. Since $\|((A \Rightarrow B) \Rightarrow A) \Rightarrow A\| = |(A \Rightarrow B) \Rightarrow A| \cdot \|A\|$, we have $\pi \equiv t \cdot \pi'$ for some $t \in |(A \Rightarrow B) \Rightarrow A|$ and $\pi' \in \|A\|$.

Now observe that $\text{œ} \star \pi \equiv \text{œ} \star t \cdot \pi' \succ t \star \mathrm{k}_{\pi'} \cdot \pi'$. Therefore it remains to prove that $t \star \mathrm{k}_{\pi'} \cdot \pi' \in \perp\!\!\!\perp$ (using the closure by anti-evaluation). For that, we observe that $\mathrm{k}_{\pi'} \in |A \Rightarrow B|$ (using the previous proposition) and $\pi' \in \|A\|$, hence

$$\mathrm{k}_{\pi'} \cdot \pi \in |A \Rightarrow B| \cdot \|A\| = \|(A \Rightarrow B) \Rightarrow A\|$$

But since $t \in |(A \Rightarrow B) \Rightarrow A|$, we conclude that $t \star \mathrm{k}_{\pi'} \cdot \pi' \in \perp\!\!\!\perp$ as desired. □

Anatomy of the model (1/3)

- **Denotation of universal quantification:**

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

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

(and similarly for 2nd-order universal quantification)

- **Denotation of implication:**

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

Truth value: $$|A \Rightarrow B| \;\subseteq\; |A| \rightarrow |B| \qquad \text{(by orthogonality)}$$

writing $$|A| \rightarrow |B| \;=\; \{t \in \Lambda \;:\; \forall u \in |A| \;\; tu \in |B|\} \qquad \text{(Kleene arrow)}$$

- **Note:** In general, we have $|A| \rightarrow |B| \not\subseteq |A \Rightarrow B|$. Nevertheless:

$$t \in |A| \rightarrow |B| \quad \text{implies} \quad \lambda x\,.\,t\,x \;\in\; |A \Rightarrow B| \qquad \text{(Exercise)}$$

Anatomy of the model (2/3)

Degenerate case: $\perp\!\!\!\perp = \varnothing$

- Classical realizability mimics the Tarski interpretation:

Degenerated interpretation

In the case where $\perp\!\!\!\perp = 0$, 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: $\perp\!\!\!\perp \neq \varnothing$

- Every truth value $|A|$ is inhabited:

Existence of paraproofs

If $\perp\!\!\!\perp \neq \varnothing$, then there is a term $✠_{\perp\!\!\!\perp} \in \Lambda$ (a "paraproof")
such that: $✠_{\perp\!\!\!\perp} \Vdash A$ for all closed formulas A

Proof. Since $\perp\!\!\!\perp \neq \varnothing$, pick a process $t_0 \star \pi_0 \in \perp\!\!\!\perp$ and write $✠_{\perp\!\!\!\perp} :\equiv k_{\pi_0}\, t_0$.
For all stacks π, we have: $✠_{\perp\!\!\!\perp} \star \pi \equiv \mathrm{k}_{\pi_0}\, t_0 \star \pi \succ \mathrm{k}_{\pi_0} \star t_0 \cdot \pi \succ t_0 \star \pi_0 \in \perp\!\!\!\perp$.
This immediately implies that $✠_{\perp\!\!\!\perp} \Vdash A$ for all closed formulas A. $\square$

Anatomy of the model (3/3)

The big dilemma:

$$\begin{cases} \text{When } \perp\!\!\!\perp = \varnothing\text{:} & \text{classical realizability is useless (?)} \\ & \quad \text{(since it mimics Tarski semantics)} \\ \text{When } \perp\!\!\!\perp \neq \varnothing\text{:} & \text{classical realizability is inconsistent (?)} \\ & \quad \text{(since } \maltese_{\perp\!\!\!\perp} \Vdash A \text{ for all closed formulas } A\text{)} \end{cases}$$

Solution: Only consider proof-like terms ($\in$ PL) as "valid" realizers

Recall: Proof-like term ($\in$ PL) = term without continuation constants (k_π)

Definition (Realized formulas)

In a given realizability model, a closed formula A with parameters is realized (notation: $\Vdash A$) when A is realized by at least a proof-like term:

$$\begin{array}{lcl} \Vdash A \ \text{(“}A\text{ is realized”)} & :\equiv & t \Vdash A \ \text{ for some } t \in \mathrm{PL} \\ & \Leftrightarrow & |A| \cap \mathrm{PL} \neq \varnothing \end{array}$$

Plan

Adequacy (1/2)

Aim: Prove the theorem of adequacy:

$t : A$ (in the sense of λNK2) implies $t \Vdash A$ (in the sense of realizability)

… and since t is proof-like (from λNK2), we deduce that A is realized (in each pole $\perp\!\!\!\perp$)

- Closing typing judgments $z_1 : A_1, \ldots, z_n : A_n \vdash t : A$
  - We close logical objects (1st-order terms, formulas, predicates) using semantic objects (natural numbers, falsity values, falsity functions)
  - We close proof-terms using realizers

Definition (Valuation)

1. A valuation is a function ρ such that
   - $\rho(x) \in \mathbb{N}$ for each 1st-order variable x
   - $\rho(X) : \mathbb{N}^k \to \mathfrak{P}(\Pi)$ for each 2nd-order variable X of arity k
2. The closure of A with ρ is written $A[\rho]$ (formula with parameters)

Adequacy (2/2)

Definition (Adequate judgment, adequate rule)

Given a fixed pole $\perp\!\!\!\perp$:

1. A judgment $z_1 : A_1, \ldots, z_n : A_n \vdash t : A$ is adequate if for every valuation ρ and for all $u_1 \Vdash A_1[\rho]$, ..., $u_n \Vdash A_n[\rho]$ we have:

$$t[z_1 := u_1, \ldots, z_n := u_n] \Vdash A[\rho]$$

2. A typing rule is adequate if it preserves the property of adequacy (from the premises to the conclusion of the rule)

Theorem

1. All typing rules of λNK2 are adequate
2. All derivable judgments of λNK2 are adequate

Proof: Exercise!

Corollary: If $\vdash t : A$ (A closed formula), then $t \Vdash\!\!\vdash A$ (with $t \in$ PL)

Extending adequacy to subtyping

Definition (Adequate subtyping judgment)

Judgment $A \leq B$ adequate $:\equiv$ $\|B[\rho]\| \subseteq \|A[\rho]\|$ (for all valuations ρ)

Remark: Implies that $|A[\rho]| \subseteq |B[\rho]|$ (for all ρ), but strictly stronger

- Some 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:** $\underbrace{\forall X\, \forall Y\, (((X \Rightarrow Y) \Rightarrow X) \Rightarrow X)}_{\text{Peirce's law}} \;\leq\; \underbrace{\forall X\, (\neg\neg X \Rightarrow X)}_{\text{DNE}}$

(derivable from the above rules)

Realizing equalities

- **Recall:** Equality between individuals is 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 $\perp\!\!\!\perp$)

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

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

Proof: Exercise!

- **Intuitions:**
  - A realizer of a true equality (in the model) behaves as the identity function $\lambda z\,.\,z$
  - A realizer of a false equality (in the model) behaves as a point of backtrack (breakpoint)

Realizing axioms

Corollary 1 (Realizing true equations)

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

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

Corollary 2

All defining equations of primitive recursive function symbols ($+$, $-$, $\times$, $/$, mod, $\uparrow$, etc.) are universally realized by $\mathbf{I} \equiv \lambda z\,.\,z$

Corollary 3 (Realizing Peano axioms 3 and 4)

$$\begin{array}{rcl} \mathbf{I} & \Vdash & \forall x\,\forall y\,(s(x) = s(y) \Rightarrow x = y) \\ \lambda z\,.\,z\,\mathbf{I} & \Vdash & \forall x\,\neg(s(x) = 0) \end{array}$$

Theorem: If $\mathrm{PA2}^- \vdash A$, then $\theta \Vdash A$ for some $\theta \in \mathrm{PL}$

Realizing true Horn formulas

Definition (Horn formulas)

1. A (positive/negative) literal is a formula L of the form
$$L \equiv e_1 = e_2 \qquad \text{or} \qquad L \equiv e_1 \neq e_2$$
2. A (positive/negative) Horn formula is a closed formula H of the form
$$H \equiv \forall\vec{x}\,[L_1 \Rightarrow \cdots \Rightarrow L_p \Rightarrow L_{p+1}] \qquad (p \geq 0)$$
where $L_1, \ldots, L_p$ are positive; L_{p+1} positive or negative

Theorem (Realizing true Horn formulas) [M. 2014]

If $\mathscr{M} \models H$, then:
$$\mathbf{I} \equiv \lambda z\,.\,z \Vdash H \qquad \text{(if } H \text{ positive)}$$
$$\lambda z_1 \cdots z_{p+1}\,.\,z_1\,(\cdots(z_{p+1}\,\mathbf{I})\cdots) \Vdash H \qquad \text{(if } H \text{ negative)}$$

- All axioms of $\mathrm{PA2}^- := \mathrm{PA2} - \mathrm{Ind}$ are Horn formulas
- Quantifications not relativized to $\mathbb{N}$ $\rightsquigarrow$ H holds for all individuals

Provability, universal realizability and truth

- From what precedes:
  1. A provable $\Rightarrow$ A universally realized (by a proof-like term)
  2. A universally realized $\Rightarrow$ A true (in the full standard model)

  $\rightsquigarrow$ Provability $\subsetneq$ Universal realizability $\subsetneq$ Truth

- **Beware!**

$$\begin{array}{ccc} \text{Intuitionistic proofs of } A & \subseteq & \text{Classical proofs of } A \\ \cap & & \cap \\ \text{Intuitionistic realizers of } A & \begin{array}{c}\not\subseteq \\ \not\supseteq\end{array} & \text{Classical realizers of } A \end{array}$$

- **Counter-example:**

$$\begin{array}{rlll} & \lambda z\,.\,z & \Vdash_{\text{Kleene}} & \forall x\,\forall y\,(s(x)=s(y)\Rightarrow x=y) \\ & \lambda z\,.\,\mathrm{refl} & \Vdash_{\text{Kleene}} & \forall x\,\forall y\,(s(x)=s(y)\Rightarrow x=y) \\ & \lambda z\,.\,z & \Vdash_{\text{Krivine}} & \forall x\,\forall y\,(s(x)=s(y)\Rightarrow x=y) \\ \text{but:} & \lambda z\,.\,\mathrm{refl} & \nVdash_{\text{Krivine}} & \forall x\,\forall y\,(s(x)=s(y)\Rightarrow x=y) \end{array}$$

(where refl $\equiv 0$ (Kleene) or refl $\equiv \mathbf{I}$ (Krivine) uniformly realizes true equalities)

Program extraction

Extracting a program from a proof in PA2

If $\mathrm{PA2} \vdash A$, then there is $\theta \in \mathrm{PL}$ such that $\theta \Vdash A^{\mathbb{N}}$
($A^{\mathbb{N}}$ obtained from A by relativizing all 1st-order quantifications to $\mathbb{N}$)

- **In practice:**
  - Only apply the adequacy theorem to the computationally relevant parts of the proof
  - For the computationally irrelevant parts (i.e. Horn formulas), use 'default realizers' $\rightsquigarrow$ realizer optimization
- **Example 1:** $\lambda\_,\_ \,.\, \mathbf{I} \;\Vdash\; (\forall x, y \in \mathbb{N})\,(x + y = y + x)$
- **Example 2:** Fermat's last theorem[1]

$$(\forall x, y, z, n \in \mathbb{N})\,(x \geq 1 \Rightarrow y \geq 1 \Rightarrow n \geq 3 \Rightarrow x^n + y^n \neq z^n)$$

1. realized by: $\lambda\_,\_,\_,\_, u_1, u_2, u_3, v \,.\, u_1\,(u_2\,(u_3\,(v\,\mathbf{I})))$

Plan

Some problems of classical realizability

1. **The specification problem**

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

   Specifying Peirce's law [Guillermo-M. 2014]

2. **Witness extraction from classical realizers** (cf next slides)

3. **Realizability algebras + Cohen forcing**

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

4. **Models induced by classical realizability**

   What are the interesting formulas that 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 [K. 2012]

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 is not 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.)

- Two possible compromises:
  - **Intuitionistic logic:** Restrict the shape of the realizer t_0 (by only keeping intuitionistic reasoning principles)
  - **Classical logic:** Restrict the shape of the formula $A(x)$ (typically: Δ^0_0-formulas)

Storage operators (1/3)

- The call-by-value implication:

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

with the semantics: $\qquad \|\{e\} \Rightarrow A\| \;:=\; \{\bar{n} \cdot \pi \;:\; n = e^{\mathbb{N}},\ \pi \in \|A\|\}$

[**Recall:** $\|e \in \mathbb{N} \Rightarrow A\| \;:=\; \{u \cdot \pi \;:\; u \in |e \in \mathbb{N}|,\ \pi \in \|A\|\}$]

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

so that: $\qquad \mathbf{I} \;\Vdash\; \forall x \forall Z\,[(x \in \mathbb{N} \Rightarrow Z) \Rightarrow (\{x\} \Rightarrow Z)]$ $\qquad$ (direct implication)

Definition (Storage operator) [Krivine]

A storage operator is a closed proof-like term M such that:

$$M \;\Vdash\; \forall x \forall Z\,[(\{x\} \Rightarrow Z) \Rightarrow (x \in \mathbb{N} \Rightarrow Z)] \qquad \text{(converse implication)}$$

Theorem (Existence)

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

Proof: Postponed.

Storage operators (2/3)

- Intuitively, a storage operator

$$M \Vdash \forall x \forall Z\,[(\{x\} \Rightarrow Z) \Rightarrow (x \in \mathbb{N} \Rightarrow Z)]$$

is a proof-like term that is intended to be applied to

  - a function f that only accepts values (i.e. intuitionistic integers)
  - a classical integer $t \Vdash n \in \mathbb{N}$ (that may contain continuations k_π)

and that evaluates (or 'smoothes') the classical integer t into a value of the form $\bar{n}$ before passing this value to f

- By subtyping, we also have:

$$M \Vdash \forall Z\,[\forall x\,(\{x\} \Rightarrow Z(x)) \;\Rightarrow\; (\forall x \in \mathbb{N})\,Z(x)]$$

This means that if a property $Z(x)$ holds for all intuitionistic integers, then it holds for all classical integers too

- **Conclusion:** $e \in \mathbb{N} \Rightarrow A$ and $\{e\} \Rightarrow A$ are equivalent

Storage operators (3/3)

Proof of existence of storage operators: Take $M \;:\equiv\; \lambda fn\,.\,n\,f\,(\lambda hx\,.\,h\,(\bar{s}\,x))\,\bar{0}$.

Given a pole $\perp\!\!\!\perp$, we want to prove that $M \Vdash \forall x\,\forall Z\,[(\{x\} \Rightarrow Z) \Rightarrow (x \in \mathbb{N} \Rightarrow Z)]$.
This amounts to prove that $M \Vdash (\{n\} \Rightarrow \dot{S}) \Rightarrow n \in \mathbb{N} \Rightarrow \dot{S}$ for all $n \in \mathbb{N}$ and $S \subseteq \Pi$.
For that, pick a stack in $\|(\{n\} \Rightarrow \dot{S}) \Rightarrow n \in \mathbb{N} \Rightarrow \dot{S}\|$, that is of the form $t \cdot u \cdot \pi$, where $t \in |\{n\} \Rightarrow \dot{S}|$, $u \in |n \in \mathbb{N}|$ and $\pi \in S$. We want to prove that $M \star t \cdot u \cdot \pi \in \perp\!\!\!\perp$.
Since $M \star t \cdot u \cdot \pi \;\succ\; u\,t\,(\lambda hx\,.\,h\,(\bar{s}\,x))\,\bar{0} \star \pi \;\succ\; u \star t \cdot (\lambda hx\,.\,h\,(\bar{s}\,x)) \cdot \bar{0} \cdot \pi$, it suffices to prove that $u \star t \cdot (\lambda hx\,.\,h\,(\bar{s}\,x)) \cdot \bar{0} \cdot \pi \;\in\; \perp\!\!\!\perp$ (by anti-evaluation).

Let us now consider the falsity function $F : \mathbb{N} \to \mathfrak{P}(\Pi)$ defined by:

$$F(p) \;:=\; \begin{cases} \|\{n-p\} \Rightarrow \dot{S}\| \;:=\; \{\overline{n-p}\} \cdot S & \text{if } p < n \\ \|\top\| \;:=\; \varnothing & \text{if } p \geq n \end{cases} \qquad \text{(for all } n \in \mathbb{N}\text{)}$$

We easily check that $\lambda hx\,.\,h\,(\bar{s}\,x) \;\in\; |\dot{F}(p) \Rightarrow \dot{F}(p+1)|$ for all $p \in \mathbb{N}$, (Exercise)
and therefore: $\lambda hx\,.\,h\,(\bar{s}\,x) \;\in\; |\forall x(\dot{F}(x) \Rightarrow \dot{F}(s(x)))|$.

Now observing that: $u \;\in\; |n \in \mathbb{N}| \;\subseteq\; |\dot{F}(0) \Rightarrow \forall x\,(\dot{F}(x) \Rightarrow \dot{F}(s(x))) \Rightarrow \dot{F}(n)|$

whereas: $t \;\in\; |\{n\} \Rightarrow \dot{S}| \;=\; |\dot{F}(0)|$

$\lambda hx\,.\,h\,(\bar{s}\,x) \;\in\; |\forall x(\dot{F}(x) \Rightarrow \dot{F}(s(x)))|$

and $\bar{0} \cdot \pi \;\in\; \|\{\bar{0}\} \Rightarrow \dot{S}\| \;=\; \|\dot{F}(n)\|$

we deduce that $u \star t \cdot (\lambda hx\,.\,h\,(\bar{s}\,x)) \cdot \bar{0} \cdot \pi \;\in\; \perp\!\!\!\perp$ as desired. $\square$

Computing with storage operators

- Given a k-ary function symbol f, we let:

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

Theorem (Specification of the formula Comput(f))

For all $t \in \Lambda$, the following assertions are equivalent:

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

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

Proof: Same technique as for: "t identity-like iff $t \Vdash \forall X \, (X \Rightarrow X)$" (Exercise!)

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

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

The naive extraction method

- 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 there are $n \in \mathbb{N}$ and $u \in \Lambda$ such that:

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

(where $u \Vdash A(n)$ w.r.t. the particular pole needed to prove the property)

Proof. Take $\perp\!\!\!\perp_\pi := \{p \in \Lambda \star \Pi \;:\; p \succ \mathsf{stop} \star \bar{n} \cdot u \cdot \pi$ for some $n \in \mathbb{N}$ and $u \in \Lambda\}$ and prove that $M(\lambda xy\,.\,\mathsf{stop}\,x\,y) \cdot \pi \;\in\; \|(\exists x \in \mathbb{N}) A(x)\|$ (w.r.t. $\perp\!\!\!\perp_\pi$). □

- But $n \in \mathbb{N}$ might be a false witness because the justification $u \Vdash A(n)$ is cheating! (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

  (This can be proved using Kleene realizability adapted to $\mathrm{PA2}^-$)

Extraction in the Σ^0_1-case

Extraction in the Σ^0_1-case (+ display intermediate results)

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

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

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

Proof. Take $\perp\!\!\!\perp_\pi \ := \ \{p \in \Lambda \star \Pi \ : \ p \succ \mathrm{stop} \star \bar{n} \cdot \pi \ \text{ for some } n \in \mathbb{N} \text{ s.t. } f(x) = 0\}$ and prove that $\quad M(\lambda xy\,.\,\mathrm{print}\,x\,y\,(\mathrm{stop}\,x)) \cdot \pi \ \in \ \|(\exists x \in \mathbb{N})(f(x) = 0)\|$. □

- Storage operator M used to evaluate 1st component (x)
- 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 has to implement the storage operator M accordingly)

Example: the minimum principle

- Given a unary function symbol f, write:

$$\begin{array}{rcll} \mathrm{Total}(f) & := & (\forall x \in \mathbb{N})(f(x) \in \mathbb{N}) & \text{(totality predicate)} \\ x \leq y & := & x - y = 0 & \text{(truncated subtraction)} \end{array}$$

Theorem (Minimum principle – MinP)

$$\mathrm{PA2}^- \ \vdash \ \mathrm{Total}(f) \ \Rightarrow \ (\exists x \in \mathbb{N}) \underbrace{(\forall y \in \mathbb{N})\,(f(x) \leq f(y))}_{\text{undecidable}}$$

Proof. Reductio ad absurdum + course by value induction

- The minimum principle is not intuitionistically provable (oracle)
- We cannot apply the Σ^0_1-extraction technique to the above proof (applied to a totality proof of f), since the conclusion is Σ^0_2

  The body $(\forall y \in \mathbb{N})\,(f(x) \leq f(y))$ of $\exists$-quantification is undecidable

Using the minimum principle to prove a Σ_1^0-formula

- **Idea:** The value x given by the minimum principle can be used to prove a Σ_1^0-formula, so that we can perform program extraction:

Corollary

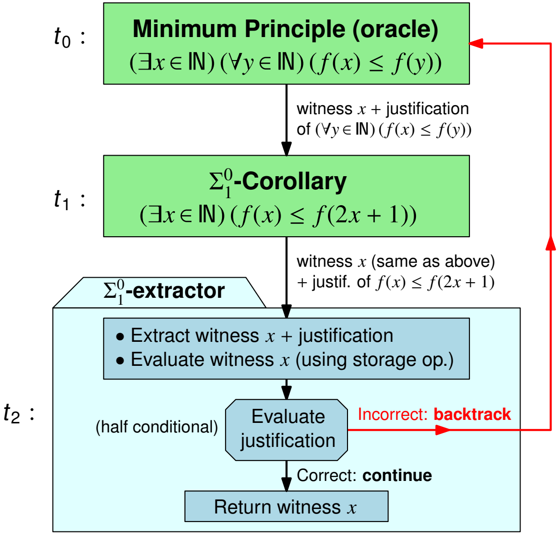
$$\mathrm{PA2}^- \ \vdash \ \mathrm{Total}(f) \ \Rightarrow \ (\exists x \in \mathbb{N})\ \underbrace{(f(x) \leq f(2x+1))}_{\text{decidable}}$$

More generally: $\mathrm{PA2}^- \ \vdash \ \mathrm{Total}(f) \wedge \mathrm{Total}(g) \ \Rightarrow \ (\exists x \in \mathbb{N})\,(f(x) \leq f(g(x)))$

Proof. Take the point x given by the minimum principle

- Applying Σ_1^0-extraction to the above non-constructive proof, we get a correct witness after finitely many evaluation steps
- How is this witness computed?

The algorithm underlying Σ^0_1-extraction



Transcript of the extraction process

Take $f(x) = |x - 1000|$ (real minimum at $x = 1000$)

and apply Σ^0_1-extraction to the proof of $(\exists x \in \mathbb{N})\,(f(x) \le f(2x+1))$

Step 1	Oracle says:	take $x = 0$	since $(\forall y \in \mathbb{N})\,(f(0) \le f(y))$	(false)
	Corollary says:	take $x = 0$	since $f(0) \le f(1)$	(false)
	Σ^0_1-extractor evaluates incorrect justification and backtracks			
Step 2	Oracle says:	take $x = 1$	since $(\forall y \in \mathbb{N})\,(f(1) \le f(y))$	(false)
	Corollary says:	take $x = 1$	since $f(1) \le f(3)$	(false)
	Σ^0_1-extractor evaluates incorrect justification and backtracks			
Step 3	Oracle says:	take $x = 3$	since $(\forall y \in \mathbb{N})\,(f(3) \le f(y))$	(false)
	Corollary says:	take $x = 3$	since $f(3) \le f(7)$	(false)
	Σ^0_1-extractor evaluates incorrect justification and backtracks			
Step 4	Oracle says:	take $x = 7$	since $(\forall y \in \mathbb{N})\,(f(7) \le f(y))$	(false)
		. . . .		
Step 11	Oracle says:	take $x = 1023$	since $(\forall y \in \mathbb{N})\,(f(1023) \le f(y))$	(false)
	Corollary says:	take $x = 1023$	since $f(1023) \le f(2047)$	(true)
	Σ^0_1-extractor evaluates correct justification and returns $x = 1023$			

Note that answer $x = 1023$ is correct... but not the point where f reaches its minimum

Extraction in the Σ_n^0/Π_n^0-case (1/2)

Definition (Conditional refutation) [M. 2010]

$r_A \in \Lambda$ is a conditional refutation of the predicate $A(x)$ if

$$\text{For all } n \in \mathbb{N} \text{ such that } \mathscr{M} \not\models A(n): \quad r_A\,\overline{n} \ \Vdash\ \neg A(n)$$

- Such a conditional refutation can be constructed for every predicate $A(x)$ of 1st-order arithmetic

  This result is a consequence of the following

Theorem (Realizing true arithmetic formulas) [Krivine, Miquey]

For every formula $A(x_1, \ldots, x_k)$ of 1st-order arithmetic, there exists a closed proof-like term t_A such that:

$$\text{If} \quad \mathscr{M} \models A(n_1, \ldots, n_k), \quad \text{then} \quad t_A\,\bar{n}_1 \cdots \bar{n}_k \ \Vdash\ A(n_1, \ldots, n_k)$$

(for all $n_1, \ldots, n_k \in \mathbb{N}$)

Extraction in the Σ^0_n/Π^0_n-case (2/2)

The Kamikaze extraction method [M. 2010]

Let

1. $t_0 \Vdash (\exists x \in \mathbb{N})\, A(x)$
2. r_A a conditional refutation of the predicate $A(x)$

Then the process

$$t_0 \star M\,(\lambda xy\,.\,\mathrm{print}\,x\,(r_A\,x\,y)) \cdot \pi$$

displays a correct witness after finitely many evaluation steps

Proof. Take $\perp\!\!\!\perp := \{p \in \Lambda \star \Pi \;:\; p \succ \mathrm{stop} \star \bar{n} \cdots$ for some $n \in \mathbb{N}$ s.t. $\mathscr{M} \models A(n)\}$ and prove that $M\,(\lambda xy\,.\,\mathrm{print}\,x\,(r_A\,x\,y)) \cdot \pi \;\in\; \|(\exists x \in \mathbb{N})(f(x) = 0)\|$. □

- **Remark:** No correctness invariant is ensured as soon as the (first) correct witness has been displayed!

  After, anything may happen: displaying incorrect witnesses, infinite loop, crash, etc. (Kamikaze behavior)

Interlude: on numeration systems

- Numeration systems used in the History:

Tally sticks	(35000 BC)	𝍸𝍸𝍸𝍸𝍸𝍸𝍸𝍸 ||
Babylonian	(3100 BC)	𒌋𒌋𒌋𒌋𒐖
Egyptian	(3000 BC)	𓎆𓎆𓎆𓎆𓏻
Roman	(1000 BC)	XLII
Hindu-Arabic	(300 AD)	42

- Numeration systems used in Logic:

Peano: $ssssssssssssssssssssssssssssssssssssssssss0$

Church:
$$\lambda xf\,.\,f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f(f\,x)))$$

Krivine:
$$\begin{array}{l}
(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))(\\
(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))(\\
(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))(\\
(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))(\\
(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))(\\
(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))(\\
(\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))((\lambda nxf.f(nxf))(\\
(\lambda xf.x))
\end{array}$$

Primitive numerals (1/2)

To get rid of Krivine numerals $\bar{n} \;=\; \bar{s}^n\bar{0}$ (cf paleolithic numeration)
we extend the machine with the following instructions: [M. 2010]

- For each number $n \in \mathbb{N}$, an instruction $\widehat{n} \in \mathcal{K}$ (primitive numeral)
  with no evaluation rule (i.e. inert constant: pure data)

  Intuition: $\widehat{n} \star \pi \;\succ\;$ `segmentation fault`

- An instruction null $\in \mathcal{K}$ with the rules

$$\mathrm{null} \star \widehat{n} \cdot u \cdot v \cdot \pi \quad \succ \quad \begin{cases} u \star \pi & \text{if } n = 0 \\ v \star \pi & \text{otherwise} \end{cases}$$

- Instructions $\check{f} \in \mathcal{K}$ with the rules

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

  for all the usual arithmetic operations f

Primitive numerals (2/2)

- Call-by-value implication, yet another definition:

Formulas $\quad A, B \quad ::= \quad \cdots \quad | \quad [e] \Rightarrow A$

with the semantics: $\quad \|[e] \Rightarrow A\| \;=\; \{\widehat{n} \cdot \pi \;:\; n = e^{\mathbb{N}},\; \pi \in \|A\|\}$

- Redefining the set of natural numbers:

$$\mathbb{N}' \;:=\; \{x \;:\; \forall Z\,(([x] \Rightarrow Z) \Rightarrow Z)\}$$

$$\begin{array}{lcl}
\mathsf{box} \;:=\; \lambda xk\,.\,k\,x & \Vdash & \forall x\,([x] \Rightarrow x \in \mathbb{N}') \\
\mathsf{box}\,\widehat{n} & \Vdash & n \in \mathbb{N}' \\
\lambda n\,.\,n\,\lambda x\,.\,\check{s}\,x\,\mathsf{box} & \Vdash & (\forall x \in \mathbb{N}')(s(x) \in \mathbb{N}') \\
\lambda nm\,.\,n\,\lambda x\,.\,m\,\lambda y\,.\,(\check{+})\,x\,y\,\mathsf{box} & \Vdash & (\forall x, y \in \mathbb{N}')(x + y \in \mathbb{N}')
\end{array}$$

$$\begin{array}{rl}
\mathsf{rec_cbv} \;:= & \lambda z_0 z_s\,.\,\mathbf{Y}\,\lambda rx\,.\,\mathsf{null}\,x\,z_0\,((\check{-})\,x\,\hat{1}\,\lambda y\,.\,z_s\,y\,(r\,y)) \\
\Vdash & \forall Z\,[Z(0) \;\Rightarrow\; \forall y\,([y] \Rightarrow Z(y) \Rightarrow Z(s(y))) \;\Rightarrow\; \forall x\,([x] \Rightarrow Z(x))] \\
\mathsf{rec} \;:= & \lambda z_0 z_s n\,.\,n\,\lambda x\,.\,\mathsf{rec_cbv}\,z_0\,(\lambda yz\,.\,z_s\,(\mathsf{box}\,y)\,z)\,x \\
\Vdash & \forall Z\,[Z(0) \Rightarrow (\forall y \in \mathbb{N}')(Z(y) \Rightarrow Z(s(y))) \;\Rightarrow\; (\forall x \in \mathbb{N}')Z(x)]
\end{array}$$

- **Conclusion:** $\quad \Vdash\; \forall x\,(x \in \mathbb{N}' \;\Leftrightarrow\; x \in \mathbb{N})$

Krivine's realizability vs the LRS-translation (1/2)

- Krivine's realizability can be seen as the composition of the Lafont-Reus-Streicher (LRS) translation with Kleene realizability:

$$\text{CPS} \circ \text{Krivine} \quad = \quad \text{Kleene} \circ \text{LRS}$$

[Oliva-Streicher 2008]

The dictionary

Classical realizability (Krivine)	**Lafont-Reus-Streicher translation**
Pole $\perp\!\!\!\perp$	Return formula R
Falsity value $\|A\|$	Negative translation $A^{\perp}$
$\|A \Rightarrow B\| \;:=\; \lvert A\rvert \cdot \|B\|$	$(A \Rightarrow B)^{\perp} \;:=\; A^{LRS} \wedge B^{\perp}$
Truth value $\lvert A\rvert \;:=\; \|A\|^{\perp\!\!\!\perp}$	$A^{LRS} \;:=\; A^{\perp} \Rightarrow R$

- Through the CPS-translation, Krivine's extraction method in the Σ^0_1-case is exactly Friedman's trick (transposed to LRS) [M. 2010]

Krivine's realizability vs the LRS-translation (2/2)

Beware of reductionism!

- The decomposition holds only for *pure* classical reasoning
  (extra instructions are not taken into account)
- Classical realizers are easier to understand than their CPS-translations (and more efficient)
- Classical realizability is more than Kleene's realizability composed with the Lafont-Reus-Streicher translation

An image:

$$2H_2 + O_2 \quad \longrightarrow \quad 2H_2O$$

but can we deduce the properties of *water* from the ones of H_2 and O_2?