Classical realizability and forcing
Part 2: Classical realizability interpretation

Alexandre Miquel

UNIVERSIDAD DE LA REPUBLICA URUGUAY

FACULTAD DE INGENIERIA

EQUIPO DE LÓGICA UDELAR

Logic Colloquium (LC'14)
Vienna Summer of Logic – July 18th, 2014 – Vienna

Plan

1. The λ_c-calculus
2. Realizability interpretation
3. Adequacy
4. Realizability algebras

Plan

1. The λ_c-calculus
2. Realizability interpretation
3. Adequacy
4. Realizability algebras

Terms, stacks and processes

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

Terms, stacks and processes

Terms	t, u	$::=$	$x \mid \lambda x\,.\,t \mid tu \mid \kappa \mid \mathrm{k}_\pi$	$(\kappa \in \mathcal{K})$
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 æ
  - Continuation constants k_π, one for every stack π (generated by æ)
- **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 $\quad t, u \quad ::= \quad x \quad | \quad \lambda x \,.\, t \quad | \quad tu \quad | \quad \kappa \qquad (\kappa \in \mathcal{K})$

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

- Natural numbers encoded as proof-like terms by:

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

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

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

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

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

Krivine Abstract Machine (KAM)

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

(+ reflexivity & transitivity)

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

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

- Rules **Push** and **Grab** implement weak head β-reduction:

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	$\text{cc} \star u \cdot \pi$	$\succ$	$u \star \mathrm{k}_\pi \cdot \pi$
Restore	$\mathrm{k}_\pi \star u \cdot \pi'$	$\succ$	$u \star \pi$

  - Instruction cc most often used in the pattern

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

Example of extra instructions

- The instruction quote

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

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

  - Useful to realize the Axiom of Dependent Choices (DC) [Krivine 03]

- The instruction eq

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

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

- The instruction ⫛ (‘fork’)

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

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

Plan

1 The λ_c-calculus

2 Realizability interpretation

3 Adequacy

4 Realizability algebras

Classical realizability: principles

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

- Classical realizability model parameterized by a pole $\perp\!\!\!\perp$
  = set of processes closed under anti-evaluation

- Each formula A is interpreted as two sets:
  - A set of stacks $\|A\|$ (falsity value)
  - A set of terms $|A|$ (truth value)

- Falsity value $\|A\|$ defined by induction on A (negative interpretation)

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

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

Architecture of the realizability model

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

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

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

Interpreting closed formulas with parameters

Let A be a closed formula (with parameters)

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

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

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

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

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 (1/2)

Fundamental idea: The computational behavior of a term determines the formula(s) 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: converse implication holds – exercise!)

- Examples of identity-like terms:
  - $\lambda x\,.\,x$, $(\lambda x\,.\,x)\,(\lambda x\,.\,x)$, etc.
  - $\lambda x\,.\,\mathfrak{cc}\,(\lambda k\,.\,x)$, $\lambda x\,.\,\mathfrak{cc}\,(\lambda k\,.\,k\,x)$, $\lambda x\,.\,\mathfrak{cc}\,(\lambda k\,.\,k\,x\,\omega)$, etc.
  - $\lambda x\,.\,\mathrm{quote}\,x\,\lambda n\,.\,\mathrm{unquote}\,n\,(\lambda z\,.\,z)$

From computation to realizability (2/2)

Example 2: Control operators:

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

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

Lemma

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

Proof: Exercise

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

Proposition (Realizing Peirce's law)

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

Proof: Exercise

Anatomy of the model (1/2)

- **Denotation of universal quantification:**

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

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

(and similarly for 2nd-order universal quantification)

- **Denotation of implication:**

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

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

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

Anatomy of the model (2/2)

- **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:

    If $t_0 \star \pi_0 \in \perp\!\!\!\perp$, then $\mathsf{k}_{\pi_0} t_0 \in |A|$ for all A (paraproof)
  - We shall only consider realizers that are proof-like terms ($\in$ PL)

Plan

1 The λ_c-calculus

2 Realizability interpretation

3 Adequacy

4 Realizability algebras

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)

- Closing typing judgments $x_1 : A_1, \ldots, x_n : A_n \vdash t : A$
  - 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 (Valuations)

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. Closure of A with ρ written $A[\rho]$ (formula with parameters)

Adequacy (2/2)

Definition (Adequate judgment, adequate rule)

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

1. A judgment $x_1 : A_1, \ldots, x_n : A_n \vdash t : A$ is adequate if for every valuation ρ and for all $u_1 \Vdash A_1[\rho]$, $\ldots$, $u_n \Vdash A_n[\rho]$ we have:

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

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

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

Realizing equalities

- Equality between individuals defined by

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

Denotation of Leibniz equality

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

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

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

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

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

Corollary 2

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

Corollary 3 (Realizing Peano axioms 3 and 4)

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

Theorem: If SOL $\vdash A$, then $\theta \Vdash A$ for some $\theta \in$ 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:

$$\mathbf{I} \equiv \lambda z\,.\,z \Vdash H \qquad \text{(if } H \text{ positive)}$$

$$\lambda z_1 \cdots z_{p+1}\,.\,z_1\,(\cdots(z_{p+1}\,\mathbf{I})\cdots) \Vdash H \qquad \text{(if } H \text{ negative)}$$

- Peano axioms 3 and 4 are particular cases of Horn formulas
- Quantifications not relativized to $\mathbb{N}$ $\rightsquigarrow$ H holds for all individuals

Realizing the axiom of dependent choices

Dependent choice, 'quote' and the clock [Krivine 03]

- The instruction quote

$$\text{quote} \star t \cdot u \cdot \pi \quad \succ \quad u \star \overline{\ulcorner t \urcorner} \cdot \pi \qquad\qquad (\ulcorner t \urcorner = \text{code of } t)$$

- Used to realize the Non Extensional Axiom of Choice:

$$\lambda x\,.\,\text{quote}\ x\ x \quad \Vdash$$
$$\forall X\,[(\forall n \in \mathbb{N}) A(X, \varepsilon_A(X, n)) \ \Rightarrow\ \forall Y\, A(X, Y)] \qquad \text{(NEAC)}$$

(with a suitable interpretation of 3rd-order symbol ε_A)

- In 2nd-order logic, NEAC does not imply full AC, but is sufficient to realize the axiom of dependent choices:

$$\forall X\, \exists Y\, R(X, Y) \ \Rightarrow$$
$$\forall X_0\, \exists U\, [U(0) = X_0 \ \wedge\ (\forall n \in \mathbb{N})\, R(U(n), U(s(n)))] \qquad \text{(DC)}$$

Plan

1. The λ_c-calculus

2. Realizability interpretation

3. Adequacy

4. Realizability algebras

Extensions

- Realizability model initially designed for classical 2nd-order logic, but this construction extends to:
  - Higher-order arithmetic
  - The Calculus of constructions with universes (Coq proof assistant)
  - Zermelo-Fraenkel set theory (ZF)
    - Need to work in an intensional presentation of ZF: ZF_ε
    - Intensional membership ε vs. extensional $\in/=$ [Friedman]
  - Each of these extensions supports Dependent Choices (DC)
- Based on Krivine's λ_c-calculus... (possibly enriched with extra instructions) but can be generalized to classical realizability algebras [Krivine 10]

Cohen forcing versus classical realizability

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

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

Combining Cohen forcing with classical realizability

- **Forcing in classical realizability** [Krivine 09]
  - Introduce realizability algebras, generalizing the λ_c-calculus
  - Discover the program transformation underlying forcing
  - Extend iterated forcing to classical realizability
  - Show how to force the existence of a well-ordering over $\mathbb{R}$ (while keeping evaluation deterministic)
- **Computational analysis of forcing** [Miquel 11]
  - Focus on the underlying program transformation (no generic filter)
  - Hard-wire the program transformation into the abstract machine

Underlying methodology

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

Realizability algebras

Definition

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

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

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

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

Examples (1/2)

- From an implementation of λ_c:

Standard realizability algebra

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

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

Examples (2/2)

- From a forcing poset $(C, \leq)$ defined as an upwards closed subset of a meet semi-lattice $(\mathcal{L}, \leq)$: $C \subseteq \mathcal{L}$, $C \neq \varnothing$ upwards closed

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

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

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

- Corresponding realizability model equivalent to the forcing model defined from the poset $(C, \leq)$