Realizability toposes from specifications

Jonas Frey

Warsaw, 3 July 2015

Overview

- In this talk
  - New Krivine realizability models using side effects (I/O)
  - Slogan: *'Specifications are poles'*
- Use categorical approach to realizability, via
  - indexed preorders (triposes)
  - [and categories of partial equivalence relations (toposes)]
- Structure of talk:
  - Warm up: categorical formulation of Kleene realizability
  - 'Classical' Krivine realizability formulated categorically
  - New poles using side effects
  - Open questions, speculations, future work

- BHK: informal **proof semantics** of constructive predicate logic

  Slogan: *"The meaning of a formula is the set of its proofs"*

- 'Proof' defined by induction on structure of formulas:

Proof of $P \wedge Q$:	Pair $\langle a, b \rangle$ of proof of P and proof of Q
Proof of $P \Rightarrow Q$:	Function f mapping proofs of P to proofs of Q
Proof of $\forall x : \mathbb{N} \,.\, \varphi(x)$:	Function f mapping integers n to proofs of $\varphi(n)$
Proof of $\bot$:	None

  ...

- Realizability: replace informal 'proofs' by representations as mathematical objects, called 'realizers'
  - integers
  - closed λ-terms
  - combinators
  - strategies from game semantics
  - ...

- BHK: informal **proof semantics** of constructive predicate logic

  Slogan: *"The meaning of a formula is the set of its proofs"*

- 'Proof' defined by induction on structure of formulas:

Proof of $P \wedge Q$:	Pair $\langle a, b \rangle$ of proof of P and proof of Q
Proof of $P \Rightarrow Q$:	**Function f mapping proofs of P to proofs of Q**
Proof of $\forall x : \mathbb{N} \,.\, \varphi(x)$:	Function f mapping integers n to proofs of $\varphi(n)$
Proof of $\bot$:	None
...	

- Realizability: replace informal 'proofs' by representations as mathematical objects, called 'realizers'
  - integers
  - closed λ-terms
  - combinators
  - strategies from game semantics
  - ...

- **Implication most important for categorical formulation**

Categorical approach – ordering on truth values

- Fix set Λ of realizers (say closed λ-terms modulo β-equivalence)
- **Truth value:** Set $P \subseteq \Lambda$ of realizers
- For $P, Q \subseteq \Lambda$, a **realizer** of $P \Rightarrow Q$ is a $t \in \Lambda$ such that

$$\forall u \in P \,.\, tu \in Q \qquad \text{[Notation: } t \Vdash P \Rightarrow Q\text{]}$$

- Define ordering on set $\mathcal{P}(\Lambda)$ of truth values by

$$\begin{aligned} P \leq Q \quad &:\Leftrightarrow \quad \exists t \in \Lambda \,.\, t \Vdash P \Rightarrow Q \\ &\Leftrightarrow \quad \exists t \in \Lambda \; \forall u \in P \,.\, tu \in Q \end{aligned}$$

- $(\mathcal{P}(\Lambda), \leq)$ is a preorder (with only two elements up to equivalence)

Categorical approach – ordering on semantic predicates

- **(Semantic) predicates** are used to interpret **open formulas**
- Predicates are **families of truth values**

$$\varphi, \psi : J \to \mathcal{P}(\Lambda) \qquad (J \textbf{ indexing set})$$

- **Ordering** on predicates

$$\varphi \leq \psi \qquad :\Leftrightarrow \qquad \exists t \in \Lambda \;\; \forall j \in J \quad . \quad t \Vdash \varphi(j) \Rightarrow \psi(j)$$

- $(\mathcal{P}(\Lambda)^J, \leq)$ is a **Heyting algebra**

Categorical approach – ordering on semantic predicates

- **(Semantic) predicates** are used to interpret **open formulas**
- Predicates are **families of truth values**

$$\varphi, \psi : J \to \mathcal{P}(\Lambda) \qquad (J \textbf{ indexing set})$$

- **Ordering** on predicates

$$\varphi \leq \psi \qquad :\Leftrightarrow \qquad \exists t \in \Lambda \;\; \forall j \in J \quad . \quad t \Vdash \varphi(j) \Rightarrow \psi(j)$$

- $(\mathcal{P}(\Lambda)^J, \leq)$ is a **Heyting algebra**
- Compare to **pointwise ordering**

$$\varphi \leq_{\text{ptw}} \psi \qquad :\Leftrightarrow \qquad \forall j \in J \;\; \exists t \in \Lambda \quad . \quad t \Vdash \varphi(j) \Rightarrow \psi(j)$$

Semantic predicates as indexed preorder

- Functions $f : K \to J$ induce monotone **reindexing maps**

$$f^* : (\mathcal{P}(\Lambda)^J, \leq) \to (\mathcal{P}(\Lambda)^K, \leq)$$

- The mappings

$$\begin{array}{lcl} J & \mapsto & (\mathcal{P}(\Lambda)^J, \leq) \\ f & \mapsto & f^* \end{array}$$

  constitute an **indexed preorder**

$$\mathcal{Q} : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Ord}$$

$\mathbb{Q} : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Ord}$ is a **tripos**, i.e.

- All preorders $(\mathcal{P}(\Lambda)^J, \leq)$ are **Heyting algebras**
- All reindexing maps $f^* : (\mathcal{P}(\Lambda)^J, \leq) \to (\mathcal{P}(\Lambda)^K, \leq)$ have adjoints

$$\exists_f \dashv f^* \dashv \forall_f$$

  satisfying **Frobenius** and **Beck-Chevalley conditions**
- There is a **generic predicate** $\mathrm{tr} \in \mathbb{Q}(\mathrm{Prop})$ satisfying

$$\forall J \in \mathbf{Set}\ \forall \varphi \in \mathbb{Q}(J)\ \exists f : J \to \mathrm{Prop}\,.\,\varphi \cong f^*(\mathrm{tr})$$

From the tripos $\mathbb{Q}$, we can construct a **realizability topos** $\mathbf{Set}[\mathbb{Q}]$

Reference:

- J. van Oosten. *Realizability: An Introduction to its Categorical Side.*

Krivine's classical realizability – basic ideas

Guiding intuitions:

- Combination of **Kleene realizability** and **negative translation**
- Syntax with **continuations** instead of **CPS translation**
- **Negation** replaced by $(-) \Rightarrow R$, for convenient R ('pole')
- different R give different **realizability models**
- new: encode **specifications** in R

Three syntactic classes

$$\begin{array}{lllll} \text{Terms:} & t & ::= & x \mid \lambda x.t \mid tt \mid \text{cc} \mid \text{k}_\pi & \\ \text{Stacks:} & \pi & ::= & \pi_0 \mid t{\cdot}\pi & (\ t \text{ closed}, \pi_0 \in \Pi_0\ ) \\ \text{Processes:} & p & ::= & t \star \pi & (\ t \text{ closed}\ ) \end{array}$$

where Π_0 is a fixed set of **stack constants**

- Λ set of closed terms
- Π set of stacks
- $\Lambda \star \Pi$ set of processes

Reduction relation on processes:

$$\begin{array}{rrcl} \text{(push)} & tu \star \pi & \succ & t \star u{\cdot}\pi \\ \text{(pop)} & (\lambda x\,.\,t[x]) \star u{\cdot}\pi & \succ & t[u] \star \pi \\ \text{(save)} & \text{cc} \star t{\cdot}\pi & \succ & t \star \text{k}_\pi{\cdot}\pi \\ \text{(restore)} & \text{k}_\pi \star t{\cdot}\rho & \succ & t \star \pi \end{array}$$

The first two rules implement **weak head reduction**
The other two capture and restore continuations.

Quasi-Proofs and Poles

To define realizability interpretations we need two more ingredients:

Definition

$QP \subseteq \Lambda$ is the set of **quasi-proofs**, i.e. closed terms not containing any k_π.

Observation: t is a quasi-proof iff it does not contain any stack constant π_0

Definition

A **pole** is a set $\perp\!\!\!\perp \subseteq \Lambda \star \Pi$ of processes *closed under inverse reduction*

Different poles $\perp\!\!\!\perp$ give different realizability models ... as we shall see now

Truth values, predicates, ordering on predicates

- Fix a pole $\bot\!\!\!\bot$
- **Truth values** are subsets $P \subseteq \Pi$, elements are 'refutations'
- For $t \in \Lambda$, $P, Q \in \mathcal{P}(\Pi)$ define

$$\begin{array}{lcll} t \Vdash P & \Leftrightarrow & \forall \pi \in P \,.\, t \star \pi \in \bot\!\!\!\bot & (\text{'}t \text{ realizes } P\text{'}) \\ P \Rightarrow Q & = & \{u{\cdot}\pi \mid u \Vdash P, \pi \in Q\} & \end{array}$$

- **Predicates** are families of truth values, i.e. functions $\varphi, \psi : J \to \mathcal{P}(\Pi)$
- Define **ordering** on set $\mathcal{P}(\Pi)^J$ of predicates on J by

$$\varphi \leq \psi \quad :\Leftrightarrow \quad \exists t \in QP \; \forall j \in J \,.\, t \Vdash \varphi(j) \Rightarrow \psi(j)$$

  [Restriction to *quasi-proofs* to avoid degeneracy]
- $(\mathcal{P}(\Pi)^J, \leq)$ is a **Boolean algebra**
- The assignment $J \mapsto (\mathcal{P}(\Pi)^J, \leq)$ extends to a **Boolean tripos**

$$\mathcal{K}_{\bot\!\!\!\bot} : \mathbf{Set}^{\mathrm{op}} \to \mathbf{Ord},$$

  which gives rise to a **Boolean topos** $\mathbf{Set}[\mathcal{K}_{\bot\!\!\!\bot}]$

Consistency

- Falsity is the set Π of *all* stacks
- $\mathcal{K}_{\perp\!\!\!\perp}$ is **consistent**, if there is no quasi-proof realizing Π

$$\begin{aligned} & \neg\exists t \in QP \,.\, t \Vdash \Pi \\ \Leftrightarrow \quad & \neg\exists t \in QP \,\forall \pi \in \Pi \,.\, t \star \pi \in \perp\!\!\!\perp \\ \Leftrightarrow \quad & \forall t \in QP \,\exists \pi \in \Pi \,.\, t \star \pi \notin \perp\!\!\!\perp \end{aligned}$$

- This motivates Krivine's 'thread model'. It is obtained by postulating a stack constant π_t for each quasi-proof t, and defining

$$\perp\!\!\!\perp = \{p \mid \forall t \in QP \,.\, t \star \pi_t \not\rightarrow^* p\}$$

  - Krivine, J.L. *Realizability algebras II: new models of ZF+DC.* 2011

Adding effects

- Krivine uses non-logical instructions to realize **axioms**

> *"Indeed, when we realize usual axioms of mathematics, we need to introduce, one after the other, the very standard tools in system programming: for the law of Peirce, these are continuations (particularly useful for exceptions); for the axiom of dependent choice, these are the clock and the process numbering; for the ultrafilter axiom and the well ordering of* $\mathbb{R}$*, these are no less than I/O instructions on a global memory, in other words assignment." (Realizability algebras II)*

- instructions in Krivine's sense are not real 'side'-effects
- to model actual 'side'-effects, Alexandre Miquel proposes operational semantics based on pairs (p, s) where p is a process and s is a state which can be changed by non-logical instructions
  - Miquel, A. *Classical modal realizability and side effects.* 2009
- Our account is based on Miquel's, but different notion of pole

The Krivine Machine with read/write (IOKAM)

Syntax extended by I/O and termination instructions:

Terms:	t	$::=$	$x \mid \lambda x.t \mid tt \mid \mathsf{cc} \mid \mathsf{k}_\pi \mid \mathsf{r} \mid \mathsf{w1} \mid \mathsf{w0} \mid \mathsf{end}$	
Stacks:	π	$::=$	$\varepsilon \mid t{\cdot}\pi$	(t closed)
Processes:	p	$::=$	$t \star \pi \mid \top$	(t closed)

Operational semantics defined on **configurations** (p, σ, τ) with $\sigma, \tau \in \{0,1\}^*$

$$
\begin{array}{lrcl}
(\tau) & (\ t \star \pi & , \ \iota, \omega) & \rightsquigarrow \ (u \star \rho, \ \iota, \ \omega) \quad \text{whenever } t \star \pi \succ u \star \rho \\
(\mathsf{r0}) & (\ \mathsf{r} \star t{\cdot}u{\cdot}v{\cdot}\pi & , \ 0{\cdot}\iota, \omega) & \rightsquigarrow \ (t \star \pi, \ \iota, \ \omega) \\
(\mathsf{r1}) & (\ \mathsf{r} \star t{\cdot}u{\cdot}v{\cdot}\pi & , \ 1{\cdot}\iota, \omega) & \rightsquigarrow \ (u \star \pi, \ \iota, \ \omega) \\
(\mathsf{r}\varepsilon) & (\ \mathsf{r} \star t{\cdot}u{\cdot}v{\cdot}\pi & , \ \varepsilon, \omega) & \rightsquigarrow \ (v \star \pi, \ \varepsilon, \ \omega) \\
(\mathsf{w0}) & (\ \mathsf{w0} \star t{\cdot}\pi & , \ \iota, \omega) & \rightsquigarrow \ (t \star \pi, \ \iota, \ 0{\cdot}\omega) \\
(\mathsf{w1}) & (\ \mathsf{w1} \star t{\cdot}\pi & , \ \iota, \omega) & \rightsquigarrow \ (t \star \pi, \ \iota, \ 1{\cdot}\omega) \\
(\mathsf{e}) & (\mathsf{end} \star \pi & , \ \iota, \omega) & \rightsquigarrow \ (\top \ , \ \iota, \ \omega)
\end{array}
$$

Intuition:

Model of computation that explicitly includes reading and writing of data – instead of 'values' or Church numerals.

Properties of the IOKAM

Definition

A process p **implements** a partial function $f : \mathbb{N} \rightharpoonup \mathbb{N}$, if we have

$$(p, \mathrm{bin}(n), \varepsilon) \rightsquigarrow^* (\top, \varepsilon, \mathrm{bin}(f(n)))$$

for all $n \in \mathrm{dom}(f)$, where $\mathrm{bin}(n)$ is the binary representation of n.

Theorem (Turing completeness)

Every computable $f : \mathbb{N} \rightharpoonup \mathbb{N}$ is implemented by some process p.

Proof uses **storage operators**, which serve to simulate CBV in CBN.

Redefine ‘quasi-proof’ and ‘pole’ in presence of side effects:

Definition

The set $QP \subseteq \Lambda$ of *quasi-proofs* is the set of closed terms not containing any I/O instructions $\mathsf{r}, \mathsf{w0}, \mathsf{w1}, \mathsf{end}$.

Idea : *Quasi-proofs are ‘purely logical’* (continuations allowed)

Definition

A **pole** is a set $\perp\!\!\!\perp \subseteq \Lambda \star \Pi$ of processes *closed under inverse* ***effect-free*** *reduction* $\succ$.

Compare: For Miquel (loc. cit.), poles are sets of **configurations** closed under all inverse transitions.

Theorem

With these definitions, poles $\perp\!\!\!\perp$ *give rise to triposes* $\mathcal{K}_{\perp\!\!\!\perp}$ *and toposes* $\mathbf{Set}[\mathcal{K}_{\perp\!\!\!\perp}]$ *just as before.*

Consistency in presence of I/O

Theorem

With the new definitions, a pole $\perp\!\!\!\perp \subseteq \Lambda \star \Pi$ is consistent iff every $t * \pi \in \perp\!\!\!\perp \setminus \{\top\}$ contains a non-logical instruction.

Proof.

Recall that consistency means $\forall t \in QP\ \exists \pi \in \Pi\,.\, t \star \pi \notin \perp\!\!\!\perp$.
If all elements of $\perp\!\!\!\perp \setminus \{\top\}$ contain non-logical instructions, then $t \star \varepsilon \notin \perp\!\!\!\perp$ for any quasi-proof t.
Conversely, if $t \star \pi \in \perp\!\!\!\perp$ is 'pure', then $\mathsf{k}_\pi\, t$ is a quasi-proof-realizing Π. □

Poles from specifications

The slogan ‘specifications are poles’ is made precise as follows.

Lemma

If $\perp\!\!\!\perp \subseteq \Lambda\star\Pi$ *is closed under* ***observational equivalence****, then* $\perp\!\!\!\perp$ *is a pole.*

- Paper uses two different notions of equivalence.
- Finer one captures interactivity, based on **weak bisimulation**
- Prime examples do not require interactivity:

Example

Let $f : \mathbb{N} \rightharpoonup \mathbb{N}$ be a computable function. Then the set

$$\perp\!\!\!\perp_f = \{p \mid p \text{ implements } f\}$$

is a pole, which is consistent if f is not everywhere undefined.

Toposes from computable functions

- Poles $\perp\!\!\!\perp_f$ give triposes $\mathcal{K}_f$ and toposes **Set**$[\mathcal{K}_f]$ (for for computable f)
- Plan: use toposes **Set**$[\mathcal{K}_f]$ to study f, in particular wrt complexity
- View **Set**$[\mathcal{K}_f]$ as **geometric** object, more precisely as

'space of solutions to algorithmic problem of computing f'

- Use geometric invariants like **cohomology** to study **shape** of **Set**$[\mathcal{K}_f]$
- Problem: Well-definedness of cohomology not even clear
- Look for other ways of understanding the structure of **Set**$[\mathcal{K}_f]$

First steps in understanding the structure of $\mathbf{Set}[\mathcal{K}_f]$

- Are the categories $\mathbf{Set}[\mathcal{K}_f]$ **Grothendieck toposes**?
  - Probably not, at least not in a 'standard' way
    ($\Delta : \mathbf{Set} \to \mathbf{Set}[\mathcal{K}_f]$ not part of a geometric morphism)
- What is the **finite type hierarchy** in $\mathbf{Set}[\mathcal{K}_f]$?
  - No idea
- What are the **truth values** of $\mathbf{Set}[\mathcal{K}_f]$?
  - Not full picture yet, but some structure of f can be encoded in truth values

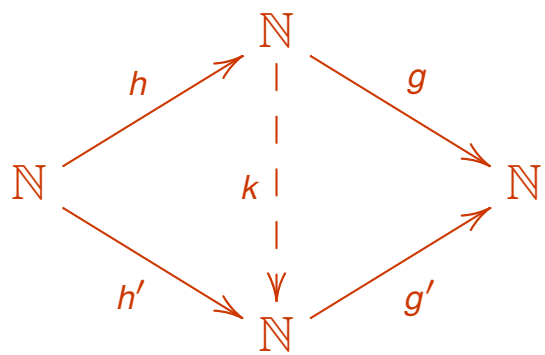
Relating f and $\mathcal{K}_f(1)$

- $\mathcal{K}_f(1) = (\mathcal{P}(\Pi), \leq)$ is the order of truth values in **Set**$[\mathcal{K}_f]$
- Assume f total
- For every **computable factorization** $f = g \circ h$ define truth value

$$P_g = \{t{\cdot}\varepsilon \mid \forall n \in \mathbb{N}\,.\,(t \star \overline{n}{\cdot}\varepsilon, \varepsilon, \varepsilon) \rightsquigarrow^* (\top, \varepsilon, \mathrm{bin}(g(n)))\}$$

Theorem

$P_g \leq P_{g'}$ iff g factors through g' computably.



Thanks for your attention!