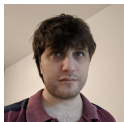


Generic approaches to effective realizability models

Liron COHEN



Ariel GRUNFELD



Dominik KIRST



Étienne MIQUEY



ROSS TATE



CIRM, September 2025

What are *realizability models*?

Realizability

- 1 Provides **models** for theories $\mathcal{Q}$ (such as HA2 / HOL / ZF / ...)

Tarski

$$A \mapsto |A| \in \mathbb{B}$$

(intuition: level of truthness)

Boolean
algebra



Realizability

$$A \mapsto \{t : t \Vdash A\}$$

(intuition: programs whose computational behavior is guided by A)

- 2 a for analyzing programs computational behavior

Realizability

- ❶ Provides **models** for theories $\mathcal{Q}$ (such as HA2 / HOL / ZF / ...)

Tarski

$$A \mapsto |A| \in \mathbb{B}$$

(intuition: level of truthness)

Boolean
algebra

Realizability

$$A \mapsto \{t : t \Vdash A\}$$

(intuition: programs whose computational behavior is guided by A)

- ❷ a **tool** for analyzing programs computational behavior

Realizability, a 3-steps recipe

❶ **formulas** (a.k.a. types)

$\mapsto$ simple types, 2nd – order logic, ZF, ...

❷ a **computational system** (a.k.a. your favorite calculus)

$\mapsto$ some λ – calculus, a combinatory algebra, PCF, etc.

❸ formulas **interpretation**

Adequacy

If $p : (\Gamma \vdash A)$ and $\sigma \Vdash \Gamma$ then $\sigma(p^*) \in |A|$.

Realizability, a 3-steps recipe

next slide

❶ **formulas** (a.k.a. types)

$\mapsto$ simple types, 2^{nd} – order logic, ZF, ...

❷ a **computational system** (a.k.a. your favorite calculus)

$\mapsto$ some λ – calculus, a combinatory algebra, PCF, etc.

❸ formulas **interpretation** (a.k.a. truth values)

$\mapsto |A| = \{t \in \Lambda : t \Vdash A\}$

Adequacy

If $\Gamma \vdash t : A$ and $\sigma \Vdash \Gamma$ then $\sigma(t) \in |A|$.

A simple realizability interpretation

Types & terms:

(excerpt)

<i>1st-order exp.</i>	$e ::= x \mid 0 \mid S(e) \mid f(e_1, \dots, e_n)$
<i>Formulas</i>	$A, B ::= \text{Nat}(e) \mid X(e_1, \dots, e_n) \mid A \rightarrow B \mid \dots$ $\mid \forall x.A \mid \exists x.A \mid \forall X.A \mid \exists X.A$
<i>Terms</i>	$t, u ::= x \mid 0 \mid \mathbf{succ} \mid \mathbf{rec} \mid \lambda x.t \mid t u \mid \dots$

where $f : \mathbb{N}^n \rightarrow \mathbb{N}$ is any arithmetical function.

Typing rules:

(excerpt)

$\overline{\Gamma \vdash 0 : \text{Nat}(0)}$	$\overline{\Gamma \vdash \mathbf{rec} : \forall Z. Z(0) \rightarrow (\forall^{\mathbb{N}} y. (Z(y) \rightarrow Z(S(y)))) \rightarrow \forall^{\mathbb{N}} x. Z(x)}$	
$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x. t : A \rightarrow B}$	$\frac{\Gamma \vdash t : A \rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t u : B}$	$\frac{\Gamma \vdash t : A[x := n]}{\Gamma \vdash t : \exists x. A}$
$\frac{\Gamma \vdash t : \forall x. A}{\Gamma \vdash t : A[x := n]}$	$\frac{\Gamma \vdash t : A \quad x \notin FV(\Gamma)}{\Gamma \vdash t : \forall x. A}$	$\frac{\Gamma \vdash t : A[X(x_1, \dots, x_n) := B]}{\Gamma \vdash t : \exists X. A}$

A simple realizability interpretation

Types & terms:

(excerpt)

<i>1st-order exp.</i>	$e ::= x \mid 0 \mid S(e) \mid f(e_1, \dots, e_n)$
<i>Formulas</i>	$A, B ::= \text{Nat}(e) \mid X(e_1, \dots, e_n) \mid A \rightarrow B \mid \dots$ $\mid \forall x.A \mid \exists x.A \mid \forall X.A \mid \exists X.A$
<i>Terms</i>	$t, u ::= x \mid 0 \mid \text{succ} \mid \text{rec} \mid \lambda x.t \mid t u \mid \dots$

where $f : \mathbb{N}^n \rightarrow \mathbb{N}$ is any arithmetical function.

Typing rules:

(excerpt)

...

Reductions:

$$\overline{(\lambda x.t)u \triangleright_{\beta} t[u/x]} \quad \overline{\text{rec } u_0 u_1 (\text{succ } t) \triangleright_{\beta} u_1 t (\text{rec } u_0 u_1 t)} \quad \dots$$

A simple realizability interpretation

Realizability interpretation:

$$\begin{aligned} |\text{Nat}(e)|_\rho &\triangleq \{t \in \Lambda : t \triangleright^* \text{succ}^n 0, \text{ where } n = \llbracket e \rrbracket_\rho\} \\ |X(e_1, \dots, e_n)|_\rho &\triangleq \rho(X)(\llbracket e_1 \rrbracket_\rho, \dots, \llbracket e_n \rrbracket_\rho) \\ |A \rightarrow B|_\rho &\triangleq \{t \in \Lambda : \forall u \in |A|_\rho. (t u \in |B|_\rho)\} \\ |\forall x. A|_\rho &\triangleq \bigcap_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\ |\exists x. A|_\rho &\triangleq \bigcup_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\ |\forall X. A|_\rho &\triangleq \bigcap_{F: \mathbb{N}^k \rightarrow \text{SAT}} |A|_{\rho, X \leftarrow F} \\ |\exists X. A|_\rho &\triangleq \bigcup_{F: \mathbb{N}^k \rightarrow \text{SAT}} |A|_{\rho, X \leftarrow F} \end{aligned}$$

Adequacy

If $\Gamma \vdash t : A$ and $\sigma \Vdash \Gamma$ then $\sigma(t) \in |A|$.

Key ideas:

- realizers

A simple realizability interpretation

Realizability interpretation:

$$\begin{aligned} |\text{Nat}(e)|_\rho &\triangleq \{t \in \Lambda : t \triangleright^* \text{succ}^n 0, \text{ where } n = \llbracket e \rrbracket_\rho\} \\ |X(e_1, \dots, e_n)|_\rho &\triangleq \rho(X)(\llbracket e_1 \rrbracket_\rho, \dots, \llbracket e_n \rrbracket_\rho) \\ |A \rightarrow B|_\rho &\triangleq \{t \in \Lambda : \forall u \in |A|_\rho. (t u \in |B|_\rho)\} \\ |\forall x. A|_\rho &\triangleq \bigcap_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\ |\exists x. A|_\rho &\triangleq \bigcup_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\ |\forall X. A|_\rho &\triangleq \bigcap_{F: \mathbb{N}^k \rightarrow \text{SAT}} |A|_{\rho, X \leftarrow F} \\ |\exists X. A|_\rho &\triangleq \bigcup_{F: \mathbb{N}^k \rightarrow \text{SAT}} |A|_{\rho, X \leftarrow F} \end{aligned}$$

Key ideas:

- realizers *compute*
- realizers *defend the validity* of their formula
- truth values are *saturated*: $t \triangleright^* t' \wedge t' \in |A| \Rightarrow t \in |A|$

A simple realizability interpretation

Realizability interpretation:

$$\begin{aligned} |\text{Nat}(e)|_\rho &\triangleq \{t \in \Lambda : t \triangleright^* \text{succ}^n 0, \text{ where } n = \llbracket e \rrbracket_\rho\} \\ |X(e_1, \dots, e_n)|_\rho &\triangleq \rho(X)(\llbracket e_1 \rrbracket_\rho, \dots, \llbracket e_n \rrbracket_\rho) \\ |A \rightarrow B|_\rho &\triangleq \{t \in \Lambda : \forall u \in |A|_\rho. (t u \in |B|_\rho)\} \\ |\forall x. A|_\rho &\triangleq \bigcap_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\ |\exists x. A|_\rho &\triangleq \bigcup_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\ |\forall X. A|_\rho &\triangleq \bigcap_{F: \mathbb{N}^k \rightarrow \text{SAT}} |A|_{\rho, X \leftarrow F} \\ |\exists X. A|_\rho &\triangleq \bigcup_{F: \mathbb{N}^k \rightarrow \text{SAT}} |A|_{\rho, X \leftarrow F} \end{aligned}$$

Key ideas:

- realizers *compute*
- realizers *defend the validity* of their formula

- truth values are *saturated*: $t \triangleright^* t' \wedge t' \in |A| \Rightarrow t \in |A|$

A simple realizability interpretation

Realizability interpretation:

$$\begin{aligned} |\text{Nat}(e)|_\rho &\triangleq \{t \in \Lambda : t \triangleright^* \mathbf{succ}^n 0, \text{ where } n = \llbracket e \rrbracket_\rho\} \\ |X(e_1, \dots, e_n)|_\rho &\triangleq \rho(X)(\llbracket e_1 \rrbracket_\rho, \dots, \llbracket e_n \rrbracket_\rho) \\ |A \rightarrow B|_\rho &\triangleq \{t \in \Lambda : \forall u \in |A|_\rho. (t u \in |B|_\rho)\} \\ |\forall x. A|_\rho &\triangleq \bigcap_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\ |\exists x. A|_\rho &\triangleq \bigcup_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\ |\forall X. A|_\rho &\triangleq \bigcap_{F: \mathbb{N}^k \rightarrow \mathbf{SAT}} |A|_{\rho, X \leftarrow F} \\ |\exists X. A|_\rho &\triangleq \bigcup_{F: \mathbb{N}^k \rightarrow \mathbf{SAT}} |A|_{\rho, X \leftarrow F} \end{aligned}$$

Key ideas:

- realizers *compute*
- realizers *defend the validity* of their formula

- truth values are *saturated*: $t \triangleright^* t' \wedge t' \in |A| \Rightarrow t \in |A|$

A simple realizability interpretation

Realizability interpretation:

$$\begin{aligned} |\text{Nat}(e)|_\rho &\triangleq \{t \in \Lambda : t \triangleright^* \text{succ}^n 0, \text{ where } n = \llbracket e \rrbracket_\rho\} \\ |X(e_1, \dots, e_n)|_\rho &\triangleq \rho(X)(\llbracket e_1 \rrbracket_\rho, \dots, \llbracket e_n \rrbracket_\rho) \\ |A \rightarrow B|_\rho &\triangleq \{t \in \Lambda : \forall u \in |A|_\rho. (t u \in |B|_\rho)\} \\ |\forall x. A|_\rho &\triangleq \bigcap_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\ |\exists x. A|_\rho &\triangleq \bigcup_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\ |\forall X. A|_\rho &\triangleq \bigcap_{F: \mathbb{N}^k \rightarrow \text{SAT}} |A|_{\rho, X \leftarrow F} \\ |\exists X. A|_\rho &\triangleq \bigcup_{F: \mathbb{N}^k \rightarrow \text{SAT}} |A|_{\rho, X \leftarrow F} \end{aligned}$$

Key ideas:

- realizers *compute*
- realizers *defend the validity* of their formula

- truth values are *saturated*: $t \triangleright^* t' \wedge t' \in |A| \Rightarrow t \in |A|$

Formal definition

?

Formal definition?

iki/Realizability



Realizability

4 languages ▾

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

From Wikipedia, the free encyclopedia

In [mathematical logic](#), **realizability** is a collection of methods in [proof theory](#) used to study [constructive proofs](#) and extract additional information from them.^[1] Formulas from a formal theory are "realized" by objects, known as "realizers", in a way that knowledge of the [realizer gives knowledge about the truth of the formula](#). There are many variations of realizability; exactly which class of formulas is studied and which objects are realizers differ from one variation to another.

Realizability can be seen as a formalization of the [BHK interpretation](#) of intuitionistic logic; in realizability the notion of "proof" (which is left undefined in the BHK interpretation) is replaced with a formal notion of "realizer". [Most variants of realizability begin with a theorem that any statement that is provable in the formal system being studied is realizable](#). The realizer, however, usually gives more information about the formula than a formal proof would directly provide.

Beyond giving insight into intuitionistic provability, realizability can be applied to prove the [disjunction and existence properties](#) for intuitionistic theories and to extract programs from proofs, as in [proof mining](#). It is also related to [topos theory](#) via the [realizability topos](#).

Example: Kleene's 1945-realizability [\[edit \]](#)

[Kleene](#)'s original version of realizability uses natural numbers as realizers for formulas in [Heyting arithmetic](#). A few pieces of notation are required: first, an ordered pair (n,m) is treated as a single number using a fixed [primitive recursive pairing function](#); second, for each natural number n , φ_n is the [computable function](#) with index n . The following clauses are used to define a relation " n realizes A "

Formal definition?

wiki/Boolean_algebra_(structure)



Definition [\[edit \]](#)

A **Boolean algebra** is a six-tuple consisting of a [set](#) A , equipped with two [binary operations](#) $\wedge$ (called "meet" or "and"), $\vee$ (called "join" or "or"), a [unary operation](#) $\neg$ (called "complement" or "not") and two elements 0 and 1 in A (called "bottom" and "top", or "least" and "greatest" element, also denoted by the symbols $\perp$ and $\top$, respectively), such that for all elements a , b and c of A , the following [axioms](#) hold:^[2]

$a \vee (b \vee c) = (a \vee b) \vee c$	$a \wedge (b \wedge c) = (a \wedge b) \wedge c$	associativity
$a \vee b = b \vee a$	$a \wedge b = b \wedge a$	commutativity
$a \vee (a \wedge b) = a$	$a \wedge (a \vee b) = a$	absorption
$a \vee 0 = a$	$a \wedge 1 = a$	identity
$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$	$a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$	distributivity
$a \vee \neg a = 1$	$a \wedge \neg a = 0$	complements

Note, however, that the absorption law and even the associativity law can be excluded from the set of axioms as they can be derived from the other axioms (see [Proven properties](#)).

A Boolean algebra with only one element is called a **trivial Boolean algebra** or a **degenerate Boolean algebra**. (In older works, some authors required 0 and 1 to be *distinct* elements in order to exclude this case.)^{[\[citation needed\]](#)}

It follows from the last three pairs of axioms above (identity, distributivity and complements), or from the absorption axiom, that

$$a = b \wedge a \quad \text{if and only if} \quad a \vee b = b.$$

The relation $\leq$ defined by $a \leq b$ if these equivalent conditions hold, is a [partial order](#) with least element 0 and greatest element 1. The meet $a \wedge b$ and the join $a \vee b$ of two elements coincide with their [infimum](#) and [supremum](#), respectively, with respect to $\leq$.

The first four pairs of axioms constitute a definition of a [bounded lattice](#).

It follows from the first five pairs of axioms that any complement is unique.

The set of axioms is [self-dual](#) in the sense that if one exchanges $\vee$ with $\wedge$ and 0 with 1 in an axiom, the result is again an axiom.

realizability topos

Contents

[1. Idea](#)

[2. Constructions](#)

[Via tripos theory](#)

[Via assemblies](#)

[3. Properties](#)

[Axiomatic characterization](#)

[4. Related concepts](#)

[5. References](#)

Context

Topos Theory

**Constructivism,
Realizability,
Computability**

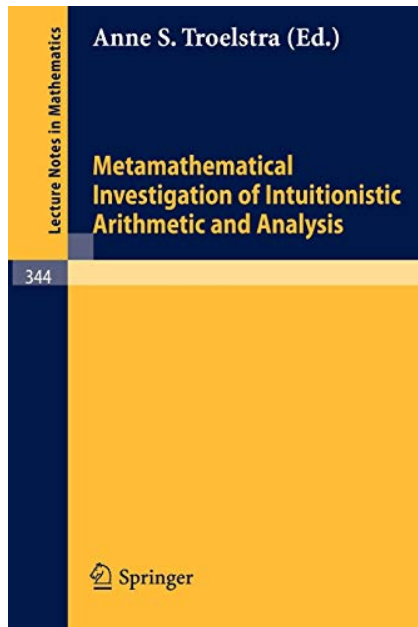
1. Idea

A realizability topos is a [topos](#) which embodies the [realizability interpretation](#) of [intuitionistic number theory](#) (due to Kleene) as part of its [internal logic](#). Realizability toposes form an important class of [elementary toposes](#) that are not [Grothendieck toposes](#), and don't even have a [geometric morphism](#) to [Set](#).

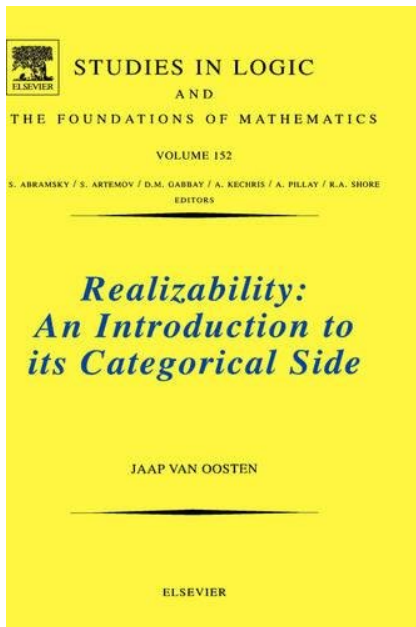
The input datum for forming a realizability topos is a [partial combinatory algebra](#), or PCA.

- When the PCA is [Kleene's first algebra](#) $\mathcal{K}_1$, the resulting topos is called the [effective topos](#) $\mathbf{RT}(\mathcal{K}_1)$.

Formal definition?

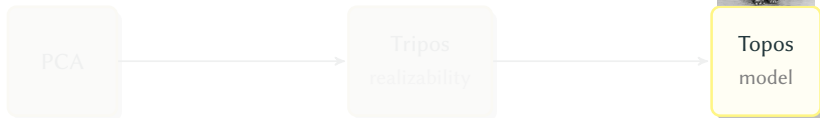


Formal definition?



Topoi, Tripases and PCAs

Pitts' PhD (80s)



A realizability model

$$\mathcal{M} \models A \quad \Leftrightarrow \quad \exists t. t \Vdash A$$

Goal #1

Introduce an intermediate structure that connects
the **logical** and **computational** aspects

Topoi, Tripases and PCAs

Pitts' PhD (80s)



- A functor $\mathcal{T} : \mathbf{Set}^{op} \rightarrow \mathbf{HpA}$ with:
 - quantifiers
 - computability w. substitutions
 - generic predicate

Goal #1

Introduce an intermediate structure that connects
the **logical** and **computational** aspects

Topoi, Triples and PCAs

Pitts' PhD (80s)



Codes: a set C

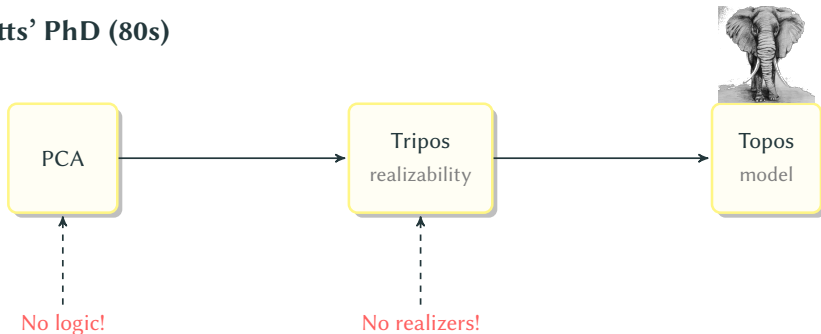
Application: a partial binary operator
 $\cdot : C \times C \rightarrow C$ with the functional
completeness property.

Goal #1

Introduce an intermediate structure that connects
the **logical** and **computational** aspects

Topoi, Triposes and PCAs

Pitts' PhD (80s)

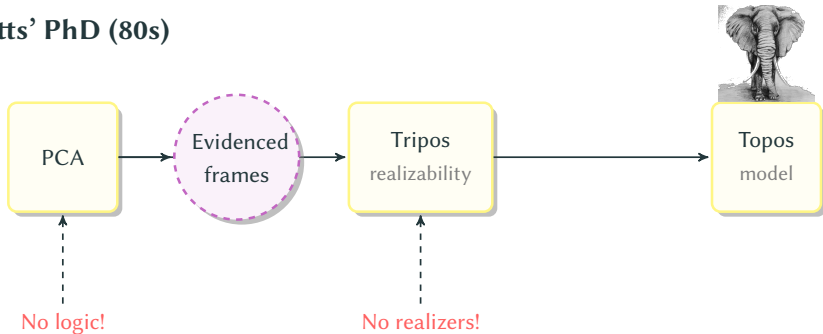


Goal #1

Introduce an intermediate structure that connects the **logical** and **computational** aspects

This Talk: Goal #1

Pitts' PhD (80s)



Goal #1

Introduce an intermediate structure that connects
the **logical** and **computational** aspects

Computational Choices Matter

With side-effects come new reasoning principles:

- exceptions ~ Markov's principle
- control operators ~ classical logic
- quote instruction ~ dependent choice
- memoization ~ dependent choice
- monotonic memory ~ Cohen's forcing
- monotonic memory ~ nonstandard analysis
- ...

But PCAs can only support non-termination!

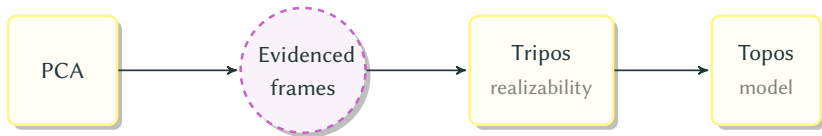
Computational Choices Matter

With side-effects come new reasoning principles:

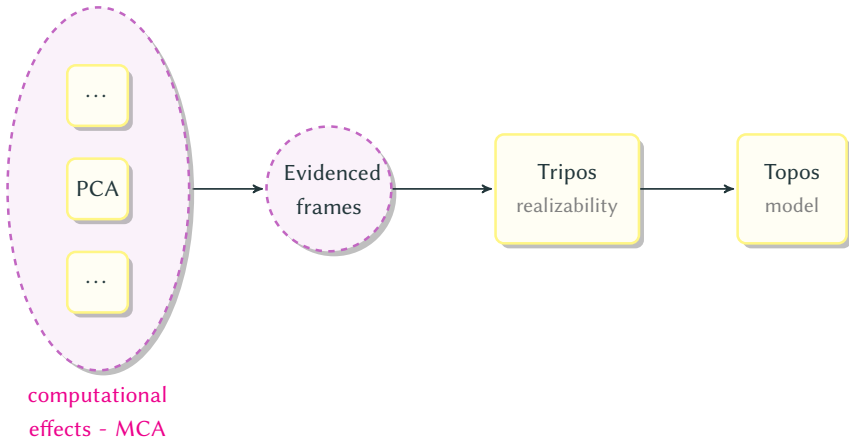
- exceptions ~ Markov's principle
- control operators ~ classical logic
- quote instruction ~ dependent choice
- memoization ~ dependent choice
- monotonic memory ~ Cohen's forcing
- monotonic memory ~ nonstandard analysis
- ...

But PCAs can only support non-termination!

This Talk: Goal #2



This Talk: Goal #2



Goal #2

Smooth integration of useful computational effects

Realizability, the historical recipe

Partial Combinatory Algebras

Definition - PAS

A **Partial Applicative Structure (PAS)** $(\mathbb{A}, \cdot)$ is given by:

- a set $\mathbb{A}$ of “codes”,
- a partial application $\cdot : \mathbb{A} \times \mathbb{A} \rightarrow \mathbb{A}$.

Notations:

- $c_f \cdot c_a \downarrow c_r$ means that c_r is the (successful) result of applying c_f to c_a ,
- $c_f \cdot c_a \downarrow$ means that there exists some $c_r \in \mathbb{A}$ such that $c_f \cdot c_a \downarrow c_r$.

Definition - PCA

A **Partial Combinatory Algebra** is a PAS that is “functionally complete”.

$\forall$ For every expression e with n free variables, there is a code $\langle \lambda_n.e \rangle \in \mathbb{A}$ such that:

$$\lambda_{n+1}.e \cdot c_a \downarrow \lambda_n.e\{c_a\} \quad \text{and} \quad \lambda_0.e \cdot c_a \downarrow c_r \iff e\{c_a\} \downarrow c_r$$

Partial Combinatory Algebras

Definition - PAS

A **Partial Applicative Structure (PAS)** $(\mathbb{A}, \cdot)$ is given by:

- a set $\mathbb{A}$ of “codes”,
- a partial application $\cdot : \mathbb{A} \times \mathbb{A} \rightarrow \mathbb{A}$.

Notations:

- $c_f \cdot c_a \downarrow c_r$ means that c_r is the (successful) result of applying c_f to c_a ,
- $c_f \cdot c_a \downarrow$ means that there exists some $c_r \in \mathbb{A}$ such that $c_f \cdot c_a \downarrow c_r$.

Definition - PCA

A **Partial Combinatory Algebra** is a PAS that is “functionally complete”.

$\Uparrow$ For every expression e with n free variables, there is a code $\langle \lambda_n.e \rangle \in \mathbb{A}$ such that:

$$\lambda_{n+1}.e \cdot c_a \downarrow \lambda_n.e\{c_a\} \quad \text{and} \quad \lambda_0.e \cdot c_a \downarrow c_r \iff e\{c_a\} \downarrow c_r$$

Partial Combinatory Algebras: examples

Examples of PCAs:

- Kleene's first model $\mathcal{K}_1$

$\mathbb{N}$ with partial recursive application $a, b \mapsto \varphi_a(b)$,

- Kleene's first model $\mathcal{K}_2$

$\mathbb{N}^{\mathbb{N}}$ with application $F : \mathbb{N}^{\mathbb{N}} \times \mathbb{N}^{\mathbb{N}} \rightarrow \mathbb{N}^{\mathbb{N}}$,

- Scott graph model: $\mathcal{P}(\mathbb{N})$ as a graph:

$\text{graph} : F \mapsto \{\langle n, m \rangle \mid m \in F(e_n)\} \quad \in (\mathcal{P}(\mathbb{N}) \rightarrow \mathcal{P}(\mathbb{N})) \rightarrow \mathcal{P}(\mathbb{N})$

$\text{fun} : A \mapsto B \mapsto \{\langle m \mid \exists n. e_n \subseteq B \wedge \langle n, m \rangle \in A\} \quad \in \mathcal{P}(\mathbb{N}) \rightarrow \mathcal{P}(\mathbb{N}) \rightarrow \mathcal{P}(\mathbb{N})$

- Domains: $D \simeq D \rightarrow D$
- Any model $\mathcal{M}$ of Peano Arithmetic:

$$(ab = c) \triangleq \mathcal{M} \models \exists y. T(a, b, y) \wedge U(y) = c$$

Historical factory: $\mathcal{P}(\mathbb{A})$ -valued predicate

Predicates

A $\mathcal{P}(\mathbb{A})$ -valued predicate on a set X is a function $\varphi : X \rightarrow \mathcal{P}(\mathbb{A})$.

$a \in \varphi(x)$: a is a realizer of $\varphi(x)$. We can define:

- $(\varphi \Rightarrow \psi)(x) = \{a \in \mathbb{A} \mid \forall b \in \varphi(x), ab \downarrow \text{ and } ab \in \psi(x)\},$
- $(\varphi \wedge \psi)(x) = \{\langle a, b \rangle \in \mathbb{A} \mid a \in \varphi(x), b \in \psi(x)\},$
- $(\varphi \vee \psi)(x) = \{\iota_1(a) \in \mathbb{A} \mid a \in \varphi(x)\} \cup \{\iota_2(b) \in \mathbb{A} \mid b \in \psi(x)\}.$

Entailment

We pose $\varphi \preceq \psi \triangleq \exists a \in \mathbb{A}, \forall x \in X, \forall b \in \varphi(x), ab \downarrow \text{ and } ab \in \psi(x).$

Proposition

$(\mathcal{P}(\mathbb{A}), \preceq)$ is a pre-Heyting algebra.

Historical factory: $\mathcal{P}(\mathbb{A})$ -valued predicate

Predicates

A $\mathcal{P}(\mathbb{A})$ -valued predicate on a set X is a function $\varphi : X \rightarrow \mathcal{P}(\mathbb{A})$.

$a \in \varphi(x)$: a is a realizer of $\varphi(x)$. We can define:

- $(\varphi \Rightarrow \psi)(x) = \{a \in \mathbb{A} \mid \forall b \in \varphi(x), ab \downarrow \text{ and } ab \in \psi(x)\},$
- $(\varphi \wedge \psi)(x) = \{\langle a, b \rangle \in \mathbb{A} \mid a \in \varphi(x), b \in \psi(x)\},$
- $(\varphi \vee \psi)(x) = \{\iota_1(a) \in \mathbb{A} \mid a \in \varphi(x)\} \cup \{\iota_2(b) \in \mathbb{A} \mid b \in \psi(x)\}.$

Entailment

We pose $\varphi \preceq \psi \triangleq \exists a \in \mathbb{A}, \forall x \in X, \forall b \in \varphi(x), ab \downarrow \text{ and } ab \in \psi(x).$

Proposition

$(\mathcal{P}(\mathbb{A}), \preceq)$ is a pre-Heyting algebra.

Historical factory: $\mathcal{P}(\mathbb{A})$ -valued predicate

Predicates

A $\mathcal{P}(\mathbb{A})$ -valued predicate on a set X is a function $\varphi : X \rightarrow \mathcal{P}(\mathbb{A})$.

$a \in \varphi(x)$: a is a realizer of $\varphi(x)$. We can define:

- $(\varphi \Rightarrow \psi)(x) = \{a \in \mathbb{A} \mid \forall b \in \varphi(x), ab \downarrow \text{ and } ab \in \psi(x)\},$
- $(\varphi \wedge \psi)(x) = \{\langle a, b \rangle \in \mathbb{A} \mid a \in \varphi(x), b \in \psi(x)\},$
- $(\varphi \vee \psi)(x) = \{\iota_1(a) \in \mathbb{A} \mid a \in \varphi(x)\} \cup \{\iota_2(b) \in \mathbb{A} \mid b \in \psi(x)\}.$

Entailment

We pose $\varphi \preceq \psi \triangleq \exists a \in \mathbb{A}, \forall x \in X, \forall b \in \varphi(x), ab \downarrow \text{ and } ab \in \psi(x).$

Proposition

$(\mathcal{P}(\mathbb{A}), \preceq)$ is a pre-Heyting algebra.

Historical factory: $\mathcal{P}(\mathbb{A})$ -valued predicate

Predicates

A $\mathcal{P}(\mathbb{A})$ -valued predicate on a set X is a function $\varphi : X \rightarrow \mathcal{P}(\mathbb{A})$.

$a \in \varphi(x)$: a is a realizer of $\varphi(x)$. We can define:

- $(\varphi \Rightarrow \psi)(x) = \{a \in \mathbb{A} \mid \forall b \in \varphi(x), ab \downarrow \text{ and } ab \in \psi(x)\},$
- $(\varphi \wedge \psi)(x) = \{\langle a, b \rangle \in \mathbb{A} \mid a \in \varphi(x), b \in \psi(x)\},$
- $(\varphi \vee \psi)(x) = \{\iota_1(a) \in \mathbb{A} \mid a \in \varphi(x)\} \cup \{\iota_2(b) \in \mathbb{A} \mid b \in \psi(x)\}.$

Entailment

We pose $\varphi \preceq \psi \triangleq \exists a \in \mathbb{A}, \forall x \in X, \forall b \in \varphi(x), ab \downarrow \text{ and } ab \in \psi(x).$

Proposition

$(\mathcal{P}(\mathbb{A}), \preceq)$ is a pre-Heyting algebra.

Models and triposes

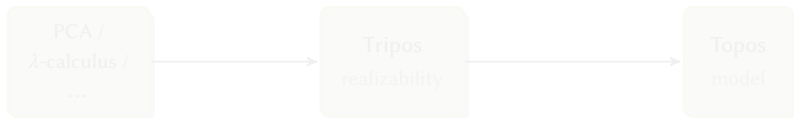
Realizability model

$$\mathcal{M} \models A \quad \Leftrightarrow \quad \exists t. t \Vdash A$$

Categorically speaking

a topos

Triposes



Models and triposes

Realizability model

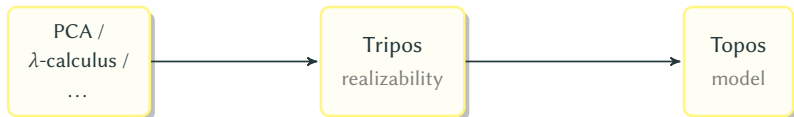
$$\mathcal{M} \models A \iff \exists t. t \Vdash A$$

Categorically speaking



a topos

Triposes



Models and triposes

A *tripos* is a functor $\mathcal{T} : \mathbf{Set}^{op} \rightarrow \mathbf{HpA}$ s.t.:

$$\begin{array}{l} \hookrightarrow \text{Intuition} \quad \mathcal{T}(\Gamma) : \text{predicates } \varphi(\vec{x}) \quad s^* \left(\underbrace{\varphi}_{\in \mathcal{T}(\Gamma')} \right) \triangleq \varphi \left(\underbrace{s}_{s: \Gamma \rightarrow \Gamma'} (\vec{y}) \right) \end{array}$$

Quantifiers. Any s^* has left/right adjoints $\coprod_s / \prod_s : \mathcal{T}(\Gamma) \rightarrow \mathcal{T}(\Gamma')$:

$$\varphi \leq s^*(\psi) \quad \Leftrightarrow \quad \coprod_s(\varphi) \leq \psi \quad s^*(\varphi) \leq \psi \quad \Leftrightarrow \quad \varphi \leq \prod_s(\psi)$$

Compatibility w. substitutions.

$$\text{If } \begin{array}{ccc} \Gamma & \xrightarrow{r} & \Gamma' \\ v \downarrow \lrcorner & & \downarrow u \\ \Gamma'' & \xrightarrow{s} & \Gamma''' \end{array} \quad \text{then } \prod_v \circ r^* = s^* \circ \prod_u \text{ and } s^* \circ \coprod_u = \coprod_v \circ r^*.$$

Generic predicate There exists $\Omega \in \mathbf{Set}$ and $\text{holds} \in \mathcal{T}(\Omega)$, s.t.:

for all $\phi \in \mathcal{T}(\Gamma)$, we have $\chi_\phi : \Gamma \rightarrow \Omega$ satisfying

$$\chi_{s^*(\text{holds})} = \phi \quad \chi_{s^*(\text{holds})} = s$$

Models and triposes

A *tripos* is a **functor** $\mathcal{T} : \mathbf{Set}^{op} \rightarrow \mathbf{HpA}$ s.t.:

$$\leadsto \text{Intuition} \quad \mathcal{T}(\Gamma) : \text{predicates } \varphi(\vec{x}) \quad s^* \left(\underbrace{\varphi}_{\in \mathcal{T}(\Gamma')} \right) \triangleq \varphi \left(\underbrace{s}_{s: \Gamma \rightarrow \Gamma'} (\vec{y}) \right)$$

Quantifiers. Any s^* has left/right adjoints $\coprod_s / \prod_s : \mathcal{T}(\Gamma) \rightarrow \mathcal{T}(\Gamma')$:

$$\varphi \leq s^*(\psi) \iff \coprod_s(\varphi) \leq \psi \quad s^*(\varphi) \leq \psi \iff \varphi \leq \prod_s(\psi)$$

Compatibility w. substitutions.

$$\text{If } \begin{array}{ccc} \Gamma & \xrightarrow{r} & \Gamma' \\ v \downarrow \lrcorner & & \downarrow u \\ \Gamma'' & \xrightarrow{s} & \Gamma''' \end{array} \quad \text{then } \prod_v \circ r^* = s^* \circ \prod_u \text{ and } s^* \circ \coprod_u = \coprod_v \circ r^*.$$

Generic predicate There exists $\Omega \in \mathbf{Set}$ and **holds** $\in \mathcal{T}(\Omega)$, s.t.:

for all $\phi \in \mathcal{T}(\Gamma)$, we have $\chi_\phi : \Gamma \rightarrow \Omega$ satisfying

$$\chi_{s^*(\text{holds})} = \phi \quad \chi_{s^*(\text{holds})} = s$$

Models and triposes

A *tripos* is a functor $\mathcal{T} : \mathbf{Set}^{op} \rightarrow \mathbf{HpA}$ s.t.:

$$\hookrightarrow \text{Intuition} \quad \mathcal{T}(\Gamma) : \text{predicates } \varphi(\vec{x}) \quad s^*(\varphi) \triangleq \varphi(s(\vec{y}))$$

Quantifiers. Any s^* has left/right adjoints $\coprod_s / \prod_s : \mathcal{T}(\Gamma) \rightarrow \mathcal{T}(\Gamma')$:

$$\varphi \leq s^*(\psi) \Leftrightarrow \coprod_s(\varphi) \leq \psi \quad s^*(\varphi) \leq \psi \Leftrightarrow \varphi \leq \prod_s(\psi)$$

Compatibility w. substitutions.

$$\text{If } \begin{array}{ccc} \Gamma & \xrightarrow{r} & \Gamma' \\ v \downarrow & \lrcorner & \downarrow u \\ \Gamma'' & \xrightarrow{s} & \Gamma''' \end{array} \text{ then } \prod_v \circ r^* = s^* \circ \prod_u \text{ and } s^* \circ \coprod_u = \coprod_v \circ r^*.$$

Generic predicate There exists $\Omega \in \mathbf{Set}$ and $\text{holds} \in \mathcal{T}(\Omega)$, s.t.:

for all $\phi \in \mathcal{T}(\Gamma)$, we have $\chi_\phi : \Gamma \rightarrow \Omega$ satisfying

$$\chi_\phi^*(\text{holds}) = \phi \quad \chi_{s^*(\text{holds})} = s$$

Models and triposes

A *tripos* is a functor $\mathcal{T} : \mathbf{Set}^{op} \rightarrow \mathbf{HpA}$ s.t.:

$$\hookrightarrow \text{Intuition} \quad \mathcal{T}(\Gamma) : \text{predicates } \varphi(\vec{x}) \quad s^*(\varphi) \triangleq \varphi(s(\vec{y}))$$

Quantifiers. Any s^* has left/right adjoints $\coprod_s / \prod_s : \mathcal{T}(\Gamma) \rightarrow \mathcal{T}(\Gamma')$:

$$\varphi \leq s^*(\psi) \Leftrightarrow \coprod_s(\varphi) \leq \psi \quad s^*(\varphi) \leq \psi \Leftrightarrow \varphi \leq \prod_s(\psi)$$

$\hookrightarrow \text{Intuition } \exists x^X. - : \mathcal{T}(\Gamma \times X) \rightarrow \mathcal{T}(\Gamma) = \text{left adjoint to } \pi_{\Gamma, X} : \Gamma \times X \rightarrow \Gamma;$

$$\forall \vec{y}, x. [\varphi(\vec{y}, x) \Rightarrow \psi(\vec{y})] \Leftrightarrow \forall \vec{y}. [\exists x. \varphi(\vec{y}, x) \Rightarrow \psi(\vec{y})]$$

Compatibility w. substitutions.

$$\text{If } \begin{array}{ccc} \Gamma & \xrightarrow{r} & \Gamma' \\ v \downarrow \lrcorner & & \downarrow u \\ \Gamma'' & \xrightarrow{s} & \Gamma''' \end{array} \text{ then } \prod_v \circ r^* = s^* \circ \prod_u \text{ and } s^* \circ \coprod_u = \coprod_v \circ r^*.$$

Generic predicate There exists $\Omega \in \mathbf{Set}$ and $\text{holds} \in \mathcal{T}(\Omega)$, s.t.:

Models and triposes

A *tripos* is a functor $\mathcal{T} : \mathbf{Set}^{op} \rightarrow \mathbf{HpA}$ s.t.:

Quantifiers. Any s^* has left/right adjoints $\coprod_s / \prod_s : \mathcal{T}(\Gamma) \rightarrow \mathcal{T}(\Gamma')$:

$$\varphi \leq s^*(\psi) \quad \Leftrightarrow \quad \coprod_s(\varphi) \leq \psi \qquad s^*(\varphi) \leq \psi \quad \Leftrightarrow \quad \varphi \leq \prod_s(\psi)$$

Compatibility w. substitutions.

$$\text{If } \begin{array}{ccc} \Gamma & \xrightarrow{r} & \Gamma' \\ v \downarrow & \lrcorner & \downarrow u \\ \Gamma'' & \xrightarrow{s} & \Gamma''' \end{array} \quad \text{then } \prod_v \circ r^* = s^* \circ \prod_u \text{ and } s^* \circ \coprod_u = \coprod_v \circ r^*.$$

$$\leadsto \text{Intuition} \qquad (\exists x. \varphi(y, x)) [y := s(y')] = \exists x. \varphi(s(y'), x)$$

Generic predicate There exists $\Omega \in \mathbf{Set}$ and $\text{holds} \in \mathcal{T}(\Omega)$, s.t.:

for all $\phi \in \mathcal{T}(\Gamma)$, we have $\chi_\phi : \Gamma \rightarrow \Omega$ satisfying

$$\chi_\phi^*(\text{holds}) = \phi \qquad \chi_{s^*(\text{holds})} = s$$

Models and triposes

A *tripos* is a functor $\mathcal{T} : \mathbf{Set}^{op} \rightarrow \mathbf{HpA}$ s.t.:

Quantifiers. Any s^* has left/right adjoints $\coprod_s / \prod_s : \mathcal{T}(\Gamma) \rightarrow \mathcal{T}(\Gamma')$:

$$\varphi \leq s^*(\psi) \quad \Leftrightarrow \quad \coprod_s(\varphi) \leq \psi \qquad s^*(\varphi) \leq \psi \quad \Leftrightarrow \quad \varphi \leq \prod_s(\psi)$$

Compatibility w. substitutions.

$$\text{If } \begin{array}{ccc} \Gamma & \xrightarrow{r} & \Gamma' \\ v \downarrow & \lrcorner & \downarrow u \\ \Gamma'' & \xrightarrow{s} & \Gamma''' \end{array} \quad \text{then } \prod_v \circ r^* = s^* \circ \prod_u \text{ and } s^* \circ \coprod_u = \coprod_v \circ r^* .$$

Generic predicate There exists $\Omega \in \mathbf{Set}$ and $\text{holds} \in \mathcal{T}(\Omega)$, s.t.:

for all $\phi \in \mathcal{T}(\Gamma)$, we have $\chi_\phi : \Gamma \rightarrow \Omega$ satisfying

$$\chi_\phi^*(\text{holds}) = \phi \qquad \chi_{s^*(\text{holds})} = s$$

$\hookrightarrow$ *Intuition* Ω = set of propositions $\text{holds}(\chi_\phi(\vec{x})) \equiv \phi(\vec{x})$

Models and triposes

A *tripos* is a functor $\mathcal{T} : \mathbf{Set}^{op} \rightarrow \mathbf{HpA}$ s.t.:

Quantifiers. Any s^* has left/right adjoints $\coprod_s / \prod_s : \mathcal{T}(\Gamma) \rightarrow \mathcal{T}(\Gamma')$:

Compatibility w. substitutions. ...

Generic predicate. There exists $\Omega \in \mathbf{Set}$ and $\text{holds} \in \mathcal{T}(\Omega)$, s.t. ...

Standard example (with $\mathcal{H} \in \mathbf{HpA}$)

$$\mathcal{T}(I) = \mathcal{H}^I \qquad \mathcal{T}(s : I \rightarrow J) = \lambda(h : \mathcal{H}^J). h \circ s$$

Quantifiers. arbitrary meets/joins

Compatibility. yes.

Generic predicate. $\Omega \triangleq \mathcal{H}$ and $\text{holds} \triangleq \lambda x.x$

Models and triposes

A *tripos* is a functor $\mathcal{T} : \mathbf{Set}^{op} \rightarrow \mathbf{HpA}$ s.t.:

Quantifiers. Any s^* has left/right adjoints $\coprod_s / \prod_s : \mathcal{T}(\Gamma) \rightarrow \mathcal{T}(\Gamma')$:

Compatibility w. substitutions. ...

Generic predicate. There exists $\Omega \in \mathbf{Set}$ and $\text{holds} \in \mathcal{T}(\Omega)$, s.t. ...

So far so good but:

- It is a heavy structure
- What about **terms** and the realizability **interpretation**?

Recently



Models and triposes

A *tripos* is a functor $\mathcal{T} : \mathbf{Set}^{op} \rightarrow \mathbf{HpA}$ s.t.:

Quantifiers. Any s^* has left/right adjoints $\coprod_s / \prod_s : \mathcal{T}(\Gamma) \rightarrow \mathcal{T}(\Gamma')$:

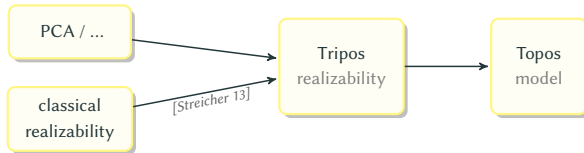
Compatibility w. substitutions. ...

Generic predicate. There exists $\Omega \in \mathbf{Set}$ and $\text{holds} \in \mathcal{T}(\Omega)$, s.t. ...

So far so good but:

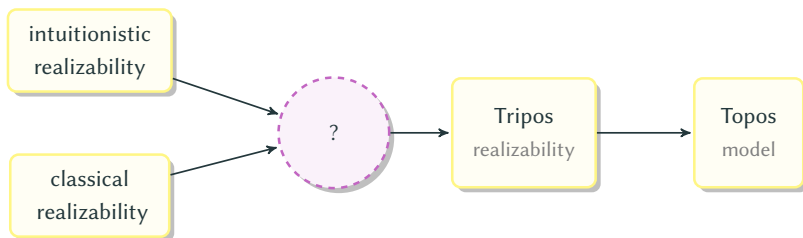
- It is a heavy structure
- What about **terms** and the realizability **interpretation**?

Recently



Effectful realizability, from an algebraic perspective

The quest for a better world



Algebrization of realizability models

Key observations : truth values live in $\mathcal{P}(\mathbb{A})$ which

- ❶ is a complete lattice,
- ❷ can be endowed with Kleene arrow :

$$\phi \Rightarrow \psi = \{a \in \mathbb{A} \mid \forall b \in \phi. ab \downarrow \wedge ab \in \psi\}$$

$\nrightarrow$ for classical realizability, this applies to $\mathcal{P}(\Pi)$.

Algebrization of realizability models

Key observations : truth values live in $\mathcal{P}(\mathbb{A})$ which

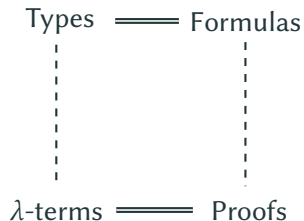
- ❶ is a complete lattice,
- ❷ can be endowed with Kleene arrow :

$$\phi \Rightarrow \psi = \{a \in \mathbb{A} \mid \forall b \in \phi. ab \downarrow \wedge ab \in \psi\}$$

$\nrightarrow$ for classical realizability, this applies to $\mathcal{P}(\Pi)$.

Implicative structures

Implicative algebras: a new (...)
Alexandre Miquel [2018]

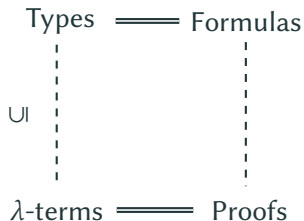


In particular, $a \preccurlyeq b$ reads:

- a is a *subtype* of b
- a is a *realizer* of b
- the realizer a is *more defined* than b

Implicative structures

Implicative algebras: a new (...)
Alexandre Miquel [2018]

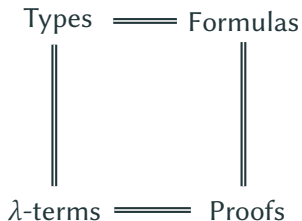


In particular, $a \preccurlyeq b$ reads:

- a is a *subtype* of b
- a is a *realizer* of b
- the realizer a is *more defined* than b

Implicative structures

Implicative algebras: a new (...)
Alexandre Miquel [2018]



In particular, $a \preccurlyeq b$ reads:

- a is a *subtype* of b
- a is a *realizer* of b
- the realizer a is *more defined* than b

Implicative algebras

An *implicative algebra* is a complete meet-semilattice $(\mathcal{A}, \preceq, \rightarrow)$ s.t.:

- if $a_0 \preceq a$ and $b \preceq b_0$ then $(a \rightarrow b) \preceq (a_0 \rightarrow b_0)$ (Variance)
- $\bigwedge_{b \in B} (a \rightarrow b) = a \rightarrow \bigwedge_{b \in B} b$ (Distributivity)

together with a **separator** $S \in \mathcal{A}$ satisfying axioms.

Main results:

- System F's terms and types can be interpreted in an adequate way:

$$\text{If } \vdash t : A \text{ then } t^{\mathcal{A}} \preceq A^{\mathcal{A}}.$$

- Any implicative algebra induces a tripos :

$$\mathcal{T} : \begin{cases} \mathbf{Set}^{op} & \rightarrow \mathbf{HA} \\ I & \mapsto \mathcal{A}^I / S[I] \end{cases}$$

Great, but...

- implicative algebras : a (too?) minimal structure

$\text{realizers} \in \mathcal{A} \ni \text{formulas}$

- tripos: does not account for realizers

propositional logic	$\leftrightarrow$	Heyting prealgebra
quantifiers	$\leftrightarrow$	adjoints to substitutions
higher-order	$\leftrightarrow$	generic predicate

- lack of a smooth integration for

states / non-determinism / dependent types / ...

In short

Two general structures, none account for realizers!

Great, but...

- implicative algebras : a (too?) minimal structure

$\text{realizers} \in \mathcal{A} \ni \text{formulas}$

- tripos: does not account for realizers

propositional logic	$\leftrightarrow$	Heyting prealgebra
quantifiers	$\leftrightarrow$	adjoints to substitutions
higher-order	$\leftrightarrow$	generic predicate

- lack of a smooth integration for

states / non-determinism / dependent types / ...

In short

Two general structures, none account for realizers!

Great, but...

- implicative algebras : a (too?) minimal structure

$\text{realizers} \in \mathcal{A} \ni \text{formulas}$

- tripos: does not account for realizers

propositional logic	$\leftrightarrow$	Heyting prealgebra
quantifiers	$\leftrightarrow$	adjoints to substitutions
higher-order	$\leftrightarrow$	generic predicate

- lack of a smooth integration for

states / non-determinism / dependent types / ...

In short

Two general structures, none account for realizers!

Great, but...

- implicative algebras : a (too?) minimal structure

$\text{realizers} \in \mathcal{A} \ni \text{formulas}$

- tripos: does not account for realizers

propositional logic	$\leftrightarrow$	Heyting prealgebra
quantifiers	$\leftrightarrow$	adjoints to substitutions
higher-order	$\leftrightarrow$	generic predicate

- lack of a smooth integration for

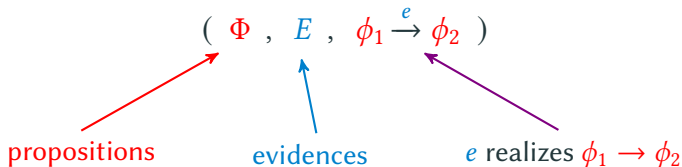
states / non-determinism / dependent types / ...

In short

Two general structures, none account for realizers!

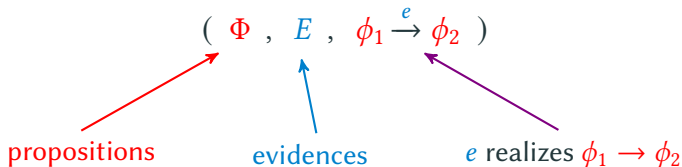
Evidenced Frames

Evidenced Frame: A Unifying Framework for Realizability Models



Intuitively: a “specification” of the minimal structure

Evidenced Frame: A Unifying Framework for Realizability Models



Intuitively: a “specification” of the minimal structure

Evidenced Frame $(\Phi, E, \cdot \xrightarrow{\cdot} \cdot)$

Reflexivity. $e_{\text{id}} \in E$ s.t.:

- $\phi \xrightarrow{e_{\text{id}}} \phi$

Transitivity. $;\in E \times E \rightarrow E$ s.t.:

- $\phi_1 \xrightarrow{e} \phi_2 \wedge \phi_2 \xrightarrow{e'} \phi_3 \implies \phi_1 \xrightarrow{e;e'} \phi_3$

Top. $\top \in \Phi$ and $e_{\top} \in E$ s.t.:

- $\phi \xrightarrow{e_{\top}} \top$

Conjunction. $\wedge \in \Phi \times \Phi \rightarrow \Phi$, $\langle \cdot, \cdot \rangle \in E \times E \rightarrow E$, and $e_{\text{fst}}, e_{\text{snd}} \in E$ s.t.:

- $\phi_1 \wedge \phi_2 \xrightarrow{e_{\text{fst}}} \phi_1$
- $\phi \xrightarrow{e_1} \phi_1 \wedge \phi \xrightarrow{e_2} \phi_2 \implies \phi \xrightarrow{\langle e_1, e_2 \rangle} \phi_1 \wedge \phi_2$
- $\phi_1 \wedge \phi_2 \xrightarrow{e_{\text{snd}}} \phi_2$
- $\phi_1 \wedge \phi_2 \xrightarrow{e_{\text{snd}}} \phi_2$

Universal implication. $\supset \in \Phi \times \mathcal{P}(\Phi) \rightarrow \Phi$, $\lambda \in E \rightarrow E$, and $e_{\text{eval}} \in E$:

- $(\forall \phi \in \vec{\phi}. \phi_1 \wedge \phi_2 \xrightarrow{e} \phi) \implies \phi_1 \xrightarrow{\lambda e} \phi_2 \supset \vec{\phi}$
- $\forall \phi \in \vec{\phi}. [(\phi_1 \supset \vec{\phi}) \wedge \phi_1 \xrightarrow{e_{\text{eval}}} \phi]$

Evidenced Frame $(\Phi, E, \cdot \rightarrow \cdot)$

Reflexivity. $e_{\text{id}} \in E$ s.t.:

- $\phi \xrightarrow{e_{\text{id}}} \phi$

Transitivity. $;\in E \times E \rightarrow E$ s.t.:

- $\phi_1 \xrightarrow{e} \phi_2 \wedge \phi_2 \xrightarrow{e'} \phi_3 \implies \phi_1 \xrightarrow{e; e'} \phi_3$

Top. $\top \in \Phi$ and $e_{\top} \in E$ s.t.:

- $\phi \xrightarrow{e_{\top}} \top$

Conjunction. $\wedge \in \Phi \times \Phi \rightarrow \Phi$, $\langle \cdot, \cdot \rangle \in E \times E \rightarrow E$, and $e_{\text{fst}}, e_{\text{snd}} \in E$ s.t.:

- $\phi_1 \wedge \phi_2 \xrightarrow{e_{\text{fst}}} \phi_1$
- $\phi \xrightarrow{e_1} \phi_1 \wedge \phi \xrightarrow{e_2} \phi_2 \implies \phi \xrightarrow{\langle e_1, e_2 \rangle} \phi_1 \wedge \phi_2$
- $\phi_1 \wedge \phi_2 \xrightarrow{e_{\text{snd}}} \phi_2$

Universal implication. $\supset \in \Phi \times \mathcal{P}(\Phi) \rightarrow \Phi$, $\lambda \in E \rightarrow E$, and $e_{\text{eval}} \in E$:

- $(\forall \phi \in \vec{\phi}. \phi_1 \wedge \phi_2 \xrightarrow{e} \phi) \implies \phi_1 \xrightarrow{\lambda e} \phi_2 \supset \vec{\phi}$
- $\forall \phi \in \vec{\phi}. [(\phi_1 \supset \vec{\phi}) \wedge \phi_1 \xrightarrow{e_{\text{eval}}} \phi]$

Goal #1: An intermediate structure

PCA to Evidenced Frame



- A set of “codes”
- *partial* application $c_1 \cdot c_2$
- functional completeness

The triple $(\mathcal{C}, \cdot, \mathcal{F}) \rightarrow \mathcal{F}$, where

- a *code* is an $c \in \mathcal{C} = \mathcal{P}(A)$ is defined by the set of realizers

- an *expression* in $\mathcal{F}$ is a code
- $c \cdot d \in \mathcal{F}$ for all $c, d \in \mathcal{C}$

where $\cdot$ is *partial* application

and $\mathcal{F} = \{c \in \mathcal{P}(A) \mid \exists x, y \rightarrow x \cdot y \in c\}$

The *realizers* and *denotations* are defined as usual in partiality theory

PCA to Evidenced Frame



- $\mathbb{A}$ set of “codes”
- *partial* application $c_1 \cdot c_2$
- functional completeness

The triple $(\mathcal{P}(\mathbb{A}), \mathbb{A}, \cdot \overset{e}{\rightarrow} \cdot)$, where:

- a **proposition** in $\Phi = \mathcal{P}(\mathbb{A})$ is defined by its set of realizers
- an **evidence** in $E = \mathbb{A}$ is a code
- $\phi_1 \overset{e}{\rightarrow} \phi_2$ if for all $e_1 \in \phi_1$:
 - $e \cdot e_1$ terminates
 - $e \cdot e_1 \downarrow e_2 \Rightarrow e_2 \in \phi_2$

$\forall$ connectives and their evidences are defined as usual in realizability models

PCA to Evidenced Frame



- $\mathbb{A}$ set of “codes”
- *partial* application $c_1 \cdot c_2$
- functional completeness

The triple $(\mathcal{P}(\mathbb{A}), \mathbb{A}, \cdot \xrightarrow{\cdot})$, where:

- a **proposition** in $\Phi = \mathcal{P}(\mathbb{A})$ is defined by its set of realizers
- an **evidence** in $E = \mathbb{A}$ is a code
- $\phi_1 \xrightarrow{e} \phi_2$ if for all $e_1 \in \phi_1$:
 - $e \cdot e_1$ terminates
 - $e \cdot e_1 \downarrow e_2 \Rightarrow e_2 \in \phi_2$

$\leadsto$ connectives and their evidences are defined as usual in realizability models

Evidenced Frame to Tripos



UFam construction

Given $\mathcal{EF} = (\Phi, E, \cdot \xrightarrow{\cdot} \cdot)$, the structure $\text{UFam}(\mathcal{EF})$ is defined by:

Predicates. $\Gamma \in \mathbf{Set}$ is mapped to $\Phi^\Gamma \in \mathbf{HpA}$

- $\phi \preceq \phi' \triangleq \exists e. \forall \gamma. \phi(\gamma) \xrightarrow{e} \phi'(\gamma)$

$\hookrightarrow$ *uniform entailment*

- Heyting prealgebra: pointwise via Φ

Substitution. $s : \Gamma \rightarrow \Gamma'$ is mapped to $\mathcal{T}(s) = \lambda h. h \circ s$

$\hookrightarrow$ *as usual*

Quantifiers. $\prod_u \in \Phi^I \rightarrow \Phi^J \triangleq \lambda \phi. \lambda j. \prod_{i \in u^{-1}(j)} \phi(i)$

$\hookrightarrow$ *as usual*

$$\coprod_u \in \Phi^I \rightarrow \Phi^J \triangleq \lambda \phi. \lambda j. \coprod_{i \in u^{-1}(j)} \phi(i).$$

Generic predicate. $\Omega \triangleq \Phi$, $\text{holds} \triangleq \text{id}_\Omega$, and $\chi_\phi \triangleq \phi$.

$\hookrightarrow$ *as usual*

From Implicative Algebras to Evidenced frames

Any implicative algebra $(\mathcal{A}, \preceq, \rightarrow, S)$ induces an evidenced frame

$$\text{UEF}(\mathcal{A}) \triangleq (\underbrace{\mathcal{A}}_{\text{prop.}}, \underbrace{S}_{\text{evidences}}, \cdot \dot{\rightarrow} \cdot) \quad \text{where} \quad a \xrightarrow{e} b \triangleq e \preceq a \rightarrow b$$

$\hookrightarrow$ **Proof.** Connectives and quantifiers from the internal logic of $\mathcal{A}$ / evidences via the expected λ -terms.

Remark

- blurs the distinction btw. evidences & propositions
- $\text{UEF}(\mathcal{A})$ is consistent if and only if $\mathcal{A}$ is.
- the implicative tripes $\mathcal{T}^{\mathcal{A}}$ and $\text{UFam}(\text{UEF}(\mathcal{A}))$ are equivalent.

From Implicative Algebras to Evidenced frames

Any implicative algebra $(\mathcal{A}, \preceq, \rightarrow, S)$ induces an evidenced frame

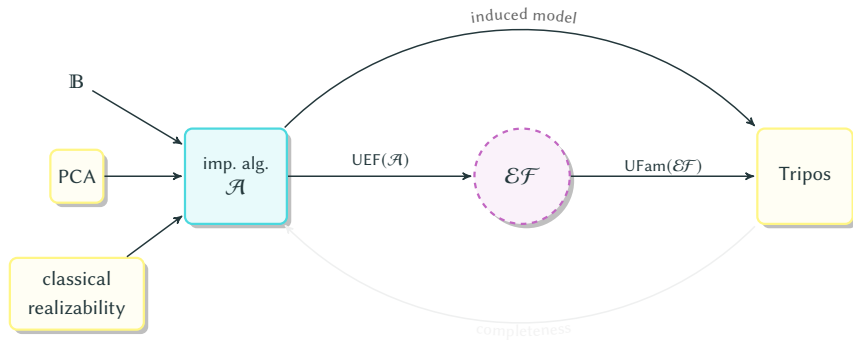
$$\text{UEF}(\mathcal{A}) \triangleq (\underbrace{\mathcal{A}}_{\text{prop.}}, \underbrace{S}_{\text{evidences}}, \cdot \dot{\rightarrow} \cdot) \quad \text{where} \quad a \xrightarrow{e} b \triangleq e \preceq a \rightarrow b$$

$\hookrightarrow$ **Proof.** Connectives and quantifiers from the internal logic of $\mathcal{A}$ / evidences via the expected λ -terms.

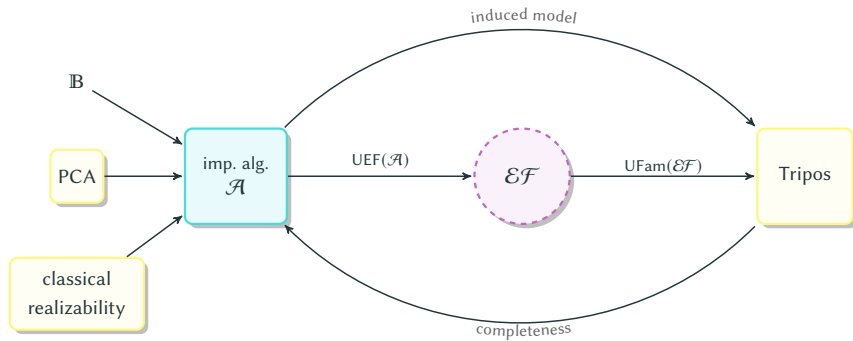
Remark

- blurs the distinction btw. evidences & propositions
- $\text{UEF}(\mathcal{A})$ is consistent if and only if $\mathcal{A}$ is.
- the implicative tripes $\mathcal{T}^{\mathcal{A}}$ and $\text{UFam}(\text{UEF}(\mathcal{A}))$ are equivalent.

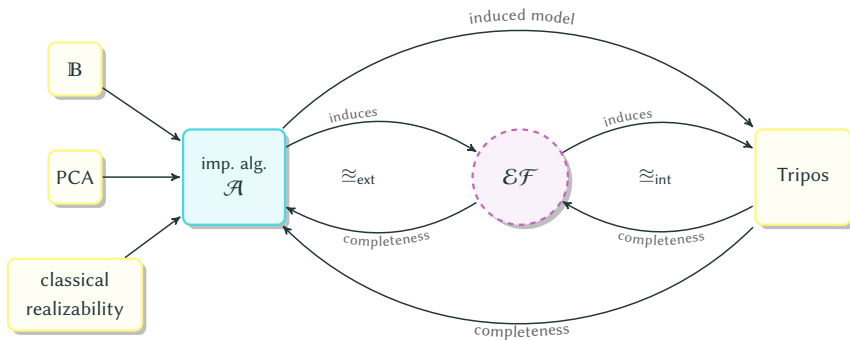
From Implicative Algebras to Evidenced frames



From Implicative Algebras to Evidenced frames



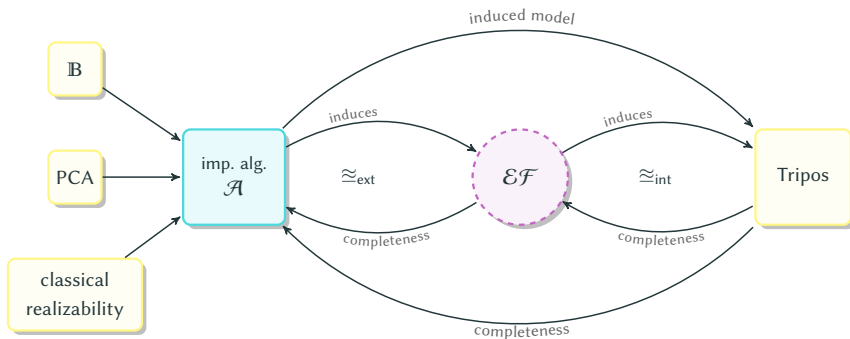
Final picture



Slogan

Tripos = evidenced frame that has forgotten its evidence.

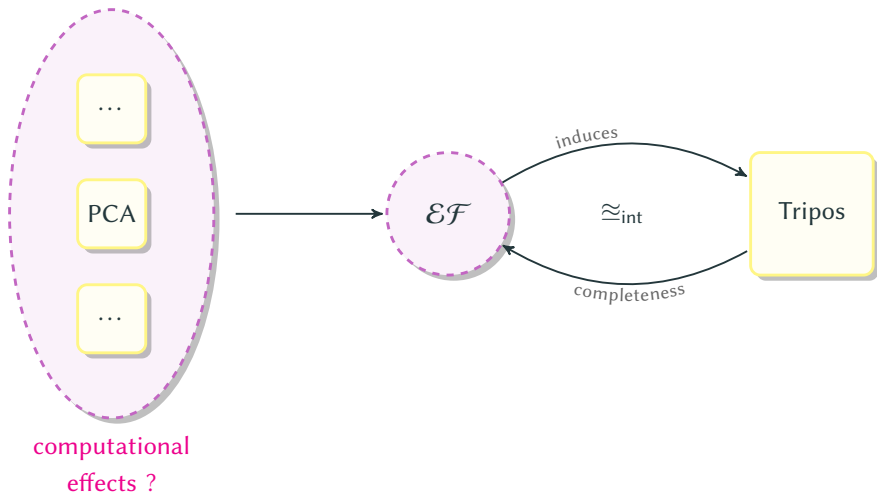
Final picture



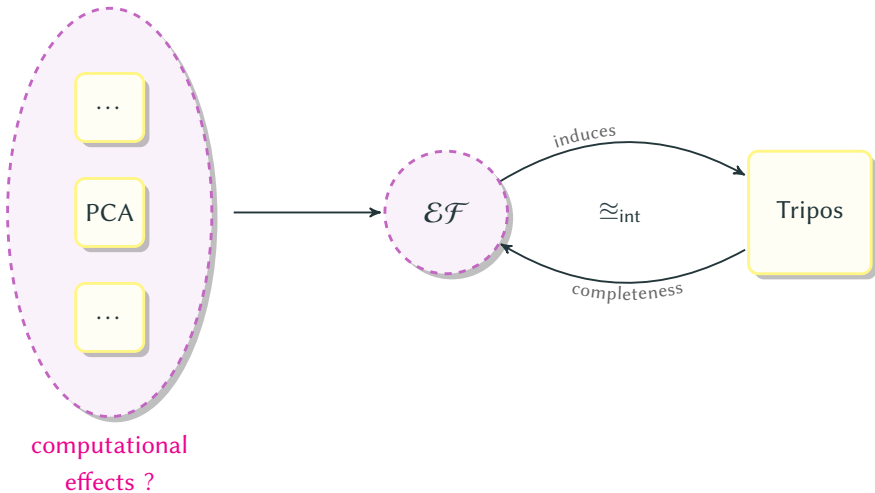
Slogan

Tripos = evidenced frame that has forgotten its evidence.

This Talk: Goal #2



This Talk: Goal #2



Goal #2

Smooth integration of useful computational effects

Monadic Combinatory Algebras

Reminder : PCA

Definition

A **Partial Applicative Structure** $(\mathbb{A}, \cdot)$ is given by:

a set $\mathbb{A}$ of “codes” + a partial “application” $\cdot : \mathbb{A} \times \mathbb{A} \rightarrow \mathbb{A}$

Notations:

- $c_f \cdot c_a \downarrow c_r$: c_r is the (successful) result of the application $c_f \cdot c_a$
- $c_f \cdot c_a \downarrow$: $\exists c_r \in \mathbb{A}, c_f \cdot c_a \downarrow c_r$.

Definition

A **Partial Combinatory Algebra** is a PAS that is “functionally complete”.

$\Uparrow$ for every expression e with n free variables, there is a code $\langle \lambda^n . e \rangle \in \mathbb{A}$ s.t.:

$$\langle \lambda^{n+1} . e \rangle \cdot c_a \downarrow \langle \lambda^n . e \{c_a\} \rangle \qquad \langle \lambda^0 . e \rangle \cdot c_a \downarrow c_r \iff e\{c_a\} \downarrow c_r$$

Monads

Definition

A **monad** M over a category $\mathbb{C}$ is a functor $M : \mathbb{C} \rightarrow \mathbb{C}$, with:

- $\eta : 1 \Rightarrow M$ (*unit*)
- $\mu : MM \Rightarrow M$ (*multiplication*)

satisfying

$$\begin{array}{ccc} M & \xrightarrow{\eta M} & MM & \xleftarrow{M\eta} & M \\ & \searrow & \downarrow \mu & \nearrow & \\ & & M & & \end{array} \qquad \begin{array}{ccc} MMM & \xrightarrow{M\mu} & MM \\ \mu M \downarrow & & \downarrow \mu \\ MM & \xrightarrow{\mu} & M \end{array}$$

Computationally :

- $f : A \rightarrow MB$: effectful program taking a value in A and returning a computation in B
- $[a] : MA$ stands for $\eta(a)$ (*return*)
- $\text{let } x \Leftarrow m \text{ in } n : MB$ with $\lambda x.n : A \rightarrow MB$ and $m : MA$ (*bind*)

Monadic Applicative Structure

In this work: M is a **Set** monad

Definition

A **Monadic Applicative Structure** (MAS) over M is a set of “codes” $\mathbb{A}$ with a (Kleisli) application $: (-) \cdot (-) : \mathbb{A} \times \mathbb{A} \rightarrow M\mathbb{A}$.

$\leadsto$ PAS where partiality is replaced by any monad.

Intuitively:

- $(-) \cdot (-)$ takes two codes and returns a ‘*computation*’ of a code.
- codes = encoding of programs or inputs of programs.
- $c_f \cdot c_a$: computation executed by running c_f on c_a

Expressions

$$e ::= i \in \mathbb{N} \mid c \in \mathbb{A} \mid e \bullet e$$

$$E_n(\mathbb{A}) ::= \{e \mid \text{all is in } e \text{ are } < n\}$$

Monadic Applicative Structure

In this work: M is a **Set** monad

Definition

A **Monadic Applicative Structure** (MAS) over M is a set of “codes” $\mathbb{A}$ with a (Kleisli) application $: (-) \cdot (-) : \mathbb{A} \times \mathbb{A} \rightarrow M\mathbb{A}$.

$\hookrightarrow$ PAS where partiality is replaced by any monad.

Intuitively:

- $(-) \cdot (-)$ takes two codes and returns a ‘*computation*’ of a code.
- **codes** = encoding of programs or inputs of programs.
- $c_f \cdot c_a$: computation executed by running c_f on c_a

Expressions

$$e ::= i \in \mathbb{N} \mid c \in \mathbb{A} \mid e \bullet e$$

$$E_n(\mathbb{A}) ::= \{e \mid \text{all is in } e \text{ are } < n\}$$

Monadic Applicative Structure

In this work: M is a **Set** monad

Definition

A **Monadic Applicative Structure** (MAS) over M is a set of “codes” $\mathbb{A}$ with a (Kleisli) application $: (-) \cdot (-) : \mathbb{A} \times \mathbb{A} \rightarrow M\mathbb{A}$.

$\hookrightarrow$ PAS where partiality is replaced by any monad.

Intuitively:

- $(-) \cdot (-)$ takes two codes and returns a ‘*computation*’ of a code.
- **codes** = encoding of programs or inputs of programs.
- $c_f \cdot c_a$: computation executed by running c_f on c_a

Expressions

$$e ::= i \in \mathbb{N} \mid c \in \mathbb{A} \mid e \bullet e$$

$$E_n(\mathbb{A}) ::= \{e \mid \text{all is in } e \text{ are } < n\}$$

Evaluation

We follow the PCA historical presentation :

CbV Evaluation

Evaluation v is defined by induction on $E_0(\mathbb{A})$ as follows:

$$v(c) := [c] \qquad v(e_f \bullet e_a) := \text{let } c_f \Leftarrow v(e_f) \text{ in let } c_a \Leftarrow v(e_a) \text{ in } c_f \cdot c_a$$

$\hookrightarrow$ *evaluation is reflected by an equality in $\mathbb{A}$*

$\hookrightarrow$ *having an algebraic counterpart of evaluation would require switching to an ordered setting*

Monadic Combinatory Algebra

MCA

A **Monadic Combinatory Algebra** is a MAS $\mathbb{A}$ s.t. for any expression $e \in E_{n+1}(\mathbb{A})$ there is a code $\langle \lambda^n.e \rangle \in \mathbb{A}$ satisfying the following laws:

$$\begin{aligned}\langle \lambda^{n+1}.e \rangle \cdot c &= \eta(\langle \lambda^n.e\{c\} \rangle) & (\forall e \in E_{n+2}(\mathbb{A})) \\ \langle \lambda^0.e \rangle \cdot c &= v(e\{c\}) & (\forall e \in E_1(\mathbb{A}))\end{aligned}$$

$\leadsto$ this can also be expressed in categorical terms, see the paper

Example -

The **sub-singleton monad**:

- $MA = \{S \subseteq A \mid \forall x_1, x_2 \in S. x_1 = x_2\}$
- $\eta_A(x) = \{x\}$
- $\mu_A(m) = \bigcup_{X \in m} X$

Riddle : what is the encoded effect ?

PCA is a special case of an MCA.

Example - PCA

The **sub-singleton monad**:

- $MA = \{S \subseteq A \mid \forall x_1, x_2 \in S. x_1 = x_2\}$
- $\eta_A(x) = \{x\}$
- $\mu_A(m) = \bigcup_{X \in m} X$

Riddle : what is the encoded effect ?

PCA is a special case of an MCA.

Example -

- $MA = (A \rightarrow R) \rightarrow R$
- $\eta_A(x) = \lambda k.k\ x$
- $\mu_A(m) = \lambda k.m\ (\lambda g.g\ (k))$

Riddle : what is the encoded effect ?

Such an MCA allows for e_{cc} and e_{K_u} which save/restore continuations :

$$(e_{cc} \cdot c_a)(u) = (c_a \cdot e_{K_u})(u) \qquad (e_{K_u} \cdot c_a)(u') = u(c_a)$$

By defunctionalization, we obtain an abstract stack machine with states:

- 1 $e \triangleright \pi$ *eval state*
- 2 $c \triangleleft \pi$ *apply state*
- 3 c *final state*

Operational semantics:

$e_f \cdot e_a \triangleright \pi \hookrightarrow e_f \triangleright t(e_a) \cdot \pi$	$c_f \triangleleft t(e_a) \cdot \pi \hookrightarrow e_a \triangleright v(c_f) \cdot \pi$
$c \triangleright \pi \hookrightarrow c \triangleleft \pi$	$c_a \triangleleft v(\langle \lambda^0.e \rangle) \cdot \pi \hookrightarrow e\{c_a\} \triangleright \pi$
$c \triangleleft \emptyset \hookrightarrow c$	$c_a \triangleleft v(\langle \lambda^{n+1}.e \rangle) \cdot \pi \hookrightarrow \langle \lambda^n.e\{c_a\} \rangle \triangleleft \pi$

Example - CPSCA

$$\bullet MA = (A \rightarrow R) \rightarrow R \quad \bullet \eta_A(x) = \lambda k.k x \quad \bullet \mu_A(m) = \lambda k.m (\lambda g.g(k))$$

Such an MCA allows for e_{cc} and e_{K_u} which save/restore continuations :

$$(e_{cc} \cdot c_a)(u) = (c_a \cdot e_{K_u})(u) \quad (e_{K_u} \cdot c_a)(u') = u(c_a)$$

By defunctionalization, we obtain an abstract stack machine with states:

$$\textcircled{1} \quad e \triangleright \pi \text{ eval state} \quad \textcircled{2} \quad c \triangleleft \pi \text{ apply state} \quad \textcircled{3} \quad c \text{ final state}$$

Operational semantics:

$$\begin{array}{lcl} e_f \cdot e_a \triangleright \pi & \hookrightarrow & e_f \triangleright t(e_a) \cdot \pi \\ c \triangleright \pi & \hookrightarrow & c \triangleleft \pi \\ c \triangleleft \emptyset & \hookrightarrow & c \end{array} \quad \begin{array}{lcl} c_f \triangleleft t(e_a) \cdot \pi & \hookrightarrow & e_a \triangleright v(c_f) \cdot \pi \\ c_a \triangleleft v(\langle \lambda^0.e \rangle) \cdot \pi & \hookrightarrow & e\{c_a\} \triangleright \pi \\ c_a \triangleleft v(\langle \lambda^{n+1}.e \rangle) \cdot \pi & \hookrightarrow & \langle \lambda^n.e\{c_a\} \rangle \triangleleft \pi \end{array}$$

Example - CPSCA

$$\bullet MA = (A \rightarrow R) \rightarrow R \quad \bullet \eta_A(x) = \lambda k.k x \quad \bullet \mu_A(m) = \lambda k.m(\lambda g.g(k))$$

Such an MCA allows for e_{cc} and e_{K_u} which save/restore continuations :

$$(e_{cc} \cdot c_a)(u) = (c_a \cdot e_{K_u})(u) \quad (e_{K_u} \cdot c_a)(u') = u(c_a)$$

By defunctionalization, we obtain an abstract stack machine with states:

$$\textcircled{1} \quad e \triangleright \pi \text{ eval state} \quad \textcircled{2} \quad c \blacktriangleleft \pi \text{ apply state} \quad \textcircled{3} \quad c \text{ final state}$$

Operational semantics:

$$\begin{array}{lcl} e_f \cdot e_a \triangleright \pi & \hookrightarrow & e_f \triangleright t(e_a) \cdot \pi \\ c \triangleright \pi & \hookrightarrow & c \blacktriangleleft \pi \\ c \blacktriangleleft \emptyset & \hookrightarrow & c \end{array} \quad \left| \quad \begin{array}{lcl} c_f \blacktriangleleft t(e_a) \cdot \pi & \hookrightarrow & e_a \triangleright v(c_f) \cdot \pi \\ c_a \blacktriangleleft v(\langle \lambda^0.e \rangle) \cdot \pi & \hookrightarrow & e\{c_a\} \triangleright \pi \\ c_a \blacktriangleleft v(\langle \lambda^{n+1}.e \rangle) \cdot \pi & \hookrightarrow & \langle \lambda^n.e\{c_a\} \rangle \blacktriangleleft \pi \end{array} \right.$$

More Examples

Comb. Alg.	Monad MA	return $\eta_A(x)$	bind $\mu_A(m)$
<i>Partial</i>	$\{X \subseteq A \mid \forall x, y \in X. x = y\}$	$\{x\}$	$\bigcup_{X \in m} X$
<i>Relational</i>	$\mathcal{P}(A)$	$\{x\}$	$\bigcup_{X \in m} X$
<i>Stateful</i>	$\Sigma \rightarrow \mathcal{P}(\Sigma \times A)$	$\lambda\sigma. \{(\sigma, x)\}$	$\lambda\sigma. \bigcup_{(\sigma', f) \in m(\sigma)} f(\sigma')$
<i>CPS</i>	$(A \rightarrow R) \rightarrow R$	$\lambda k. k(x)$	$\lambda k. m(\lambda g. g(k))$
<i>Parameterized</i>	$\mathbf{P} \rightarrow \{X \subseteq A \mid \forall x, y \in X. x = y\}$	$\lambda p. \{x\}$	$\lambda p. \bigcup \{g(p) \mid g \in m(p)\}$

Realizability models

The good old recipe :

- $\mathcal{P}(\mathbb{A})$ -valued predicates

$$\phi : X \rightarrow \mathcal{P}(\mathbb{A}).$$

- Kleene arrow

$$(\phi \Rightarrow \psi)(x) = \{a \in \mathbb{A} \mid \forall b \in \phi(x). ab \downarrow \wedge ab \in \psi(x)\}.$$

Problem

We need to lift $\psi \in \mathcal{P}(\mathbb{A})$ to an adequate set of computations $\bar{\psi} \in \mathcal{P}(MA)$

Realizability models

The good old recipe :

- $\mathcal{P}(\mathbb{A})$ -valued predicates

$$\phi : X \rightarrow \mathcal{P}(\mathbb{A}).$$

- Kleene arrow

$$(\phi \Rightarrow \psi)(x) = \{a \in \mathbb{A} \mid \forall b \in \phi(x). ab \downarrow \wedge ab \in \psi(x)\}.$$

Problem

We need to lift $\psi \in \mathcal{P}(\mathbb{A})$ to an adequate set of computations $\bar{\psi} \in \mathcal{P}(MA)$

Realizability models

The good old recipe :

- $\mathcal{P}(\mathbb{A})$ -valued predicates

$$\phi : X \rightarrow \mathcal{P}(\mathbb{A}).$$

- Kleene arrow **X**

$$(\phi \Rightarrow \psi)(x) = \{a \in \mathbb{A} \mid \forall b \in \phi(x). ab \downarrow \wedge \underbrace{ab}_{\in M\mathbb{A}} \in \underbrace{\psi(x)}_{\subseteq \mathbb{A}}\}.$$

Problem

We need to lift $\psi \in \mathcal{P}(\mathbb{A})$ to an adequate set of computations $\bar{\psi} \in \mathcal{P}(M\mathbb{A})$

M-modality

An M -modality over $(\Omega, \preceq)$ a complete **HpA** is a natural transformation:

$$\Diamond_X : MX \rightarrow (X \rightarrow \Omega) \rightarrow \Omega$$

s.t., writing $\langle x \leftarrow m \rangle \phi(x) \triangleq \Diamond(m)(\phi)$ (“after m , ϕ holds”), we have:

❶ After / return :

$$\phi(a) \preceq \langle x \leftarrow [a] \rangle \phi(x)$$

❷ After / Bind

$$\langle x \leftarrow m \rangle \langle y \leftarrow f(x) \rangle \phi(y) \preceq \langle y \leftarrow \text{let } x \leftarrow m \text{ in } f(x) \rangle \phi(y)$$

❸ Internal Monotonicity.

$$\bigwedge_c (\phi_1(c) \sqcap \phi_2(c)) \preceq \langle x \leftarrow m \rangle \phi_1(x) \sqcap \langle x \leftarrow m \rangle \phi_2(x)$$

M-modality

An M -modality over $(\Omega, \preceq)$ a complete **HpA** is a natural transformation:

$$\Diamond_X : \underbrace{MX}_{\text{computation}} \rightarrow \underbrace{(X \rightarrow \Omega)}_{\text{predicate}} \rightarrow \underbrace{\Omega}_{\text{truth value}}$$

s.t., writing $\langle x \leftarrow m \rangle \phi(x) \triangleq \Diamond(m)(\phi)$ (“after m , ϕ holds”), we have:

❶ After / return :

$$\phi(a) \preceq \langle x \leftarrow [a] \rangle \phi(x)$$

❷ After / Bind

$$\langle x \leftarrow m \rangle \langle y \leftarrow f(x) \rangle \phi(y) \preceq \langle y \leftarrow \text{let } x \leftarrow m \text{ in } f(x) \rangle \phi(y)$$

❸ Internal Monotonicity.

$$\bigwedge_c (\phi_1(c) \sqsupset \phi_2(c)) \preceq \langle x \leftarrow m \rangle \phi_1(x) \sqsupset \langle x \leftarrow m \rangle \phi_2(x)$$

M-modality

An M -modality over $(\Omega, \preceq)$ a complete **HpA** is a natural transformation:

$$\diamond_X : \underbrace{MX}_{\text{computation}} \rightarrow \underbrace{(X \rightarrow \Omega)}_{\text{predicate}} \rightarrow \underbrace{\Omega}_{\text{truth value}}$$

s.t., writing $\langle x \leftarrow m \rangle \phi(x) \triangleq \diamond(m)(\phi)$ (“after m , ϕ holds”), we have:

❶ After / return :

$$\phi(a) \preceq \langle x \leftarrow [a] \rangle \phi(x)$$

❷ After / Bind

$$\langle x \leftarrow m \rangle \langle y \leftarrow f(x) \rangle \phi(y) \preceq \langle y \leftarrow \text{let } x \leftarrow m \text{ in } f(x) \rangle \phi(y)$$

❸ Internal Monotonicity.

$$\bigwedge_c (\phi_1(c) \sqsupset \phi_2(c)) \preceq \langle x \leftarrow m \rangle \phi_1(x) \sqsupset \langle x \leftarrow m \rangle \phi_2(x)$$

M-modality

An M -modality over $(\Omega, \preceq)$ a complete **HpA** is a natural transformation:

$$\diamond_X : \underbrace{MX}_{\text{computation}} \rightarrow \underbrace{(X \rightarrow \Omega)}_{\text{predicate}} \rightarrow \underbrace{\Omega}_{\text{truth value}}$$

s.t., writing $\langle x \leftarrow m \rangle \phi(x) \triangleq \diamond(m)(\phi)$ (“after m , ϕ holds”), we have:

❶ After / return :

$$\phi(a) \preceq \langle x \leftarrow [a] \rangle \phi(x)$$

❷ After / Bind

$$\langle x \leftarrow m \rangle \langle y \leftarrow f(x) \rangle \phi(y) \preceq \langle y \leftarrow \text{let } x \leftarrow m \text{ in } f(x) \rangle \phi(y)$$

❸ Internal Monotonicity.

$$\bigcap_c (\phi_1(c) \sqsupset \phi_2(c)) \preceq \langle x \leftarrow m \rangle \phi_1(x) \sqsupset \langle x \leftarrow m \rangle \phi_2(x)$$

All good ?

Consider the **PCA** example, that is:

$$MA = \{S \subseteq A \mid \forall x_1, x_2 \in S. x_1 = x_2\}$$

What about the following M-modality?

$$\langle x \leftarrow m \rangle \phi(x) \triangleq \bigwedge_{x \in m} \phi(x)$$

Consider :

- $c_{\perp} \triangleq \langle \lambda x. \delta \delta \rangle$, that yields no value when applied,
- the predicates $\top = x \mapsto 1$ and $\perp = x \mapsto 0$

Then we get:

$$\begin{aligned} \top \xrightarrow{c_{\perp}} \perp &\Leftrightarrow \forall c \in A. 1 \not\preceq \langle x \leftarrow c_{\perp} \cdot c \rangle 0 \\ &\Leftrightarrow \forall c \in A. 1 \not\preceq \bigwedge_{x \in v(c_{\perp} \cdot c)} .0 \Leftrightarrow 1 \not\preceq 1 \end{aligned}$$



All good ?

Consider the **PCA** example, that is:

$$MA = \{S \subseteq A \mid \forall x_1, x_2 \in S. x_1 = x_2\}$$

What about the following M-modality?

$$\langle x \leftarrow m \rangle \phi(x) \triangleq \bigwedge_{x \in m} \phi(x)$$

Consider :

- $c_{\circ} \triangleq \langle \lambda x. \delta \delta \rangle$, that yields no value when applied,
- the predicates $\top = x \mapsto 1$ and $\perp = x \mapsto 0$

Then we get:

$$\begin{aligned} \top \stackrel{c_{\circ}}{\rightarrow} \perp &\Leftrightarrow \forall c \in A. 1 \not\preceq \langle x \leftarrow c_{\circ} \cdot c \rangle 0 \\ &\Leftrightarrow \forall c \in A. 1 \not\preceq \bigwedge_{x \in v(c_{\circ} \cdot c)} .0 \Leftrightarrow 1 \not\preceq 1 \end{aligned}$$



All good ?

Consider the **PCA** example, that is:

$$MA = \{S \subseteq A \mid \forall x_1, x_2 \in S. x_1 = x_2\}$$

What about the following M-modality?

$$\langle x \leftarrow m \rangle \phi(x) \triangleq \bigcap_{x \in m} \phi(x)$$

Consider :

- $c_{\circ} \triangleq \langle \lambda x. \delta \delta \rangle$, that yields no value when applied,
- the predicates $\top = x \mapsto 1$ and $\perp = x \mapsto 0$

Then we get:

$$\begin{aligned} \top \xrightarrow{c_{\circ}} \perp &\Leftrightarrow \forall c \in \mathbb{A}. 1 \preceq \langle x \leftarrow c_{\circ} \cdot c \rangle 0 \\ &\Leftrightarrow \forall c \in \mathbb{A}. 1 \preceq \bigcap_{x \in v(c_{\circ} \cdot c)} .0 \Leftrightarrow 1 \preceq 1 \end{aligned}$$

X

Separator

A **separator** for $\diamond$ is:

- a combinatory complete subset $\mathcal{S}$ of $\mathbb{A}$,
- such that, for every $c_f, c_a \in \mathcal{S}$, “progress” holds:

$$\langle r \leftarrow c_f \cdot c_a \rangle \mathbf{0} \preceq \mathbf{0}$$

$\leadsto$ when progress holds for all codes, $\mathbb{A}$ is a separator

Packing all the ingredients we define

Monadic Core

A **monadic core** is a tuple $(\mathbb{A}, \Omega, \diamond, \mathcal{S})$, where:

- $\mathbb{A}$ is an MCA over a monad M ,
- $(\Omega, \preceq)$ is a complete **Hpa**,
- $\diamond$ is an M -modality over $\mathbb{A}$
- $\mathcal{S}$ is a separator over them.

Example - PCA

Monadic core

- sub-singleton monad

$$MA = \{S \subseteq A \mid \forall x_1, x_2 \in S. x_1 = x_2\}$$

- modality :

??

$\nabla \rightarrow$ applying e s.t. $\phi_1 \xrightarrow{e} \phi_2$ to c s.t. $\phi_1(c)$, $\langle x \leftarrow e \cdot c \rangle \phi_2$ should express that $e \cdot c$ returns a valid code for ϕ_2

- separator:

the whole set A

Example - PCA

Monadic core

- sub-singleton monad

$$MA = \{S \subseteq A \mid \forall x_1, x_2 \in S. x_1 = x_2\}$$

- modality :

??

$\nabla \rightarrow$ applying e s.t. $\phi_1 \xrightarrow{e} \phi_2$ to c s.t. $\phi_1(c)$, $\langle x \leftarrow e \cdot c \rangle \phi_2$ should express that $e \cdot c$ returns a valid code for ϕ_2

- separator:

the whole set A

Example - PCA

Monadic core

- sub-singleton monad

$$MA = \{S \subseteq A \mid \forall x_1, x_2 \in S. x_1 = x_2\}$$

- modality :

$$\langle x \leftarrow m \rangle \phi(x) \triangleq \bigsqcup_{x \in m} \phi(x)$$

$\nabla \rightarrow$ applying e s.t. $\phi_1 \xrightarrow{e} \phi_2$ to c s.t. $\phi_1(c)$, $\langle x \leftarrow e \cdot c \rangle \phi_2$ should express that $e \cdot c$ returns a valid code for ϕ_2

- separator:

the whole set A

Example - PCA

Monadic core

- sub-singleton monad

$$MA = \{S \subseteq A \mid \forall x_1, x_2 \in S. x_1 = x_2\}$$

- modality :

$$\langle x \leftarrow m \rangle \phi(x) \triangleq \bigsqcup_{x \in m} \phi(x)$$

$\nabla \rightarrow$ applying e s.t. $\phi_1 \xrightarrow{e} \phi_2$ to c s.t. $\phi_1(c)$, $\langle x \leftarrow e \cdot c \rangle \phi_2$ should express that $e \cdot c$ returns a valid code for ϕ_2

- separator:

the whole set A

Example - Classical realizability (1/2)

Quick reminder from Krivine realizability:

- fix a $\perp\!\!\!\perp$ ($\simeq$ “valid” computations) $\rightsquigarrow$ here, $\perp\!\!\!\perp \subseteq R$.
- orthogonality relation between $A \rightarrow R$ and A

$$k \perp a \triangleq k(a) \in \perp\!\!\!\perp$$

$$t \perp k \triangleq t(k) \in \perp\!\!\!\perp$$

$\rightsquigarrow$ realizers of ϕ = computations in $\phi^{\perp\!\!\!\perp} = \{t \mid \forall k. k \perp \phi \Rightarrow t \perp k\}$

- if $\perp\!\!\!\perp$ is non-empty, there exists $t \Vdash \perp$

$\rightsquigarrow$ restrictions to *proof-like* realizers (no captured continuation).

Example - Classical realizability (2/2)

- continuation monad

$$MA = (A \rightarrow R) \rightarrow R.$$

- modality ($m \in \phi^{\perp\perp}$), with fixed $\perp \subseteq R$:

$$\langle x \leftarrow m \rangle \phi(x) \triangleq \bigcap_{k \in A \rightarrow R} \left(\left(\bigcap_{a \in A} (\phi(a) \sqsupset k \perp a) \right) \sqsupset m \perp k \right)$$

- separator:

PL : codes obtained by combinatorial completeness extended with e_{cc}

The induced Evidenced frame

Theorem (Evidenced Frame over Monadic Core)

Let $\mathcal{MC}_M = (\mathbb{A}, \Omega, \diamond, \mathcal{S})$ be a monadic core. The triple $(\Omega^{\mathbb{A}}, \mathcal{S}, \cdot \dot{\rightarrow} \cdot)$, where

$$\phi_1 \xrightarrow{e} \phi_2 \triangleq \forall c \in \mathbb{A}. \phi_1(c) \preceq \langle r \leftarrow e \cdot c \rangle \phi_2(r)$$

forms an evidenced frame.

Preuve : As usual! see the paper ;-)

Theorem

Let $\mathcal{MC}_M = (\mathbb{A}, \Omega, \diamond, \mathcal{S})$ be a monadic core. Then the induced evidenced frame has an evidence $e \in \mathbb{A}$ such that $\top \xrightarrow{e} \perp$ iff $1 \preceq 0$.

The induced Evidenced frame

Theorem (Evidenced Frame over Monadic Core)

Let $\mathcal{MC}_M = (\mathbb{A}, \Omega, \diamond, \mathcal{S})$ be a monadic core. The triple $(\Omega^{\mathbb{A}}, \mathcal{S}, \cdot \dot{\rightarrow} \cdot)$, where

$$\phi_1 \xrightarrow{e} \phi_2 \triangleq \forall c \in \mathbb{A}. \phi_1(c) \preceq \langle r \leftarrow e \cdot c \rangle \phi_2(r)$$

forms an evidenced frame.

Preuve : As usual! see the paper ;-)

Theorem

Let $\mathcal{MC}_M = (\mathbb{A}, \Omega, \diamond, \mathcal{S})$ be a monadic core. Then the induced evidenced frame has an evidence $e \in \mathbb{A}$ such that $\top \xrightarrow{e} \perp$ iff $1 \preceq 0$.

Conclusion & bonus

Conclusion

Goal #1 : introduce an algebraic structure for realizability models that connects the **logical** and **computational** aspects.

- Any models induces a tripos (hence a topos)
- Implicative/arrow algebras : minimalistic axiomatization for *some* models

Evidenced Frame are super cool

An algebraic structure accounting for realizability models.

Conclusion

Goal #1 : introduce an algebraic structure for realizability models that connects the **logical** and **computational** aspects.

- Any models induces a tripos (hence a topos)
- Implicative/arrow algebras : minimalistic axiomatization for *some* models

Evidenced Frame are super cool

A

<i>flexible</i>	algebraic structure accounting for realizability models.
<i>uniform</i>	
<i>complete</i>	

Conclusion

Goal #2 : Smooth and uniform integration of useful computational effects.

- Monadic combinatory algebras: natural combinatorial view on monadic computations
- M -Modality to relate predicate on values and computations

MCA are really nice

Smoothly extend all the constructions/results for PCA.

Bonus : Syntactic Realizability

Different tradition : Gödel's *Dialectica*, Kreisel *modified realizability*, etc...

In a nutshell: from a derivation of $\Gamma \vdash A$, produce

- a type A^*
- a term t s.t. $\Gamma^* \vdash t : A^*$
- which is a realizer of A

Goal #3: mimick the results obtains with MCA, and develop a unified framework for syntactic realizability models.

We aim at a framework that:

- models expressive systems like HOL or type theory
- supports effectful computational interpretations
- enables the perspective of syntactic proof transformations
- can be formalized in proof assistants

Bonus : Syntactic Realizability

Different tradition : Gödel's *Dialectica*, Kreisel *modified realizability*, etc...

In a nutshell: from a derivation of $\Gamma \vdash A$, produce

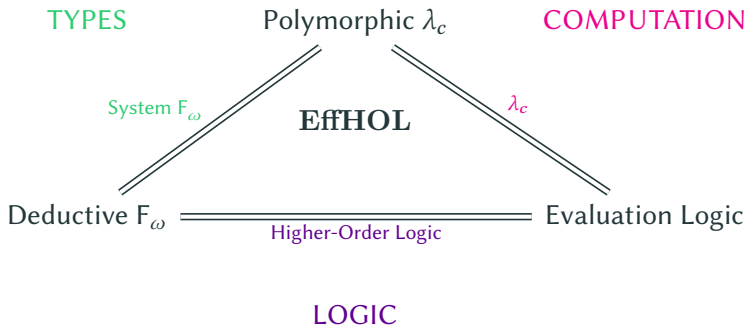
- a type A^*
- a term t s.t. $\Gamma^* \vdash t : A^*$
- which is a realizer of A

Goal #3: mimick the results obtains with MCA, and develop a unified framework for syntactic realizability models.

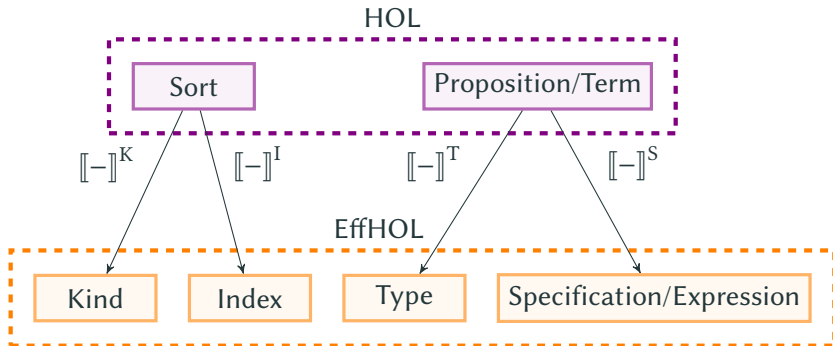
We aim at a framework that:

- models expressive systems like HOL or type theory
- supports effectful computational interpretations
- enables the perspective of syntactic proof transformations
- can be formalized in proof assistants

Contribution: Effectful Higher-Order Logic (EffHOL)



Realizability Translation



Theorem (Soundness)

If $\Gamma \vdash_{\text{HOL}} \varphi$ then there is $p : M(\llbracket \varphi \rrbracket^T)$ such that $\llbracket \Gamma \rrbracket^K \mid \llbracket \Gamma \rrbracket^I \vdash_{\text{EffHOL}} \langle x \leftarrow p \rangle \llbracket \varphi \rrbracket^S(x)$.

Bibliography



L. Cohen, É. Miquey, and R. Tate.

Evidenced frames: A unifying framework broadening realizability models.

In *Proceeding of LICS 2021*.



L. Cohen, A. Grunfeld, D. Kirst, and E. Miquey.

From Partial to Monadic: Combinatory Algebra with Effects.

In *Proceedings of FSCD'25*, volume 337 of *Leibniz International Proceedings in Informatics (LIPIcs)*, 2025.



L. Cohen, A. Grunfeld, D. Kirst, and E. Miquey.

Syntactic Effectful Realizability in Higher-Order Logic.

In *Proceedings of LICS 2025*, June 2025.

URL: <https://hal.science/hal-05106028>.

MCA, categorically (1/2)

Definition

For a Freyd category $\langle \mathbb{C}, \mathbb{K}, J \rangle$, an *applicative object* is an object $\mathbb{A} \in \mathbf{Obj}(\mathbb{C})$ equipped with an application morphism $\otimes \in \mathbb{K}(\mathbb{A} \times \mathbb{A}, \mathbb{A})$.

Definition

Given $\mathbb{A}$ applicative in $\langle \mathbb{C}, \mathbb{K}, J \rangle$, a morphism $f \in \mathbb{K}(\mathbb{A}^n, \mathbb{A})$ is **$\mathbb{A}$ -computable** when, for all $k < n$, and all $c_1, \dots, c_k \in \mathbb{C}(\mathbf{1}, \mathbb{A})$, there is a code $e_{f(c_1, \dots, c_k)} \in \mathbb{C}(\mathbf{1}, \mathbb{A})$ s.t.:

$$\begin{array}{ccc}
 \mathbf{1} \times \mathbb{A}^{n-k} & \xrightarrow{J e_{f(c_1, \dots, c_k)} \times \mathbb{A}^{n-k}} & \mathbb{A} \times \mathbb{A}^{n-k} \\
 J \langle c_1, \dots, c_k \rangle_k \times \mathbb{A}^{n-k} \downarrow & & \downarrow \otimes^{n-k} \\
 \mathbb{A}^k \times \mathbb{A}^{n-k} & \xrightarrow{\alpha^k} \mathbb{A}^n \xrightarrow{f} & \mathbb{A}
 \end{array}$$

$$\begin{array}{ccc}
 \mathbf{1} & \xrightarrow{J \langle e_{f(c_1, \dots, c_k)} \rangle_{k+1}} & \mathbb{A} \times \mathbb{A}^k \\
 J e_{f(c_1, \dots, c_k)} \searrow & & \downarrow \otimes^k \\
 & & \mathbb{A}
 \end{array}$$

MCA, categorically (2/2)

An **$\mathbb{A}$ -monomial** is a morphism in $\mathbb{K}(\mathbb{A}^n, \mathbb{A})$ defined using only projections, morphisms in $\mathbb{C}(\mathbf{1}, \mathbb{A})$, and application $\otimes$, used in an applicative order.

Definition

An applicative object $\mathbb{A}$ is called a *combinatory object* when all positive $\mathbb{A}$ -monomials are $\mathbb{A}$ -computable.

Theorem

$\mathbb{A}$ is an MCA over M iff if it is a combinatory object in $\mathbf{Set}_M$.