Transforming classical mathematics in type theory into oracle computation

Sewon Park (Inštitut za Matematiko, Fiziko in Mehaniko, Ljubljana, Slovenia)

based on j.w.w. Andrej Bauer (University of Ljubljana, Slovenia)
and Danel Ahman (University of Tartu, Estonia)

July 30, 2026
CCA 2026, Trier, Germany

Motivation – Classical Mathematics in Type Theory

Classical Principles

Many classical principles are of the form

$$\forall(x : \mathcal{A}).\ \mathcal{P}\ x \quad \text{where} \quad \mathcal{A} : \text{Type and } \mathcal{P} : \mathcal{A} \to \text{Prop}$$

- The law of excluded middle: $\forall(\varphi : \text{Prop}).\ \varphi \vee \neg\varphi$
- The Limited Principle of Omniscience (LPO):
  $\forall(f : \mathbb{N} \to \mathbb{B}).\ (\exists(n : \mathbb{N}).\ f\ n = 0) \vee (\forall(n : \mathbb{N}).\ f\ n = 1)$
- Trichotomy of real numbers:
  $\forall(x, y : \mathbb{R}).\ (x < y) \vee (x = y) \vee (x > y)$

Some principles give data

$$\Pi(x : \mathcal{A}).\ \mathcal{P}\ x \quad \text{where} \quad \mathcal{A} : \text{Type and } \mathcal{P} : \mathcal{A} \to \text{Type}$$

$f : \Pi(x : \mathcal{A}).\ \mathcal{P}\ x$ is a dependent function s.t. $f\ a : \mathcal{P}\ a$ for each $a : \mathcal{A}$
Prop is just another type universe, and $\forall$ and Π are the same thing.

- A strong form of the law of excluded middle: $\Pi(\varphi : \text{Prop}).\ \varphi + \neg\varphi$
- Even stronger form of the law of excluded middle:
  $\Pi(X : \text{Type}).\ X + \neg X$
- The global axiom of choice: $\Pi(X : \text{Type}).\ (\exists(x : X).\ \text{True}) \to X$

Classical Mathematics in Type Theory

- Given a classical principle $\Pi(x : \mathcal{A}).\ \mathcal{P}\ x$,

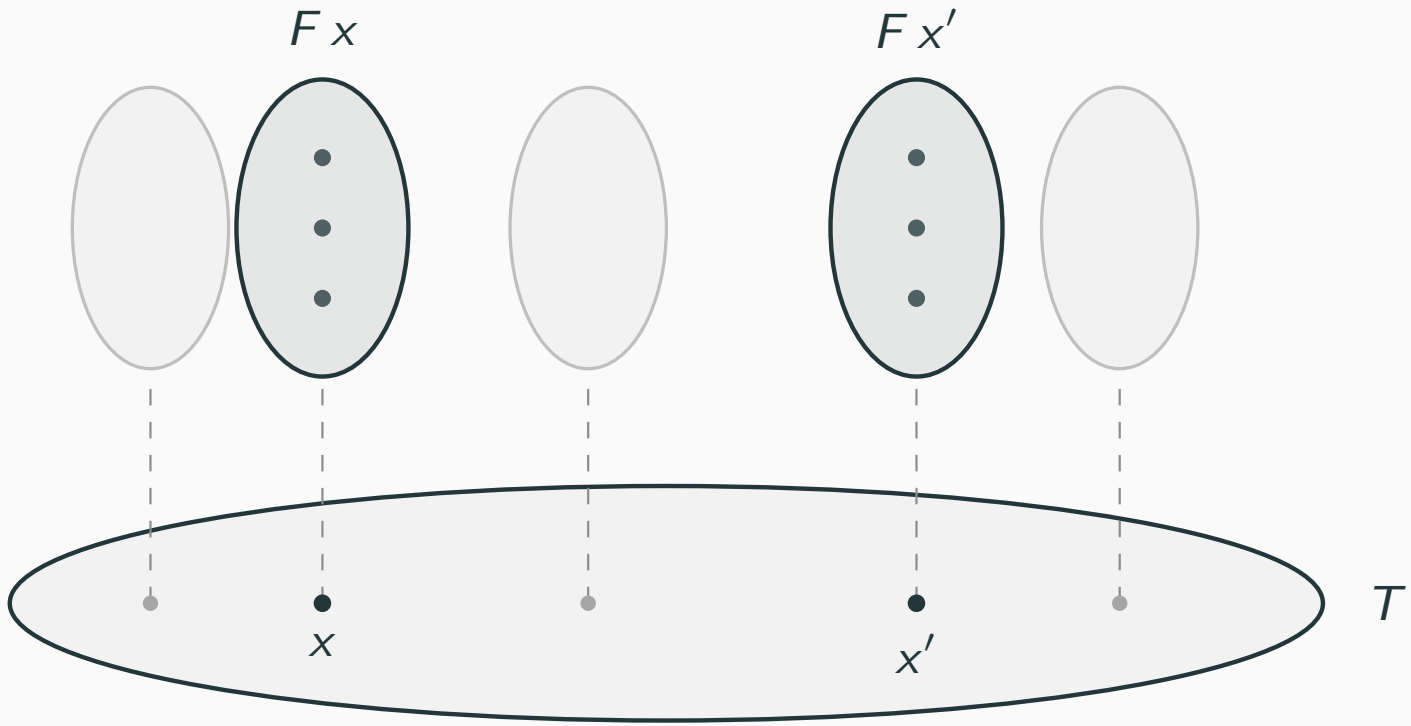
$$\vdash \big(\lambda(f : \Pi(x : \mathcal{A}).\ \mathcal{P}\ x).\ t\big) : \big(\Pi(x : \mathcal{A}).\ \mathcal{P}\ x\big) \to X$$

  where t may have subterm $f\ x : \mathcal{P}\ x$ for some $x : \mathcal{A}$ locally consulting the classical principle.
- This is a usual way of representing a classical proof (or a construction) of X : Type.
- This constructive representation of classical mathematics fails when the principle

$$\Pi(x : \mathcal{A}).\ \mathcal{P}\ x$$

  can be falsified in the computational model.
- $[\![\mathsf{LPO}]\!] =$ *there is a TM that, given* $f : \mathbb{N} \to \mathbb{B}$, *decides* $\forall n.\ f\ n = 1$.
  $= \bot$
  Hence, $[\![\mathsf{LPO} \to X]\!]$ is trivially $\top$.
- An alternative way of representing classical mathematics, sound in all models, is needed.

Type Families, Pictorially



A family $F : T \to \text{Type}$ assigns a type $F\,x$ — a *fiber* — to each $x : T$.

The union $\Sigma(x : T).\ F\,x$ is the type of *pairs* (x, y) with $x : T$ and $y : F\,x$: all the fibers gathered together.

Theorem (Countable union of countable sets is countable)

Define Countable $(T : \text{Type}) :\equiv \exists(f : T \to \mathbb{N}).$ Injective f
where Injective $(f : S \to T) :\equiv \forall(x, y : S).\ f\ x = f\ y \to x = y.$

For any countable $T : \text{Type}$ and a family of countable types
$F : T \to \text{Type}$ (with $H : \forall(x : T).$ Countable $(F\ x)$)
the union $\Sigma(x : T).\ F\ x$ is countable.

For simplicity, assume we already have an injective $f : T \to \mathbb{N}$

Axiom `Choice` : $\Pi(T : \text{Type}).\ (\exists(x : T).\ \text{True}) \to T$ as a function.

Proof. Define a mapping $\Sigma(x : T).\ F\ x \to \mathbb{N}$ by

1. Given input arguments $x : T$ and $y : F\ x$.
2. Call `Choice` on $H\ x : \text{Countable}\ (F\ x)$ to obtain $g_x : (F\ x) \to \mathbb{N}$.
3. Return the Cantor pair $\langle f\ x,\ g_x\ y \rangle : \mathbb{N}$.

Then prove the mapping is injective.

Remarks:

1. g_x in (2) is reconstructed for each input (x, y), but is **consistent** across other inputs (x', y') if $x = x'$ because `Choice` is a function.
2. Since `Choice` is an axiomatic term, the function **does not compute, even if** an explicit labeling for $F\ x$ is already available.

Oracle Computation in Type Theory

Oracle Computation

An **oracle** is specified by a pair $(\mathcal{A}, \mathcal{P})$ of a type $\mathcal{A} : \text{Type}$ of **queries** and a family of types $\mathcal{P} : \mathcal{A} \to \text{Type}$ of **answers**.

Example (Law of Excluded Middle)

$\mathcal{A} \equiv \text{Prop} \qquad \mathcal{P}\ \varphi \equiv \varphi \vee \neg\varphi$

Example (Trichotomy of Real Numbers)

$\mathcal{A} \equiv \mathbb{R} \times \mathbb{R} \qquad \mathcal{P}\ (r, s) \equiv r < s \vee r = s \vee r > s$

Example (Axiom of Choice)

$\mathcal{A} \equiv \Sigma(X : \text{Type}).\ \big(\exists(x : X).\ \text{True}\big) \qquad \mathcal{P}\ (X, \_) \equiv X$

For an oracle $(\mathcal{A}, \mathcal{P})$, the type of **oracle computations** is defined inductively:

$$
\begin{aligned}
&\textbf{inductive}\ O_{\mathcal{A} \triangleleft \mathcal{P}}\ (X : \text{Type}) : \text{Type} \\
&\mid \texttt{ret} : X \to O_{\mathcal{A} \triangleleft \mathcal{P}}\ X \\
&\mid \texttt{ask} : \Pi(a : \mathcal{A}).\ \big(\mathcal{P}\ a \to O_{\mathcal{A} \triangleleft \mathcal{P}}\ X\big) \to O_{\mathcal{A} \triangleleft \mathcal{P}}\ X
\end{aligned}
$$

Oracle Computation (cont.)

An **oracle** is specified by a pair $(\mathcal{A}, \mathcal{P})$ of a type $\mathcal{A} : \mathrm{Type}$ of **queries** and a family of types $\mathcal{P} : \mathcal{A} \to \mathrm{Type}$ of **answers**.

Example (Axiom of Choice)

$\mathcal{A} \equiv \Sigma(X : \mathrm{Type}).\ \big(\exists(x : X).\ \mathrm{True}\big) \qquad \mathcal{P}\ (X, \_) \equiv X$

For an oracle $(\mathcal{A}, \mathcal{P})$, the type of **oracle computations** is defined inductively:

$$\begin{aligned} &\textbf{inductive}\ O_{\mathcal{A} \triangleleft \mathcal{P}}\ (X : \mathrm{Type}) : \mathrm{Type} \\ &\mid \mathtt{ret} : X \to O_{\mathcal{A} \triangleleft \mathcal{P}}\ X \\ &\mid \mathtt{ask} : \Pi(a : \mathcal{A}).\ \big(\mathcal{P}\ a \to O_{\mathcal{A} \triangleleft \mathcal{P}}\ X\big) \to O_{\mathcal{A} \triangleleft \mathcal{P}}\ X \end{aligned}$$

Example (Axiom of Choice)

Write $\mathtt{AC}$ for the oracle corresponding to the Axiom of Choice. Then:

1. $\mathtt{ret}\ 42 : O_{\mathtt{AC}}\ \mathbb{N}$.
2. $\mathtt{ask}\big(\{\mathtt{true}, \mathtt{false}\}, \lambda b.\ \textbf{if}\ b\ \textbf{then}\ \mathtt{ret}\ 42\ \textbf{else}\ \mathtt{ret}\ 0\big) : O_{\mathtt{AC}}\ \mathbb{N}$.
3. $\mathtt{ask}(S, \kappa) : O_{\mathtt{AC}}\ \mathbb{N}$ for a nonempty S and continuation $\kappa : S \to O_{\mathtt{AC}}\ \mathbb{N}$.

Goal: Reformulate classical proofs and constructions using only oracle

Classical Mathematics as Oracle Computation – First attempt

Theorem (Axiom-free, oracle-version)

Define **Oracle**-Countable $T :\equiv \exists(f : T \to O_{\mathtt{AC}}\,\mathbb{N})$. **Oracle**-Injective f
&**Oracle**-Injective$(f : S \to O_{\mathtt{AC}}\,T) :\equiv \forall(x, y : S).\ f\,x \approx f\,y \to O_{\mathtt{AC}}(x = y)$

For any countable T : Type and a family of countable types
$F : T \to$ Type (with $H : \forall(x : T).$ Countable $(F\,x)$)
the union $\Sigma(x : T).\ F\,x$ is **oracle**-countable.

For simplicity, assume we already have an injective $f : T \to \mathbb{N}$

Proof. Define a mapping $\Sigma(x : T).\ F\,x \to O_{\mathtt{AC}}\,\mathbb{N}$ by:

- Given $x : T$ and $y : F\,x$, we need to return an element of $O_{\mathtt{AC}}\,\mathbb{N}$.
- Instead of calling `Choice`, `ask` the oracle with the query $(H\,x)$ to obtain a labeling $g_x : F\,x \to \mathbb{N}$, and `return` $\langle f\,x, g_x\,y\rangle$.

Remarks:

1. We obtain an explicit, axiom-free term:

   $$h :\equiv \lambda(x, y).\ \mathtt{ask}(H\,x, \lambda g_x.\ \mathtt{ret}\langle fx, g_x y\rangle)$$

   an oracle computation of type $\big(\Sigma(x : T).\ F\,x\big) \to O_{\mathtt{AC}}\mathbb{N}$

2. When g_x is already known, we can access the value $\langle f\,x,\ g_x\,y\rangle : \mathbb{N}$.

Consistency Problem

Recall:

- Oracle-Injective$(f : S \to O_{\mathtt{AC}}\, T) :\equiv \forall (x, y : S).\ f\ x = f\ y \to O_{\mathtt{AC}}(x = y)$
- $h :\equiv \lambda(x, y).\ \mathtt{ask}(H\ x, \lambda g_x.\ \mathtt{ret}\langle f\ x, g_x\ y\rangle)$

Problem: Now we need to prove the injectivity of h.

$$\mathtt{ask}(H\ x, \lambda g_x.\ \mathtt{ret}\langle f\ x, g_x\ y\rangle) \approx \mathtt{ask}(H\ x', \lambda g_{x'}.\ \mathtt{ret}\langle f\ x', g_{x'}\ y'\rangle)$$

implying $O_{\mathtt{AC}}\big((x, y) = (x', y')\big)$ for all x, y, x', y'.

Observation: We need equations in $O_{\mathtt{AC}}\mathbb{N}$.
Candidates:(1) $\mathtt{ask}(a, \lambda\_.t) \approx t$ and (2) $\mathtt{ask}(a, \kappa) \approx \kappa\ s$ for any $s : \mathcal{P}\ a$
Equation (1) doesn't help because κ in h is not constant.
Assume Equation (2):

1. Instantiate g_x and $g_{x'}$ by asking, since we are now inside $O_{\mathtt{AC}}\big((x, y) = (x', y')\big)$.
2. $\mathrm{LHS} \approx \mathtt{ret}\langle f\ x, g_x\ y\rangle$ and $\mathrm{RHS} \approx \mathtt{ret}\langle f\ x', g_{x'}\ y'\rangle$
3. We are **stuck** here because, even when $x = x'$, we do not ha $g_x = g_{x'}$: they are independent oracle answers whose queries happen to coincide.
4. This is because the call to $\mathtt{AC}$ is no longer a function: the g_x chosen at input (x, y) and the g_x chosen at input (x, y') need not agree.

Consistency Problem – Another Example

Classical construction of natural numbers:

$$x :\equiv \textbf{let } b := \texttt{Choice } \mathbb{B} \textbf{ in let } n := \texttt{Choice } \mathbb{N} \textbf{ in}$$
$$\textbf{if } b \textbf{ then } 2 \times n \textbf{ else } 2 \times n + 1$$

$$y :\equiv \textbf{let } n := \texttt{Choice } \mathbb{N} \textbf{ in let } b := \texttt{Choice } \mathbb{B} \textbf{ in}$$
$$\textbf{if } b \textbf{ then } 2 \times n \textbf{ else } 2 \times n + 1$$

Then, $x = y$ is provable.
Corresponding oracle computation of natural numbers $O_{\mathtt{AC}}\mathbb{N}$:

$$s :\equiv \mathtt{ask}\Big(\mathbb{B}, \lambda b.\ \mathtt{ask}\big(\mathbb{N}, \lambda n.$$
$$\textbf{if } b \textbf{ then } \mathtt{ret}(2 \times n) \textbf{ else } \mathtt{ret}(2 \times n + 1)\big)\Big)$$

$$t :\equiv \mathtt{ask}\Big(\mathbb{N}, \lambda n.\ \mathtt{ask}\big(\mathbb{B}, \lambda b.$$
$$\textbf{if } b \textbf{ then } \mathtt{ret}(2 \times n) \textbf{ else } \mathtt{ret}(2 \times n + 1)\big)\Big)$$

However, $s \approx t$ is *not* provable from these equations.

The Project Overview

Goal:

- Given a construction (or proof) in a classical type theory

$$\Gamma \vdash_{\text{classical}} t : X$$

  using a constant $\texttt{axiom} : \Pi(x : \mathcal{A}).\ \mathcal{P}\ x$,
- systematically reformulate it as

$$[\![\Gamma]\!] \vdash [\![t]\!] : [\![X]\!]$$

  in a constructive type theory, where $[\![X]\!]$ is related to the original X via $O_{\mathcal{A} \triangleleft \mathcal{P}}$.
- Implement $[\![\cdot]\!]$ as a syntactic translation via Lean's metaprogramming, and translate classical Mathlib libraries.
- This requires handling Lean-specific features: proof-irrelevant `Prop`, universe polymorphism, inductive types, and *Lean's formulation of the Axiom of Choice*.

Related Work, Limitations, and Contribution

Related Work:

- When $\mathcal{P}\,x$ is propositional (i.e., at most one possible answer in $\mathcal{P}\,x$), there is no consistency problem.

E.g., The law of excluded middle: $\mathcal{A} \equiv \mathsf{Prop}$, $\mathcal{P}\,\varphi \equiv \varphi \vee \neg\varphi$.

- The quotient inductive type on $O_{\mathcal{A} \triangleleft \mathcal{P}}$ with equations

$$\mathtt{ask}\big(a,\ \lambda\_.\ t\big) = t$$

yields the sheafification (Baillon, Mahboubi & Pédrot; Bauer & Ahman).

Limitations:

- For non-propositional $\mathcal{P}$ (including $\mathtt{AC}$), the prior theory does not extend.

Contribution:

- Extend the theory of propositional oracles to arbitrary $(\mathcal{A}, \mathcal{P})$ by **lifting** into a presheaf topos where the oracle becomes propositional.

Propositional Case – Application to Real Computation

Intermediate Value Theorem – Classical

Assume the law of excluded middle (LEM)

$$\texttt{lem} : \forall(\varphi : \mathrm{Prop}).\ \varphi \vee \neg\varphi$$

and prove the Intermediate Value Theorem (IVT).
For $f : \mathbb{R} \to \mathbb{R}$ continuous, define:

$$\begin{aligned}
\texttt{Bisect} : \mathbb{R} \times \mathbb{R} &\to \mathbb{R} \times \mathbb{R} \\
(x, y) &\mapsto \textbf{let } m := (x + y)/2 \\
&\quad\ \textbf{case } \texttt{lem}(f\ m > 0) \\
&\quad\ \mid f\ m > 0 \Rightarrow (x, m) \\
&\quad\ \mid \neg(f\ m > 0) \Rightarrow (m, y)
\end{aligned}$$

construct a concrete

$$r := \lim_n(\pi_1(\texttt{Bisect}^n(a, b))) : \mathbb{R}$$

and prove that $f\ r = 0$ when $f\ a < 0 < f\ b$.
This proves LEM $\Rightarrow$ IVT.

Intermediate Value Theorem – Oracle Version

Ordinary reals:

$$\mathbb{R} :\equiv \big(\Sigma(q : \mathbb{N} \to \mathbb{Q}).\ \forall(n : \mathbb{N}).\ |q_{n+1} - q_n| < 2^{-n}\big)/{\sim}$$

Oracle computational reals:

$$\mathbb{R}_{\mathsf{LEM}} :\equiv \big(\Sigma(q : \mathbb{N} \to O_{\mathsf{LEM}}\mathbb{Q}).\ \forall(n : \mathbb{N}).\ O_{\mathsf{LEM}}\big(|q_{n+1} - q_n| < 2^{-n}\big)\big)/{\sim}$$

$\mathbb{R}_{\mathsf{LEM}}$ as a sheaf, admits:

$$\mathtt{ask} : \forall(\varphi : \mathsf{Prop}).\ (\varphi \vee \neg\varphi \to \mathbb{R}_{\mathsf{LEM}}) \to \mathbb{R}_{\mathsf{LEM}}$$

Reformulate:

$$\begin{aligned}
\mathtt{Bisect}_{\mathsf{LEM}} : \mathbb{R}_{\mathsf{LEM}} \times \mathbb{R}_{\mathsf{LEM}} &\to \mathbb{R}_{\mathsf{LEM}} \times \mathbb{R}_{\mathsf{LEM}} \\
(x, y) &\mapsto \textbf{let}\ m := (x + y)/2 \\
&\quad \mathtt{ask}\ \big(f\ m > 0, \lambda(b : f\ m > 0 \vee \neg(f\ m > 0)).\ \\
&\qquad \textbf{case}\ b \\
&\qquad \mid f\ m > 0 \Rightarrow \mathtt{ret}(x, m) \\
&\qquad \mid \neg(f\ m > 0) \Rightarrow \mathtt{ret}(m, y)\big)
\end{aligned}$$

Then, $\lim_n(\pi_1(\mathtt{Bisect}^n_{\mathsf{LEM}}(a, b)))$ is a root of $f : \mathbb{R}_{\mathsf{LEM}} \to \mathbb{R}_{\mathsf{LEM}}$

Refinement by Resolving Oracle Queries

$$\begin{aligned}\mathtt{Bisect}_{\mathsf{LEM}} : \mathbb{R}_{\mathsf{LEM}} \times \mathbb{R}_{\mathsf{LEM}} &\to \mathbb{R}_{\mathsf{LEM}} \times \mathbb{R}_{\mathsf{LEM}} \\ (x, y) &\mapsto \textbf{let } m := (x+y)/2 \\ &\quad \mathtt{ask}\left(f\ m > 0, \lambda(b : f\ m > 0 \vee \neg(f\ m > 0)).\right. \\ &\quad\quad \textbf{case } b \\ &\quad\quad \mid f\ m > 0 \Rightarrow \mathtt{ret}(x, m) \\ &\quad\quad \left. \mid \neg(f\ m > 0) \Rightarrow \mathtt{ret}(m, y)\right)\end{aligned}$$

- When $x \neq 0$, we can construct $t : x > 0 \vee \neg(x > 0)$.
- When $f\left((a+b)/2\right) \neq 0$,

$$\mathtt{Bisect}_{\mathsf{LEM}}(a, b) \approx \mathtt{ret}\ \cdots$$

- When f does not admit a dyadic root, we can refine $\lim_n(\pi_1(\mathtt{Bisect}^n_{\mathsf{LEM}}(a, b)))$ into the actual real number root.

$\Rightarrow$ Reusing the classical bisection proof to yields computation

Non-propositional Oracles via Forcing

Making $\mathcal{P}\,a$ Propositional via Forcing

- Define Level : Type as the type of partial assignments $\ell : \Pi(x : \mathcal{A}).\ \mathcal{P}\ x \to \mathrm{Prop}$ that are finite and functional. (The forcing conditions.)
- A level ℓ represents a finite history of oracle answers: $\ell(a, s) \iff$ "s is the answer to query a".
- Level is a poset: $\ell \leq m$ iff ℓ refines m consistently (i.e. $\forall(a, s).\ m(a, s) \Rightarrow \ell(a, s)$).

E.g.,

$$\{(\mathbb{B}, \texttt{true}), (\mathbb{N}, 42)\} \ \leq\ \{(\mathbb{B}, \texttt{true})\} \ \leq\ \{\}$$
$$\{(\mathbb{B}, \texttt{false}), (\mathbb{N}, 42)\} \ \not\leq\ \{(\mathbb{B}, \texttt{true})\} \ \leq\ \{\}$$

- We consider the presheaf category

$$\mathrm{Presheaf} :\equiv \mathrm{Level}^{op} \to_{\mathrm{functorial}} \mathrm{Type}$$

and for each $a : \mathcal{A}$ define a presheaf $\hat{P}_a$ by

$$\hat{P}_a(\ell) \ \equiv\ \Sigma(s : \mathcal{P}\ a).\ \ell(a, s)$$

- Each $\hat{P}_a$ is propositional (i.e., has at most one element at every ℓ).

Parameterized Oracle Computation

- Calculating the inductive type $O_{\mathcal{A} \triangleleft \mathcal{P}}\, X$ inside the presheaf topos yields:

  inductive $O_{\mathcal{A} \triangleleft \mathcal{P}}\, (X : \mathrm{Type}) : \mathrm{Level} \to \mathrm{Type}$

  $\mid \mathtt{ret}^{\ell} : X \to (O_{\mathcal{A} \triangleleft \mathcal{P}}\, X)(\ell)$

  $\mid \mathtt{ask}^{\ell} : \Pi(a : \mathcal{A}).\ \Big(\Pi(s : \mathcal{P}\ a).\ \text{“}(a, s) \text{ compat } \ell\text{”}$

  $\to (O_{\mathcal{A} \triangleleft \mathcal{P}}\, X)(\ell[a := s])\Big) \to (O_{\mathcal{A} \triangleleft \mathcal{P}}\, X)(\ell)$

- $O_{\mathcal{A} \triangleleft \mathcal{P}} X(\ell) \approx$ *“oracle computation of type X, given ℓ as the record of the oracle calls and answers so far that the oracle must respect for future oracle queries”*
- Restriction map $\restriction_{\ell \leq m} : (O_{\mathcal{A} \triangleleft \mathcal{P}}\, X)(m) \to (O_{\mathcal{A} \triangleleft \mathcal{P}}\, X)(\ell)$ works by cutting down branches.

Sheaf Condition and Computation

1. The **sheaf equation** $\mathtt{ask}\big(a,\ \lambda\_.\ t\big) \approx t$ calculates to

$$\mathtt{ask}^{\ell}\big(a,\ \lambda s\, h.\ t\restriction_{\ell[a:=s]\le\ell}\big)\ \approx\ t,$$

   for any $t : (O_{\mathcal{A}\triangleleft\mathcal{P}}\, X)(\ell)$ and any $a : \mathcal{A}$.
   Gluing the restrictions of t back together recovers t.
2. The sheaf equation, together with the naturality of $\mathtt{ask}$, yields

$$\mathtt{ask}^{\ell}(a,\kappa)\restriction_{\ell[a:=s]\le\ell} \approx \kappa\ s\ h$$

   for all $s : \mathcal{P}\ a$ and $h : (a, s)$ compat ℓ.
3. Intuitively, κ is a matching family and $\mathtt{ask}$ computes the unique amalgamation.
4. Lemma: $(O_{\mathcal{A}\triangleleft\mathcal{P}}X, \approx)$ is the sheafification of X.

Sheafification Examples

Oracle natural numbers $O_{\mathtt{AC}}\mathbb{N}$ at the empty level $\{\}$:

$$s :\equiv \mathtt{ask}^{\{\}}\Big(\mathbb{B},\lambda b\ p.\ \mathtt{ask}^{\{\mathbb{B}:=b\}}\big(\mathbb{N},\lambda n\ q.\ \textbf{if } b \textbf{ then } \mathtt{ret}^{\{\mathbb{B}:=b,\mathbb{N}:=n\}}(2\times n) \textbf{ else } \mathtt{ret}^{\{\mathbb{B}:=b,\mathbb{N}:=n\}}(2\times n+1)\big)\Big)$$

and

$$t :\equiv \mathtt{ask}^{\{\}}\Big(\mathbb{N},\lambda n\ q.\ \mathtt{ask}^{\{\mathbb{N}:=n\}}\big(\mathbb{B},\lambda b\ p.\ \textbf{if } b \textbf{ then } \mathtt{ret}^{\{\mathbb{N}:=n,\mathbb{B}:=b\}}(2\times n) \textbf{ else } \mathtt{ret}^{\{\mathbb{N}:=n,\mathbb{B}:=b\}}(2\times n+1)\big)\Big)$$

s and t, though syntactically different, are indistinguishable as oracle computations; hence $s \approx t$ holds.

Example Revisited

Theorem (Axiom-free, leveled-oracle-version)

Define Oracle-Countable $T :\equiv \exists(f : T \to O_{\mathtt{AC}}\,\mathbb{N})$. Oracle-Injective f & Oracle-Injective$(f : S \to O_{\mathtt{AC}}\,T) :\equiv \forall x, y.\ f\,x \approx f\,y \to O_{\mathtt{AC}}(x = y)$

$(\cdots)$ the union $\Sigma(x : T).\ F\,x$ is **oracle**-countable.

Recall: $h :\equiv \lambda(x, y).\ \mathtt{ask}(H\,x, \lambda g_x.\ \mathtt{ret}\langle f\,x, g_x\,y\rangle)$

$\rightsquigarrow$ **Calculates to**: $\lambda(x, y).\ \mathtt{ask}^{\ell}(H\,x, \lambda g_x.\ \mathtt{ret}^{\ell[H\,x:=g_x]}\langle f\,x, g_x\,y\rangle)$

Injectivity calculates to:

$\forall \ell.\ \big(\Pi(g_x : F\,x \to \mathbb{N})(g_{x'} : F\,x' \to \mathbb{N}).$

$\quad g_x, g_{x'}$ compat $\ell \to \langle f\,x, g_x\,y\rangle = \langle f\,x', g_{x'}\,y'\rangle\big) \to O_{\mathtt{AC}}\big((x, y) = (x', y')(\ell)\big)$

In the context of t in $\mathtt{ask}^{\ell}(H\,x, \lambda g_x\,h_x.\ \mathtt{ask}^{\ell[H\,x:=g_x]}(H\,x', \lambda g_{x'}\,h_{x'}$
$h_{x'}$ provides the consistency condition between $g_{x'}$ and g_x.

Systematic Reformulation – WIP

Idea

Given a type expression

$$\Gamma \vdash X : \mathrm{Type}$$

from a small fragment of Lean's type theory,

construct it inside the internal sheaf category

$$[\![\Gamma]\!] \vdash [\![X]\!] : \mathrm{Sheaf}$$

inside Lean. And a term

$$\Gamma \vdash t : X$$

we get an axiom-free term

$$[\![\Gamma]\!] \vdash [\![t]\!] : [\![X]\!]\{\}$$

This only makes sense if

$$[\![\vdash \Pi(x : \mathcal{A}).\ \mathcal{P}\ x]\!]$$

as a sheaf admits a global section; i.e., the subcategory of sheaves is a model of a type theory that realizes $\Pi(x : \mathcal{A}).\ \mathcal{P}\ x$

Limitation

Sheaves admit

$$\mathsf{op}_X : \Pi(a : \Delta\mathcal{A}).\ (\hat{\mathcal{P}}a \to X) \to X$$

However, $\mathcal{A}$, as interpreted in the sheaf category, is not in general compatible with $\Delta\mathcal{A}$

E.g., in LPO, $[\![\mathbb{N} \to \mathbb{B}]\!] \simeq \mathbb{N} \to O_{\mathsf{LPO}}\mathbb{B}$ but cannot be cast to be a query argument $O_{\mathsf{LPO}}(\mathbb{N} \to \mathbb{B})$

When the query type $\mathcal{A}$ is of order 0 (arrow-free), the category of sheaves can model the oracle type theory.

If not, we use Call-By-Push-Value-style type theory, where value types and computation types are distinguished.
Queries must be of value types.

Adjunction between presheaves and sheaves provides a model.

Implementation – WIP

Future Work

Universe.

1. Necessary, since axioms in Lean often appear in the construction of universe elements; e.g., x^{-1} is defined using the Axiom of Choice, so $x \cdot x^{-1} = 1$ is a type depending on the Axiom of Choice.
2. Plan: a syntactic universe given by an inductive type of universe codes (for translation).

Effectful Type Theory.

1. Formalize the CBPV type theory where there is

$$\Pi(x : (\text{value})\ \mathcal{A}).\ (\text{comp})\ \mathcal{P}\, x$$

Application.

1. Formalize the theory of oracle-computable real numbers as a bridge between classical analysis and constructive analysis formalizations.

Implementation.

1. Formalizing the theory of sheaves and their setoid models in Lean.
2. Even before formalization is complete, implementing the metaprogramming translation.

Thank you!