A robust implementation of Axioms of Choice

Liron Cohen
Cornell University, Ithaca, NY, USA
Extending the Proofs-as-Programs Paradigm

Proofs = Programs

How can modern notions of computation influence and contribute to formal foundations?
Extending the Proofs-as-Programs Paradigm

Proofs = Programs

How can modern notions of computation influence and contribute to formal foundations?
Extending the Proofs-as-Programs Paradigm

How can modern notions of computation influence and contribute to formal foundations?
The Axiom of Choice

Given any collection of nonempty sets, there is a way to assign a representative element to each set in the collection.
Motivation

- AC unifies standard constructive representations of the reals.

Dedekind cuts

Cauchy sequences
AC unifies standard constructive representations of the reals.

- **Dedekind cuts**
  - Computationally inefficient
- **Cauchy sequences**
  - Not constructively complete
Motivation

- AC unifies standard constructive representations of the reals.
  - Dedekind cuts
    - computationally inefficient
  - Cauchy sequences
    - not constructively complete

- Unclear status in constructivism.
  - Some variants are considered trivially true due to the specific interpretation of the type constructors $\Sigma$ and $\Pi$.
  - Prior constructive models of choice implicitly rely on a deterministic computation system.
    $\Rightarrow$ Fail to extend with new computational capabilities.
Logical Statements

Given any collection of nonempty sets, there is a way to assign a representative element to each set in the collection.
Given any collection of nonempty sets, there is a way to assign a representative element to each set in the collection.

Every total relation contains a function with the same domain.
Logical Statements

Given any collection of nonempty sets, there is a way to assign a representative element to each set in the collection.

Every total relation contains a function with the same domain.

For any equivalence relation, there is a choice function that picks a representative from each equivalence class.
Given any collection of nonempty sets, there is a way to assign a representative element to each set in the collection. Every total relation on a set contains a function with the same domain. For any equivalence relation, there is a choice function that picks a representative from each equivalence class. Not constructively equivalent!
(\forall x:A. \exists y:B. \varphi(x, y)) \Rightarrow (\exists f:B\to A. \forall x:A. \varphi(x, fx))

\exists f:A/\approx \to A. \forall q:A/\approx. [f(q)]\approx = q
(\forall x : A. \exists y : B. \varphi(x, y)) \Rightarrow (\exists f : B^A. \forall x : A. \varphi(x, fx))
(∀x : A. ∃y : B. \( \varphi(x, y) \)) \Rightarrow (\exists f : B^A. \forall x : A. \varphi(x, fx))

\( \exists f : A/\approx \rightarrow A. \ \forall q : A/\approx. \ [f(q)]_\approx = q \)
Type Theoretical Statements

\[(\forall x : A. \exists y : B. \varphi(x, y)) \Rightarrow (\exists f : B^A. \forall x : A. \varphi(x, fx))\]

\[\exists f : A/\approx \rightarrow A. \forall q : A/\approx. [f(q)]_\approx = q\]
Goal #1:
Provide a computational interpretation of a strong variant of AC
Goal #1:
Provide a computational interpretation of a strong variant of AC

\((A \rightarrow B/\approx)\)
Goal #1:
Provide a computational interpretation of a strong variant of AC

\[(A \rightarrow B/\approx)\]

non-deterministic computation
Goal #1:
Provide a computational interpretation of a strong variant of AC

$(A \rightarrow B/\approx)$  $(A \rightarrow B)$

non-deterministic computation
Goal #1: Provide a computational interpretation of a strong variant of AC

\[(A \rightarrow B / \approx) \rightarrow (A \rightarrow B)\]

- non-deterministic computation
- deterministic computation
Goal #1: Provide a computational interpretation of a strong variant of AC

\[ t : (A \rightarrow B/\approx) \rightarrow (A \rightarrow B) \]

s.t. \( t \) reduces in a manner that reflects a choice function.

non-deterministic computation  \rightarrow  deterministic computation
Implementation Weakening

- Implementation through **memoization**.
- **Stateful** computation.
Implementation Weakening

- Implementation through memoization.
- **Stateful** computation.
- BUT – memoizing non-deterministically generates deterministic functions.
Implementation Weakening

- Implementation through memoization.
- Stateful computation.
- BUT – memoizing non-deterministically generates deterministic functions.

\[(A \rightarrow B/\approx)\]
Implementation Weakening

- Implementation through memoization.
- **Stateful** computation.
- BUT – memoizing non-deterministically generates deterministic functions.

\[(A \rightarrow B / \approx)\]

non-deterministic computation
Implementation Weakening

- Implementation through memoization.
- **Stateful** computation.
- **BUT** – memoizing **non-deterministically** generates deterministic functions.

\[(A \rightarrow B/\approx) \quad (A\rightarrow B/A\rightarrow\approx)\]

non-deterministic computation
Implementation Weakening

- Implementation through memoization.
- Stateful computation.
- BUT – memoizing non-deterministically generates deterministic functions.

\[
\frac{A \rightarrow B/\approx}{(A \rightarrow B/A \rightarrow \approx)}
\]

- non-deterministic computation
- deterministic computation
Implementation Weakening

- Implementation through memoization.
- Stateful computation.
- BUT – memoizing non-deterministically generates deterministic functions.

\[ t : (A \rightarrow B / \equiv) \rightarrow (A \rightarrow B / A \rightarrow \equiv) \]

non-deterministic computation \[\rightarrow\] deterministic computation
- Implementation through \textit{memoization}.
- \textbf{Stateful} computation.
- BUT – memoizing \textit{non-deterministically} generates deterministic functions.

\[
t : (A \rightarrow B/\approx) \rightarrow (A\rightarrow B/A\rightarrow\approx)
\]

\textbf{NDAC}
Constructivism Weakening

Diaconescu’s Theorem

NDAC \rightarrow LEM
Constructivism Weakening

Diaconescu’s Theorem

$NDAC \rightarrow LEM$

$NDCC$

$t : (\mathbb{N} \rightarrow B/\approx) \rightarrow (\mathbb{N} \rightarrow B/_{N \rightarrow \approx})$
Key Implementation Features

Goal #2: Implement NDCC

Main features of the framework:

- **General** framework
  - higher-order abstract syntax
  - models rather than a specific calculus
- **Extensible** – no closed world assumption
- **Robust** w.r.t. (certain) extensions to the underlying calculus
The Effective Topos

**A topos**

- A categorical model of both set theory and type theory.
  - Objects $\sim$ types
  - Morphisms $\sim$ expression
- Cartesian closed – a model of simply-typed $\lambda$-calculus.
- Contains equalizers – an internal notion of equality.
- Exhibit an impredicative type of propositions $\Omega$.
- Models a powerful type theory: dependent subset and quotient types and extensionality of entailment.

**The effective topos ($\mathcal{E}_{ff}$)**

- Has a natural-numbers object
- All functions on the natural numbers are Turing-computable
Constructing the Effective Topos

- Topos $P \leftrightarrow \text{Set}(P)$
- ‘tripos-to-topos’
- Model of HOL $F \leftrightarrow \text{UFam}(F)$
- Evidenced frame
Constructing the Effective Topos

The effective topos

\[ P \mapsto \text{Set}(P) \]

‘tripo-to-topos’

Kleene’s realizability model of HOL

\[ F \mapsto \text{UFam}(F) \]

evidenced frame
An **evidenced frame** is an inhabited set $\Phi$ (propositions), a set $E$ (evidence codes), and an evidence relation $\phi_1 \xrightarrow{e} \phi_2$ s.t.

**Reflexivity**  An evidence code $e_{id} \in E$
- $\phi \xrightarrow{e_{id}} \phi$

**Transitivity**  A binary operator $\cdot; \cdot : E \times E \to E$
- $\phi_1 \xrightarrow{e} \phi_2 \implies \phi_2 \xrightarrow{e'} \phi_3 \implies \phi_1 \xrightarrow{e; e'} \phi_3$

**Conjunction**  $\land : \Phi \times \Phi \to \Phi$, $(\cdot, \cdot) : E \times E \to E$ and $e_{\text{fst}}, e_{\text{snd}} \in E$
- $\phi_1 \land \phi_2 \xrightarrow{e_{\text{fst}}} \phi_1$ ; $\phi_1 \land \phi_2 \xrightarrow{e_{\text{snd}}} \phi_2$
- $\phi' \xrightarrow{e_1} \phi_1 \implies \phi' \xrightarrow{e_2} \phi_2 \implies \phi' \xrightarrow{(e_1, e_2)} \phi_1 \land \phi_2$

**Implication**  $\subset : \Phi \times \Phi \to \Phi$, $\bigcdot \bigcap : E \to E$, and $e_{\text{eval}} \in E$
- $\phi_1 \land \phi_2 \xrightarrow{e} \phi_3 \implies \phi_1 \xrightarrow{[e]} \phi_2 \subset \phi_3$
- $\phi_1 \land (\phi_1 \subset \phi_2) \xrightarrow{e_{\text{eval}}} \phi_2$

**Quantification**  For $\{\phi_i\}_{i \in I}$, propositions $\bigcap_{i \in I} \phi_i$ and $\bigcup_{i \in I} \phi_i$
- $\forall i. \bigcap_{i \in I} \phi_i \xrightarrow{e_{id}} \phi_i$ ; $(\forall i. \phi \xrightarrow{e} \phi_i) \implies \phi \xrightarrow{e} \bigcap_{i \in I} \phi_i$
- $\forall i. \phi_i \xrightarrow{e_{id}} \bigcup_{i \in I} \phi_i$ ; $(\forall i. \phi_i \xrightarrow{e} \phi') \implies \bigcup_{i \in I} \phi_i \xrightarrow{e} \phi'$


\( \mathcal{E}ff \) exhibits NDCC for B iff the choice predicate is provable in \( P \).
$\mathcal{E}ff$ exhibits NDCC for $B$ iff the choice predicate is provable in $P$.

$R$ morphism w.r.t. $\equiv_N$ and $\approx$
NDCC in the Effective Topos

\( \mathcal{E}ff \) exhibits NDCC for \( B \) iff the choice predicate is provable in \( P \).

- \( R \) morphism w.r.t. \( =_N \) and \( \approx \)
- \( S \) determinization of \( R \) w.r.t. \( =_B \)
$\mathcal{E}ff$ exhibits NDCC for $B$ iff the choice predicate is provable in $P$.

$R$ morphism w.r.t. $=_N$ and $\approx$

$S$ determinization of $R$ w.r.t. $=_B$

$\forall R : \mathbb{N} \times B \to \Omega_P$.

- **left-total**
  
  $n =_N n \implies \exists b. n R b$

- **right-unique w.r.t. $\approx$**
  
  $n R b_1 \land n R b_2 \implies b_1 \approx b_2$

- **congruent**
  
  $n_1 =_N n_2 \land b_1 \approx b_2 \land n_1 R$

  $b_1 \implies n_2 R b_2$

- **strict**
  
  $n R b \implies n =_N n \land b \approx b$
**NDCC in the Effective Topos**

$\mathcal{E}ff$ exhibits NDCC for $B$ iff the choice predicate is provable in $P$.

**$R$**

- **morphism w.r.t. $=_{N}$ and $\approx$**

**$S$**

- **determinization of $R$ w.r.t. $=_{B}$**

∀$R : \mathbb{N} \times B \rightarrow \Omega_P$.

  - **left-total**
    
    \[ n =_{N} n \implies \exists b. \ n R b \]

  - **right-unique w.r.t. $\approx$**
    
    \[ n R b_1 \land n R b_2 \implies b_1 \approx b_2 \]

  - **congruent**
    
    \[
    n_1 =_{N} n_2 \land b_1 \approx b_2 \land n_1 R \\
    b_1 \implies n_2 R b_2
    \]

  - **strict**
    
    \[ n R b \implies n =_{N} n \land b \approx b \]

∃$S : \mathbb{N} \times B \rightarrow \Omega_P$.

  - **$R$-inclusion**
    
    \[ n S b \implies n R b \]

  - **left-total**
    
    \[ n =_{N} n \implies \exists b. \ n S b \]

  - **right-unique w.r.t. $=_{B}$**
    
    \[ n S b_1 \land n S b_2 \implies b_1 =_{B} b_2 \]

  - **congruent**
    
    \[
    n_1 =_{N} n_2 \land n_1 S b \implies n_2 S b
    \]
Let $v_{tot}$ be the $\lambda$-value that implements totality of $R$ (extracted from the given evidence).

For each $n$, computing $(v_{tot} \ n_\lambda)$ results in an element $v_n$ of $R_{n,b}$ for some $b$.

For each $n$, pick one such $b$ to be $b_n$.

Define $S_{n,b_n}$ to be the singleton set $\{v_n\}$ if such exists, otherwise let $S_{n,b}$ be empty.
The Hidden Assumption(s) in the Proof of NDCC

- Let $v_{tot}$ be the $\lambda$-value that implements totality of $R$ (extracted from the given evidence).
- For each $n$, computing $(v_{tot} \ n_\lambda)$ results in an element $v_n$ of $R_{n,b}$ for some $b$.
- For each $n$, pick one such $b$ to be $b_n$.
- Define $S_{n,b_n}$ to be the singleton set $\{v_n\}$ if such exists, otherwise let $S_{n,b}$ be empty.

assumes CC in the metatheory
Let $v_{tot}$ be the $\lambda$-value that implements totality of $R$ (extracted from the given evidence).

For each $n$, computing $(v_{tot} \ n_\lambda)$ results in an element $v_n$ of $R_{n,b}$ for some $b$.

For each $n$, pick one such $b$ to be $b_n$.

Define $S_{n,b_n}$ to be the singleton set $\{v_n\}$ if such exists, otherwise let $S_{n,b}$ be empty.

- Assumes CC in the metatheory.
- Right-uniqueness of $S$ relies on $v_{tot}$ being deterministic.
Our approach

the effective topos

\[ P \mapsto \text{Set}(P) \]

\text{‘tripos-to-topos’}

Kleene’s realizability model of HOL

\[ F \mapsto \text{UFam}(F) \]

evidenced frame

enabling internal memoization

Kripke semantics for heaps + Kleene’s realizability for programs

models NDCC
Our approach

A stateful variant of the effective topos

\[ P \mapsto \text{Set}(P) \]

`tripos-to-topos`

Stateful realizability model of HOL

\[ F \mapsto \text{UFam}(F) \]

Stateful evidenced frame
Our approach

- A stateful variant of the effective topos
  - $P \mapsto \text{Set}(P)$
  - ‘tripos-to-topos’

- Stateful realizability model of HOL
  - $F \mapsto \text{UFam}(F)$

- Stateful evidenced frame
  - Enabling internal memoization
Our approach

A stateful variant of the effective topos

\[ P \mapsto \text{Set}(P) \]

‘tripos-to-topos’

Stateful realizability model of HOL

\[ F \mapsto \text{UFam}(F) \]

Stateful evidenced frame

Kripke semantics for heaps + Kleene’s realizability for programs

Enabling internal memoization
Our approach

A stateful variant of the effective topos

\[ P \mapsto \text{Set}(P) \]

'tripos-to-topos'

Stateful realizability model of HOL

\[ F \mapsto \text{UFam}(F) \]

Stateful evidenced frame

Models NDCC

Kripke semantics for heaps +
Kleene's realizability for programs

Enabling internal memoization
Naive stateful evidenced frame:

$h\phi v$ propositions indicate which values in which heaps serve as realizers of $\phi$.

$\phi_1 \xrightarrow{e} \phi_2$ for all $h$ and $v_1$ s.t. $h \phi_1 v_1$: $e$ terminates on $v_1$ under $h$ and returns $v_2$ and results in a modified $h'$ s.t. $h' \phi_2 v_2$. 
Incorporating State

Naive stateful evidenced frame:

$h\phi v$ propositions indicate which values in which heaps serve as realizers of $\phi$.

$\phi_1 \xrightarrow{e} \phi_2$ for all $h$ and $v_1$ s.t. $h \phi_1 v_1$: $e$ terminates on $v_1$ under $h$ and returns $v_2$ and results in a modified $h'$ s.t. $h' \phi_2 v_2$.

- **Problem #1:** sequential pairing and heap modification.
  - $\Rightarrow$ propositions must be preserved by future heaps.
- **Problem #2:** ensuring the memoization function exhibits the required behavior under all potential futures.
  - $\Rightarrow$ propositions must be preserved only by well-formed futures.

- The memoized computation is put into the heap and inputs to are restricted to be $\lambda$-encodings of numbers, so the heap can independently verify the memoized data.
Operational Semantics

While evaluation might modify the heap, we are not concerned with a specific evolvement of the heap, rather all possible futures.
Operational Semantics

While evaluation might modify the heap, we are not concerned with a specific evolvement of the heap, rather all possible futures.

Reduction relation

\[ c \downarrow_h c' \]

coalgebra of certain rules

No modified heap

Termination relation

\[ c \downarrow \]

algebra of certain rules

termination must be preserved by (well-formed) futures

\[ \forall h, h', c. \quad h \preceq \text{wf} h' \land c \downarrow h = \Rightarrow c \downarrow h' \]

Progress: termination under a well-formed heap ensures reducibility under some future heap

\[ \forall h, c. \quad \vdash h \land c \downarrow h = \Rightarrow \exists h', c'. \quad h \preceq \text{wf} h' \land c \downarrow h' c' \]
While evaluation might modify the heap, we are not concerned with a specific evolvement of the heap, rather all possible futures.

**Reduction relation**
\[ c \downarrow_h c' \]
coalgebra of certain rules
No modified heap

**Termination relation**
\[ c \downarrow_h \]
algebra of certain rules

Termination must be preserved by (well-formed) futures:

\[ \forall h, h', c. h \preceq \text{wf} h' \land c \downarrow_h \Rightarrow c \downarrow_h \]

Progress: termination under a well-formed heap ensures reducibility under some future heap:

\[ \forall h, c. \vdash h \land c \downarrow_h \Rightarrow \exists h', c'. h \preceq \text{wf} h' \land c \downarrow_{h'} c' \]
While evaluation might modify the heap, we are not concerned with a specific evolvement of the heap, rather all possible futures.

**Reduction relation**

\[ c \downarrow_h c' \]

coalgebra of certain rules

No modified heap

- termination must be preserved by (well-formed) futures

\[ \forall h, h', c. \; h \preceq_{wf} h' \land c \downarrow_h \implies c \downarrow_{h'} \]

- Progress: termination under a well-formed heap ensures reducibility under some future heap

\[ \forall h, c. \; \vdash h \land c \downarrow_h \implies \exists h', c'. \; h \preceq_{wf} h' \land c \downarrow_{h'} c' \]
Stateful Evidenced Frame

$h \vdash \phi_1 \xrightarrow{e} \phi_2$: $e$ is evidence in heap $h$ that $\phi_1$ implies $\phi_2$

$\forall c_1. \ h \phi_1 c_1 \implies (e \downarrow_h c_1 \land \forall c_2. \ e \downarrow_h c_2 \implies h \phi_2 c_2)$

**Propositions** Relations $\phi$ between heaps and codes s.t.

$\forall h, c. \ h \phi c \implies \text{val}(c) \land \forall h'. \ h \preceq_{\text{wf}} h' \implies h' \phi c$

**Codes** Syntactically-encodable functions $e : C \to C$.

**Evidence** $\phi_1 \xrightarrow{e} \phi_2$: $\forall h. \ h \vdash h \implies h \vdash \phi_1 \xrightarrow{e} \phi_2$.

$h (\phi_1 \land \phi_2) c \exists c_1, c_2. \ c = \text{pair} c_1 c_2 \land h \phi_1 c_1 \land h \phi_2 c_2$

$h (\phi_1 \subset \phi_2) c \exists e. \ c = \text{lambda} \ e \land \forall h'. \ h \preceq_{\text{wf}} h' \implies h' \vdash \phi_1 \xrightarrow{e} \phi_2$

$h \bigcap_{i \in I} \phi_i c \forall i. \ h \phi_i c$

$h \bigcup_{i \in I} \phi_i c \exists i. \ h \phi_i c$
Stateful Evidenced Frame

$h \vdash \phi_1 \xrightarrow{e} \phi_2$: e is evidence in heap $h$ that $\phi_1$ implies $\phi_2$

$\forall c_1. h \phi_1 c_1 \implies (e c_1 \downarrow_h \land \forall c_2. e c_1 \downarrow_h c_2 \implies h \phi_2 c_2)$

**Propositions** Relations $\phi$ between heaps and codes s.t.

$\forall h, c. h \phi c \implies \text{val}(c) \land \forall h'. h \preceq_{wf} h' \implies h' \phi c$

**Codes** Syntactically-encodable functions $e : C \rightarrow C$.

**Evidence** $\phi_1 \xrightarrow{e} \phi_2$: $\forall h. h \vdash h \implies h \vdash \phi_1 \xrightarrow{e} \phi_2$.

$h (\phi_1 \land \phi_2) c \exists c_1, c_2. c = \text{pair} c_1, c_2 \land h \phi_1 c_1 \land h \phi_2 c_2$

$h (\phi_1 \subset \phi_2) c \exists e. c = \text{lambda} e \land \forall h'. h \preceq_{wf} h' \implies h' \vdash \phi_1 \xrightarrow{e} \phi_2$

$h \bigcap_{i \in I} \phi_i c \forall i. h \phi_i c$

$h \bigcup_{i \in I} \phi_i c \exists i. h \phi_i c$

consistent
The extended code language:

**alloc** allocation of a new memoization table in the heap.

**lookup** retrieval of a value at a specific index in the memoization table in the heap.

$h@\ell \rightarrow c_f$ location $\ell$ is allocated to the generator function $c_f$ in $h$.

$n \rightarrow c$ in the memoization table at location $\ell$ in $h$, the input $n$ has been memoized to $c$.

- Allocated locations are preserved by futures and have a unique generator function.
- Memoized entries are preserved by futures and are unique.
- Memoized entries agree with the generator function associated with the allocated location.
Proof of NDCC

\[ R \quad \text{morphism w.r.t. } \equiv_N \text{ and } \approx \]

\[ S \quad \text{determinization of } R \text{ w.r.t. } \equiv_B \]
Proof of NDCC

- Allocate a new memory location \( \ell \) in \( h \) whose generator function is the evidence that \( R \) is left-total.
- Define \( S \) s.t. \( c \) is evidence of \( S_{n,b} \) under heap \( h' \) whenever

\[
\begin{align*}
  n \xrightarrow{h'@\ell} c \land h \preceq_{\text{wf}} h' \land b = \text{Choice}_R(n, c, \cdots) \text{ holds}.
\end{align*}
\]
Proof of NDCC

- Allocate a new memory location \( \ell \) in \( h \) whose generator function is the evidence that \( R \) is left-total.
- Define \( S \) s.t. \( c \) is evidence of \( S_{n,b} \) under heap \( h' \) whenever
  \[
  n \xrightarrow{h'@\ell} c \land h \preceq_{\text{wf}} h' \land b = \text{Choice}_R(n, c, \cdots) \text{ holds}.
  \]

\[
\lambda\langle x_{\text{tot}}, x_{ru}, x_{\text{cong}}, x_{\text{str}} \rangle. \quad \text{let } \ell := \text{new\_table } x_{\text{tot}} \text{ in }
\]

- R-inclusion
- totality
- right-unique
- congruent

\[
\begin{align*}
\lambda x_s. x_s, \\
\lambda x_n. \ell[x_n], \\
\lambda\langle x_s, \_ \rangle. \text{fst}(c_{\text{str}}(\text{snd}(x_{\text{str}} x_s))), \\
\lambda\langle \_, x_s \rangle. x_s
\end{align*}
\]

\[
\text{evidence of the strictness of } \approx \text{ w.r.t. } =_B
\]
Future Work

- Eliminate the **metatheoretic assumptions**.
- Implement **stronger variants** of the AC:
  - Non-Deterministic Countable Choice.
  - Choice for any set with decidable equality
- Explore other **applications** of stateful evidenced frames.
  - By storing partially-constructed graphs of numbers, one could create a model in which all countable connected graphs have a spanning tree.
  - A constructive variant of Zorn’s Lemma.
Future Work

- Eliminate the *metatheoretic assumptions*.
- Implement stronger variants of the AC:
  - Non-Deterministic Countable Choice.
  - Choice for any set with decidable equality.
- Explore other applications of stateful evidenced frames.
  - By storing partially-constructed graphs of numbers, one could create a model in which all countable connected graphs have a spanning tree.
  - A constructive variant of Zorn’s Lemma.

Thank you!