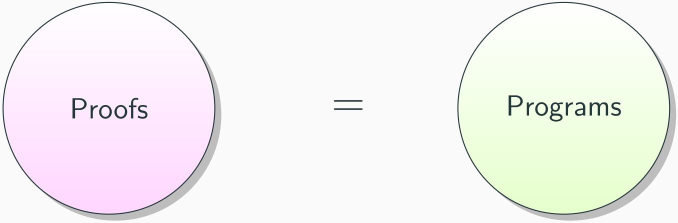
A robust implementation of Axioms of Choice

Liron Cohen

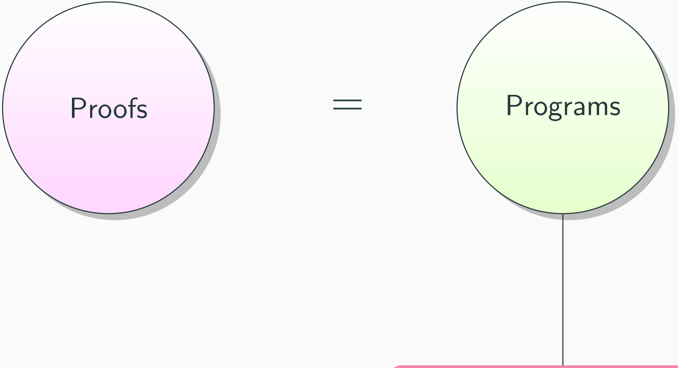
Cornell University, Ithaca, NY, USA

Extending the Proofs-as-Programs Paradigm

Extending the Proofs-as-Programs Paradigm



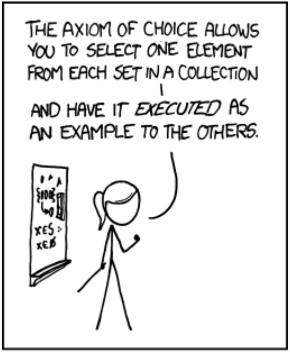
Extending the Proofs-as-Programs Paradigm



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



MY MATH TEACHER WAS A BIG BELIEVER IN PROOF BY INTIMIDATION.

Motivation

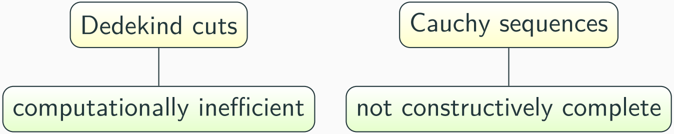
- AC unifies standard constructive representations of the reals.

Dedekind cuts

Cauchy sequences

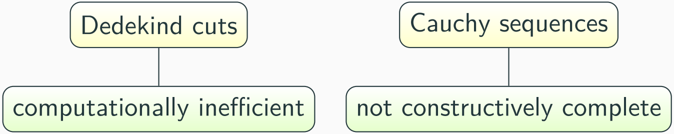
Motivation

- AC unifies standard constructive representations of the reals.



Motivation

- AC unifies standard constructive representations of the reals.



- Unclear status in constructivism.
  - Some variants are considered trivially true due to the specific interpretation of the type constructors Σ and Π.
  - 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

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

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

Logical Statements

Given any collection of … sets, there is a way to assign a representativ… set in the collection

Every total relation c… ith the same domain

For any equivalence re… ere is a choice function that picks a representative from each equivalence class

Not constructively equivalent!

Type Theoretical Statements

???

Type Theoretical Statements

???

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

Type Theoretical Statements

???

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

$$\exists f : {}^{A}/_{\approx} \to A.\ \forall q : {}^{A}/_{\approx}.\ [f(q)]_{\approx} = q$$

Type Theoretical Statements

???

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

$$\exists f : {}^{A}/_{\approx} \to A.\ \forall q : {}^{A}/_{\approx}.\ [f(q)]_{\approx} = q$$

Computational Interpretation of AC

Goal #1:

Provide a computational interpretation of a strong variant of AC

Computational Interpretation of AC

Goal #1:
Provide a computational interpretation of a strong variant of AC

$(A \to B/\approx)$

Computational Interpretation of AC

Goal #1:
Provide a computational interpretation of a strong variant of AC

$(A \to {}^{B}/_{\approx})$

non-deterministic computation

Computational Interpretation of AC

Goal #1:
Provide a computational interpretation of a strong variant of AC

$(A \to {}^{B}/_{\approx}) \qquad (A \to B)$

non-deterministic computation

Computational Interpretation of AC

Goal #1:
Provide a computational interpretation of a strong variant of AC

$(A \to {}^{B}/_{\approx})$ $(A \to B)$

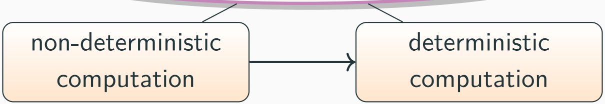
non-deterministic computation

deterministic computation

Computational Interpretation of AC

Goal #1:
Provide a computational interpretation of a strong variant of AC

$$t : (A \to {}^{B}/_{\approx}) \to (A \to B)$$



s.t. t reduces in a manner that reflects a choice function.

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.

$$(A \to {}^{B}/_{\approx})$$

Implementation Weakening

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

$$(A \to {}^{B}/_{\approx})$$

non-deterministic computation

Implementation Weakening

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

$$\left(A \to {}^{B}\!/\!_{\approx}\right) \qquad \left({}^{A\to B}\!/\!_{A\to\approx}\right)$$

non-deterministic computation

Implementation Weakening

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

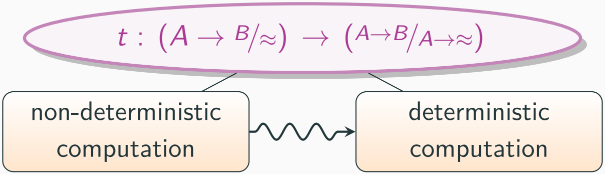
$(A \to {}^{B}/_{\approx})$ $\qquad$ $({}^{A \to B}/_{A \to \approx})$

non-deterministic computation

deterministic computation

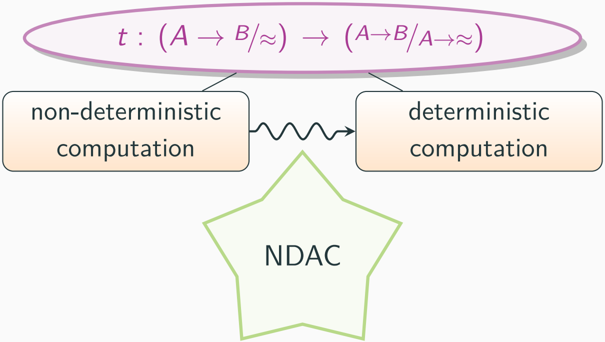
Implementation Weakening

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

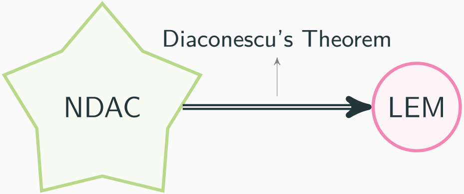


Implementation Weakening

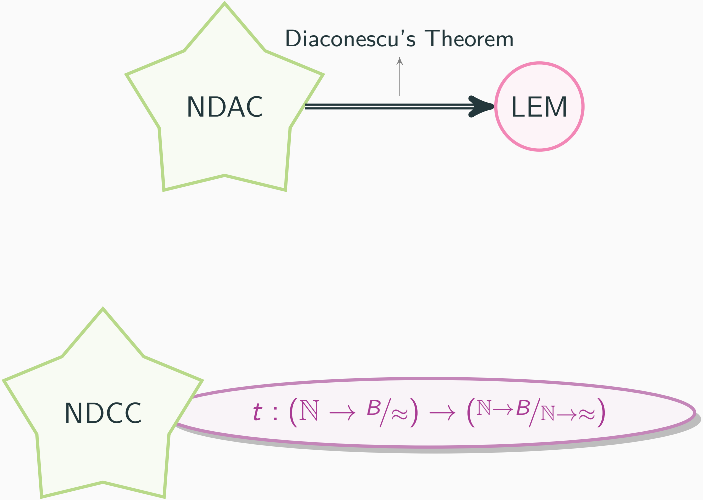
- Implementation through memoization.
- Stateful computation.
- BUT – memoizing non-deterministically generates deterministic functions.



Constructivism Weakening



Constructivism Weakening



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

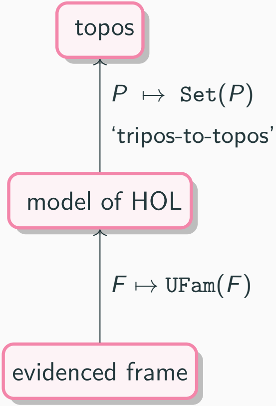
A topos

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

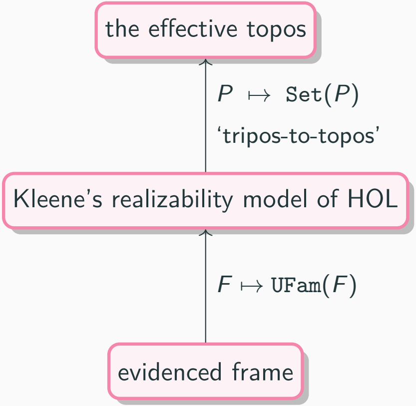
The effective topos ($\mathcal{E}\mathit{ff}$)

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

Constructing the Effective Topos



Constructing the Effective Topos



Evidenced Frame

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

Reflexivity An evidence code $e_{\texttt{id}} \in E$

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

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 $\wedge : \Phi \times \Phi \to \Phi$, $(\!|\cdot,\cdot|\!) : E \times E \to E$ and $e_{\texttt{fst}}, e_{\texttt{snd}} \in E$

- $\phi_1 \wedge \phi_2 \xrightarrow{e_{\texttt{fst}}} \phi_1 \quad ; \quad \phi_1 \wedge \phi_2 \xrightarrow{e_{\texttt{snd}}} \phi_2$
- $\phi' \xrightarrow{e_1} \phi_1 \implies \phi' \xrightarrow{e_2} \phi_2 \implies \phi' \xrightarrow{(\!|e_1,e_2|\!)} \phi_1 \wedge \phi_2$

Implication $\supset : \Phi \times \Phi \to \Phi$, $\wr \cdot \int : E \to E$, and $e_{\texttt{eval}} \in E$

- $\phi_1 \wedge \phi_2 \xrightarrow{e} \phi_3 \implies \phi_1 \xrightarrow{\wr e \int} \phi_2 \supset \phi_3$
- $\phi_1 \wedge (\phi_1 \supset \phi_2) \xrightarrow{e_{\texttt{eval}}} \phi_2$

Quantification For $\{\phi_i\}_{i\in I}$, propositions $\bigcap_{i\in I} \phi_i$ and $\bigcup_{i\in I} \phi_i$

- $\forall i.\ \bigcap_{i\in I} \phi_i \xrightarrow{e_{\texttt{id}}} \phi_i \quad ; \quad (\forall i.\ \phi \xrightarrow{e} \phi_i) \implies \phi \xrightarrow{e} \bigcap_{i\in I} \phi_i$
- $\forall i.\ \phi_i \xrightarrow{e_{\texttt{id}}} \bigcup_{i\in I} \phi_i \quad ; \quad (\forall i.\ \phi_i \xrightarrow{e} \phi') \implies \bigcup_{i\in I} \phi_i \xrightarrow{e} \phi'$

NDCC in the Effective Topos

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

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. $=_{\mathbb{N}}$ and $\approx$

NDCC in the Effective Topos

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

R
morphism w.r.t. $=_{\mathbb{N}}$ and $\approx$

→

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

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. $=_{\mathbb{N}}$ and $\approx$

$\longrightarrow$

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

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

- **left-total**
  $n =_{\mathbb{N}} n \implies \exists b.\ n\ R\ b$
- **right-unique w.r.t.** $\approx$
  $n\ R\ b_1 \wedge n\ R\ b_2 \implies b_1 \approx b_2$
- **congruent**
  $n_1 =_{\mathbb{N}} n_2 \wedge b_1 \approx b_2 \wedge n_1\ R\ b_1 \implies n_2\ R\ b_2$
- **strict**
  $n\ R\ b \implies n =_{\mathbb{N}} n \wedge b \approx b$

NDCC in the Effective Topos

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

R
morphism w.r.t. $=_{\mathbb{N}}$ and $\approx$

→

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

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

- **left-total**
  $n =_{\mathbb{N}} n \implies \exists b.\ n\ R\ b$
- **right-unique w.r.t.** $\approx$
  $n\ R\ b_1 \wedge n\ R\ b_2 \implies b_1 \approx b_2$
- **congruent**
  $n_1 =_{\mathbb{N}} n_2 \wedge b_1 \approx b_2 \wedge n_1\ R\ b_1 \implies n_2\ R\ b_2$
- **strict**
  $n\ R\ b \implies n =_{\mathbb{N}} n \wedge b \approx b$

→

$\exists S : \mathbb{N} \times B \to \Omega_P.$

- R**-inclusion**
  $n\ S\ b \implies n\ R\ b$
- **left-total**
  $n =_{\mathbb{N}} n \implies \exists b.\ n\ S\ b$
- **right-unique w.r.t.** $=_{\mathbf{B}}$
  $n\ S\ b_1 \wedge n\ S\ b_2 \implies b_1 =_B b_2$
- **congruent**
  $n_1 =_{\mathbb{N}} n_2 \wedge n_1\ S\ b \implies n_2\ S\ b$

The Hidden Assumption(s) in the Proof of NDCC

- Let v_{tot} be the λ-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 λ-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

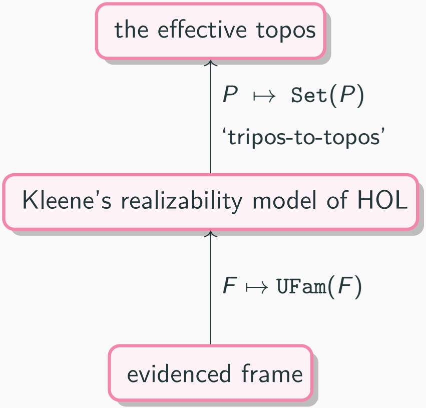
The Hidden Assumption(s) in the Proof of NDCC

- Let v_{tot} be the λ-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



Our approach

a stateful variant of the effective topos

↑ $P \mapsto \mathtt{Set}(P)$

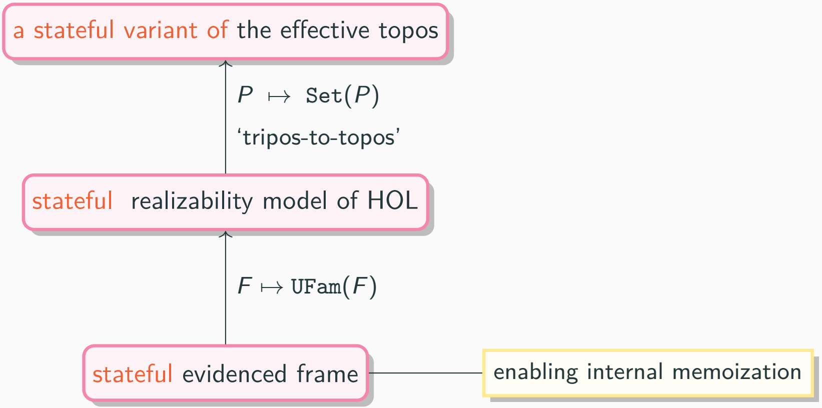
'tripos-to-topos'

stateful realizability model of HOL

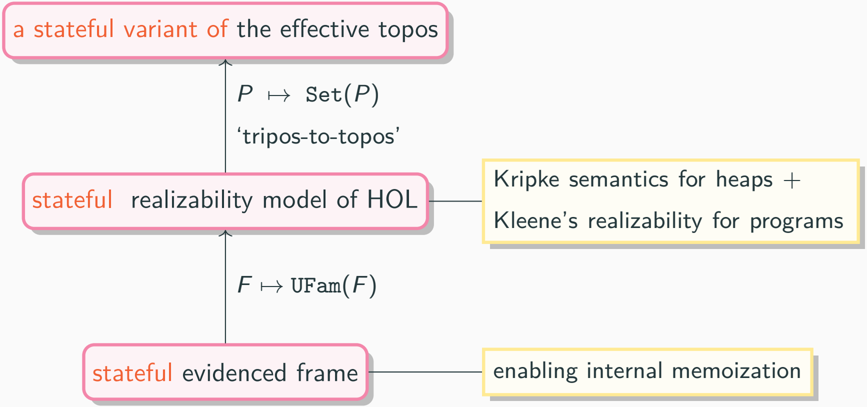
↑ $F \mapsto \mathtt{UFam}(F)$

stateful evidenced frame

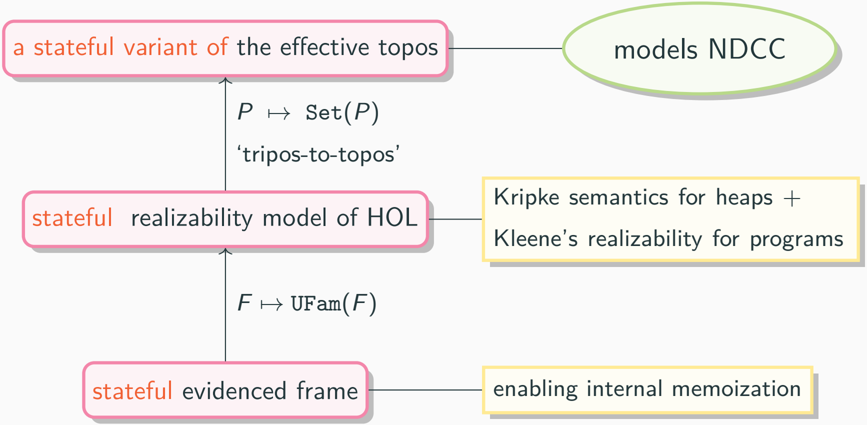
Our approach



Our approach



Our approach



Incorporating State

Naive stateful evidenced frame:

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

$\phi_1 \xrightarrow{e}\!\!\!\rightarrow \phi_2$ 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 ϕ.

$\phi_1 \xrightarrow{e} \phi_2$ 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 λ-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

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

$c \downarrow_h$

algebra of certain rules

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	Termination relation
$c \downarrow_h c'$	$c \downarrow_h$
coalgebra of certain rules	algebra of certain rules
No modified heap	

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

$$\forall h, h', c.\ h \preceq_{wf} h' \wedge c \downarrow_h \implies c \downarrow_{h'}$$

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

$$\forall h, c. \vdash h \wedge c \downarrow_h \implies \exists h', c'.\ h \preceq_{wf} h' \wedge c \downarrow_{h'} c'$$

Stateful Evidenced Frame

$h \vdash \phi_1 \overset{e}{\twoheadrightarrow} \phi_2$: e is evidence in heap h that ϕ_1 implies ϕ_2

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

Propositions Relations ϕ between heaps and codes s.t. $\forall h, c.\ h\ \phi\ c \implies \mathtt{val}(c) \wedge \forall h'.\ h \preceq_{wf} h' \implies h'\ \phi\ c$

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

Evidence $\phi_1 \overset{e}{\twoheadrightarrow} \phi_2$: $\quad \forall h.\ \vdash h \implies h \vdash \phi_1 \overset{e}{\twoheadrightarrow} \phi_2$.

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

h $(\phi_1 \supset \phi_2)$ c $\exists e.\ c = \mathtt{lambda}\ e \ \wedge\ \forall h'.\ h \preceq_{wf} h' \Rightarrow h' \vdash \phi_1 \overset{e}{\twoheadrightarrow} \phi_2$

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

$h \vdash \phi_1 \xrightarrow{e}\!\!\!\!\rightarrow \phi_2$: e is evidence in heap h that ϕ_1 implies ϕ_2

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

Propositions Relations ϕ between heaps and codes s.t. $\forall h, c.\ h\ \phi\ c \implies \texttt{val}(c) \wedge \forall h'.\ h \preceq_{wf} h' \implies h'\ \phi\ c$

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

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

$\mathbf{h\ (\phi_1 \wedge \phi_2)\ c}$ $\exists c_1, c_2.\ c = \texttt{pair}\ c_1\ c_2\ \wedge\ h\ \phi_1\ c_1\ \wedge\ h\ \phi_2\ c_2$

$\mathbf{h\ (\phi_1 \supset \phi_2)\ c}$ $\exists e.\ c = \texttt{lambda}\ e\ \wedge\ \forall h'.\ h \preceq_{wf} h' \Rightarrow h' \vdash \phi_1 \xrightarrow{e}\!\!\!\!\rightarrow \phi_2$

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

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

consistent

Extending the Framework

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.

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

$n \overset{h@\ell}{\rightharpoonup} c$ in the memoization table at location ℓ 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
morphism w.r.t. $=_{\mathbb{N}}$ and $\approx$

→

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

Proof of NDCC

R

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

$\longrightarrow$

S

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

- Allocate a new memory location ℓ 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 $n \overset{h'@\ell}{\rightarrowtriangle} c \wedge h \preceq_{wf} h' \wedge b = Choice_R(n, c, \cdots)$ holds.

Proof of NDCC

R
morphism w.r.t. $=_{\mathbb{N}}$ and $\approx$

→

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

- Allocate a new memory location ℓ 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 $n \overset{h'@\ell}{\rightarrow} c \wedge h \preceq_{wf} h' \wedge b = Choice_R(n, c, \cdots)$ holds.

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

$$\left\langle \begin{array}{l} \lambda x_s.\, x_s, \\ \lambda x_n.\, \ell[x_n], \\ \lambda\langle x_s, \_\rangle.\, \texttt{fst}\,(c_{str}\,(\texttt{snd}\,(x_{str}\, x_s))), \\ \lambda\langle \_, x_s\rangle.\, x_s \end{array} \right\rangle$$

R-inclusion — $\lambda x_s.\, x_s$

totality — $\lambda x_n.\, \ell[x_n]$

right-unique — $\lambda\langle x_s, \_\rangle.\, \texttt{fst}\,(c_{str}\,(\texttt{snd}\,(x_{str}\, x_s)))$

congruent — $\lambda\langle \_, x_s\rangle.\, x_s$

c_{str}: evidence of the strictness of $\approx$ 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!