Classical realizability: new tools and applications

Guillaume Geoffroy

Under the supervision of Laurent Regnier
I2M, ED 184, Aix-Marseille Université

29 March 2019

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)

There are infinitely many prime numbers.

Proof.

$\square$

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)

There are infinitely many prime numbers.

Proof.

Take any finite list $p_1, \ldots, p_n$ of prime numbers

□

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)

There are infinitely many prime numbers.

Proof.

`Take any finite list` $p_1, \ldots, p_n$ `of prime numbers`
`Compute their product` $A = p_1 \times \ldots \times p_n$

□

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)
There are infinitely many prime numbers.

Proof.

Take any finite list $p_1, \ldots, p_n$ of prime numbers
Compute their product $A = p_1 \times \ldots \times p_n$
Extract a prime factor q from $A + 1$

$\square$

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)

There are infinitely many prime numbers.

Proof.

Take any finite list $p_1, \ldots, p_n$ of prime numbers
Compute their product $A = p_1 \times \ldots \times p_n$
Extract a prime factor q from $A + 1$
q divides $A + 1$, therefore q does not divide A

$\square$

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)
There are infinitely many prime numbers.

Proof.

Take any finite list $p_1, \ldots, p_n$ of prime numbers
Compute their product $A = p_1 \times \ldots \times p_n$
Extract a prime factor q from $A + 1$
q divides $A + 1$, therefore q does not divide A
Therefore q is not in the list $p_1, \ldots, p_n$

□

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)

There are infinitely many prime numbers.

Proof.

```
Take any finite list  3, 5, 7  of prime numbers
Compute their product A = p_1 × ... × p_n
Extract a prime factor q from A + 1
q divides A + 1, therefore q does not divide A
Therefore q is not in the list p_1, ..., p_n
```

□

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)

There are infinitely many prime numbers.

Proof.

```
Take any finite list  3,5,7  of prime numbers
Compute their product A = 3 × 5 × 7
Extract a prime factor q from A + 1
q divides A + 1, therefore q does not divide A
Therefore q is not in the list p_1, ..., p_n
```

□

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)
There are infinitely many prime numbers.

Proof.

`Take any finite list` $3, 5, 7$ `of prime numbers`
`Compute their product` $A = 3 \times 5 \times 7 = 105$
`Extract a prime factor` q `from` $A + 1$
q `divides` $A + 1$, `therefore` q `does not divide` A
`Therefore` q `is not in the list` $p_1, \ldots, p_n$

$\square$

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)
There are infinitely many prime numbers.

Proof.

`Take any finite list` $3, 5, 7$ `of prime numbers`
`Compute their product` $A = 3 \times 5 \times 7 = 105$
`Extract a prime factor` q `from` 106
q `divides` $A + 1$`, therefore` q `does not divide` A
`Therefore` q `is not in the list` $p_1, \ldots, p_n$

$\square$

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)

There are infinitely many prime numbers.

Proof.

```
Take any finite list  3,5,7  of prime numbers
Compute their product A = 3 × 5 × 7 = 105
Extract a prime factor q from 106 = 2 × 53
q divides A + 1, therefore q does not divide A
Therefore q is not in the list p_1, ..., p_n
```

□

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)

There are infinitely many prime numbers.

Proof.

```
Take any finite list  3,5,7  of prime numbers
Compute their product A = 3 × 5 × 7 = 105
Extract a prime factor 2 from 106 = 2 × 53
q divides A + 1, therefore q does not divide A
Therefore q is not in the list p_1, ..., p_n
```

$\square$

Realizability & the proof-as-program correspondance

Theorem (Euclid – *Elements*, Book IX – c. 300 BC)

There are infinitely many prime numbers.

Proof.

`Take any finite list` $3, 5, 7$ `of prime numbers`
`Compute their product` $A = 3 \times 5 \times 7 = 105$
`Extract a prime factor 2 from` $106 = 2 \times 53$
2 `divides` 106`, therefore` 2 `does not divide` 105
`Therefore` 2 `is not in the list` $3, 5, 7$

$\square$

A program is not (always) a proof

Conjecture (Goldbach, 1742)

For each $n \geq 4$ even, there exist p, q primes such that $n = p + q$.

A program is not (always) a proof

Conjecture (Goldbach, 1742)
For each $n \geq 4$ even, there exist p, q primes such that $n = p + q$.

Realizing the conjecture $\leftrightarrow$ finding p and q for every n:

A program is not (always) a proof

Conjecture (Goldbach, 1742)

For each $n \geq 4$ even, there exist p, q primes such that $n = p + q$.

Realizing the conjecture $\leftrightarrow$ finding p and q for every n:

A program is not (always) a proof

Conjecture (Goldbach, 1742)

For each $n \geq 4$ even, there exist p, q primes such that $n = p + q$.

Realizing the conjecture $\leftrightarrow$ finding p and q for every n:

```
Take any even integer n ≥ 4.
```

A program is not (always) a proof

Conjecture (Goldbach, 1742)

For each $n \geq 4$ even, there exist p, q primes such that $n = p + q$.

Realizing the conjecture $\leftrightarrow$ finding p and q for every n:

```
Take any even integer n ≥ 4.
For all pairs of primes (p, q) do:
```

A program is not (always) a proof

Conjecture (Goldbach, 1742)

For each $n \geq 4$ even, there exist p, q primes such that $n = p + q$.

Realizing the conjecture $\leftrightarrow$ finding p and q for every n:

```
Take any even integer n ≥ 4.
For all pairs of primes (p, q) do:
   if n = p + q:  return (p, q)
```

A program is not (always) a proof

Conjecture (Goldbach, 1742)
For each $n \geq 4$ even, there exist p, q primes such that $n = p + q$.

Realizing the conjecture $\leftrightarrow$ finding p and q for every n:

```
Take any even integer n ≥ 4.
For all pairs of primes (p, q) do:
   if n = p + q:  return (p, q)
   else:  keep looking
```

Krivine's classical realizability

$$\begin{array}{ccc} \text{Model of computation} & & \text{Realizability theory} \\ (\text{programs } t, u, v, \ldots) & \longrightarrow & (\text{formulas } A, B, C, \ldots) \end{array}$$

Krivine's classical realizability

$$\begin{array}{ccc} \text{Model of computation} & \xrightarrow[\substack{t \Vdash A \\ t \text{ realizes } A}]{} & \text{Realizability theory} \\ (\text{programs } t, u, v, \ldots) & & \{\ A;\ \exists t\ (t \Vdash A)\ \} \end{array}$$

Krivine's classical realizability

$$\begin{array}{ccc} \text{Model of computation} & & \text{Realizability theory} \\ (\text{programs } t, u, v, \ldots) & \xrightarrow[\substack{t \Vdash A \\ t \text{ realizes } A}]{} & \{\ A;\ \exists t\ (t \Vdash A)\ \} \end{array}$$

$$\begin{array}{ccc} \text{Extension of } \lambda\text{-calculus} & & \\ + \text{ control (call-cc)} & \longrightarrow & \text{Extension of ZF} \end{array}$$

Krivine's classical realizability

$$\begin{array}{ccc} \text{Model of computation} & & \text{Realizability theory} \\ \text{(programs } t, u, v, \ldots) & \xrightarrow[\substack{t \Vdash A \\ t \text{ realizes } A}]{} & \{\ A;\ \exists t\ (t \Vdash A)\ \} \end{array}$$

$$\begin{array}{ccc} \text{Extension of } \lambda\text{-calculus} & & \\ +\ \text{control (call-cc)} & \longrightarrow & \text{Extension of ZF} \\ & & \downarrow \\ & & \text{Realizability models} \\ & & \text{of ZF} \end{array}$$

Krivine's classical realizability

$$
\begin{tikzcd}
\begin{array}{c}\text{Model of computation}\\ \text{(programs } t, u, v, \ldots\text{)}\end{array} \arrow[r, "{\begin{array}{c} t \Vdash A \\ t \text{ realizes } A\end{array}}"'] & \begin{array}{c}\text{Realizability theory}\\ \{\ A;\ \exists t\ (t \Vdash A)\ \}\end{array}
\end{tikzcd}
$$

$$
\begin{tikzcd}
\begin{array}{c}\text{Extension of } \lambda\text{-calculus}\\ + \text{ control (call-cc)}\end{array} \arrow[r] & \text{Extension of ZF} \arrow[d] \\
\text{Forcing poset} \arrow[u] & \begin{array}{c}\text{Realizability models}\\ \text{of ZF}\end{array}
\end{tikzcd}
$$

Krivine's realizability models

Krivine's realizability models: formulas

ZF_ε	$=$	ε	$\subseteq$

Krivine's realizability models: formulas

ZF_ε	$=$	ε	$\subseteq$
ZF	$\approx$	$\in$	$\subsetapprox$

Krivine's realizability models: formulas

ZF_ε	$=$	ε	$\subseteq$
ZF	$\approx$	$\in$	$\subsetapprox$

$$\mathsf{ZF}_\varepsilon \vdash \forall a, b \ \ (a \in b \leftrightarrow \exists a' \approx a \ a' \varepsilon b)$$

Krivine's realizability models: formulas

ZF_ε	$=$	ε	$\subseteq$
ZF	$\approx$	$\in$	$\precsim$

$$ZF_\varepsilon \vdash \forall a, b \ \ (a{\in}b \leftrightarrow \exists a'{\approx}a \ a'\varepsilon b)$$

Proposition: ZF_ε is a conservative extension of ZF

Krivine's realizability models: formulas

ZF_ε	$=$	ε	$\subseteq$
ZF	$\approx$	$\in$	$\subsetapprox$

$$ZF_\varepsilon \vdash \forall a, b \ \left(a \in b \leftrightarrow \exists a' \approx a \ a' \varepsilon b\right)$$

Proposition: ZF_ε is a conservative extension of ZF

Logical	$\forall$	$\wedge$	$\vee$	$\rightarrow$	$\ldots$
Non-logical	$\cap$	$\cup$	$\hookrightarrow$	$\ldots$	

Krivine's realizability models: programs & evaluation

$$t \star \pi$$

Krivine's realizability models: programs & evaluation

Evaluation rules:

$$\begin{array}{rcll} tu \star \pi & \succ_K & t \star u \cdot \pi & \text{(Push)} \\ \lambda x.\, t \star u \cdot \pi & \succ_K & t[x := u] \star \pi & \text{(Grab)} \end{array}$$

Krivine's realizability models: programs & evaluation

Evaluation rules:

$$
\begin{array}{rcll}
tu \star \pi & \succ_K & t \star u \cdot \pi & \text{(Push)} \\
\lambda x.\, t \star u \cdot \pi & \succ_K & t[x := u] \star \pi & \text{(Grab)} \\
\text{call-cc} \star t \cdot \pi & \succ_K & t \star k_\pi \cdot \pi & \text{(Save)} \\
k_\pi \star t \cdot \pi' & \succ_K & t \star \pi & \text{(Restore)}
\end{array}
$$

Krivine's realizability models: programs & evaluation

Evaluation rules:

$$
\begin{array}{rclr}
tu \star \pi & \succ_K & t \star u \cdot \pi & \text{(Push)} \\
\lambda x.\ t \star u \cdot \pi & \succ_K & t[x := u] \star \pi & \text{(Grab)} \\
\text{call-cc} \star t \cdot \pi & \succ_K & t \star k_\pi \cdot \pi & \text{(Save)} \\
k_\pi \star t \cdot \pi' & \succ_K & t \star \pi & \text{(Restore)}
\end{array}
$$

Plus transitivity and reflexivity.

Krivine's realizability models: poles

Pole: a set $\perp\!\!\!\perp$ of processes such that for all $q \in \perp\!\!\!\perp$, for all $p \succ_K q$, $p \in \perp\!\!\!\perp$.

Krivine's realizability models: poles

Pole: a set $\perp\!\!\!\perp$ of processes such that for all $q \in \perp\!\!\!\perp$, for all $p \succ_K q$, $p \in \perp\!\!\!\perp$. Example: $\perp\!\!\!\perp = \{\ p;\ p \succ_K \lambda x.\ x \star \mathsf{nil}\ \}$.

Krivine's realizability models: poles

Pole: a set $\perp\!\!\!\perp$ of processes such that for all $q \in \perp\!\!\!\perp$, for all $p \succ_K q$, $p \in \perp\!\!\!\perp$. Example: $\perp\!\!\!\perp = \{\ p;\ p \succ_K \lambda x.\ x \star \mathsf{nil}\ \}$.

$$\text{Formula } A + \text{pole } \perp\!\!\!\perp \rightsquigarrow \text{Falsity value } \|A\|_{\perp\!\!\!\perp} \subseteq \Pi = \{\text{stacks}\}$$

$$\text{Formula } A + \text{pole } \perp\!\!\!\perp \rightsquigarrow \text{Truth value } |A|_{\perp\!\!\!\perp} \subseteq \Lambda = \{\text{terms}\}$$

Krivine's realizability models: poles

Pole: a set $\perp\!\!\!\perp$ of processes such that for all $q \in \perp\!\!\!\perp$, for all $p \succ_K q$, $p \in \perp\!\!\!\perp$. Example: $\perp\!\!\!\perp = \{\ p;\ p \succ_K \lambda x.\ x \star \mathrm{nil}\ \}$.

$$\text{Formula } A + \text{pole } \perp\!\!\!\perp \rightsquigarrow \text{Falsity value } \|A\|_{\perp\!\!\!\perp} \subseteq \Pi = \{\text{stacks}\}$$

$$\text{Formula } A + \text{pole } \perp\!\!\!\perp \rightsquigarrow \text{Truth value } |A|_{\perp\!\!\!\perp} \subseteq \Lambda = \{\text{terms}\}$$

$$|A|_{\perp\!\!\!\perp} = (\|A\|_{\perp\!\!\!\perp})^{\perp\!\!\!\perp} = \{\ t;\ \forall \pi \in \|A\|_{\perp\!\!\!\perp}, t \star \pi \in \perp\!\!\!\perp\ \}$$

Krivine's realizability models: single-pole models

$$\text{All poles} \longrightarrow \text{Extensions of ZF}$$

$$\perp\!\!\!\perp \longmapsto \text{Theory}(\perp\!\!\!\perp)$$

Krivine's realizability models: single-pole models

$$\text{All poles} \longrightarrow \text{Extensions of ZF}$$

$$\perp\!\!\!\perp \longmapsto \text{Theory}(\perp\!\!\!\perp) =$$

$$\{\ A;\ A \text{ is realized wrt } \perp\!\!\!\perp\ \}$$

Krivine's realizability models: single-pole models

$$\text{All poles} \longrightarrow \text{Extensions of ZF}$$

$$\perp\!\!\!\perp \longmapsto \text{Theory}(\perp\!\!\!\perp) = \{\ A;\ \exists t \text{ proof-like}, t \in |A|_{\perp\!\!\!\perp}\ \}$$

Krivine's realizability models: single-pole models

$$\text{All poles} \longrightarrow \text{Extensions of ZF}$$

$$\perp\!\!\!\perp \longmapsto \text{Theory}(\perp\!\!\!\perp) = \{\ A;\ \exists t \text{ proof-like}, t \in |A|_{\perp\!\!\!\perp}\ \}$$

Proof-like term: a term in which no stack constant (k_π) appears.

Krivine's realizability models: single-pole models

$$\begin{array}{ccc} \text{All poles} & \longrightarrow & \text{Extensions of ZF} \\ \\ \perp\!\!\!\perp & \longmapsto & \text{Theory}(\perp\!\!\!\perp) = \\ & & \{\, A;\ \exists t \text{ proof-like}, t \in |A|_{\perp\!\!\!\perp} \,\} \end{array}$$

Proof-like term: a term in which no stack constant (k_π) appears.

Adequacy: if $A \vdash_{\text{classical}} B$ and A is realized, then B is realized.

Krivine's realizability models: comparison with forcing

Realizability	forcing

Krivine's realizability models: comparison with forcing

Realizability	forcing
$\Lambda, \Pi, \Lambda \star \Pi$	P

Krivine's realizability models: comparison with forcing

Realizability	forcing
$\Lambda, \Pi, \Lambda \star \Pi$	P
$\succ$	$\leq$

Krivine's realizability models: comparison with forcing

Realizability	forcing
$\Lambda, \Pi, \Lambda \star \Pi$	P
$\succ$	$\leq$
$tu, t \cdot \pi, t \star \pi$	$t \wedge u, t \wedge \pi$

Krivine's realizability models: comparison with forcing

Realizability	forcing
$\Lambda, \Pi, \Lambda \star \Pi$	P
$\succ$	$\leq$
$tu, t \centerdot \pi, t \star \pi$	$t \wedge u, t \wedge \pi$
$t \star \pi \in \perp\!\!\!\perp$	$t \perp \pi$

Krivine's realizability models: comparison with forcing

Realizability	forcing
$\Lambda, \Pi, \Lambda \star \Pi$	P
$\succ$	$\leq$
$tu, t \mathbin{.} \pi, t \star \pi$	$t \wedge u, t \wedge \pi$
$t \star \pi \in \perp\!\!\!\perp$	$t \perp \pi$
proof-like	$\mathbb{1}$

Krivine's realizability models: comparison with forcing

Realizability	forcing
$\Lambda, \Pi, \Lambda \star \Pi$	P
$\succ$	$\leq$
$tu, t \cdot \pi, t \star \pi$	$t \wedge u, t \wedge \pi$
$t \star \pi \in \perp\!\!\!\perp$	$t \perp \pi$
proof-like	$\mathbb{1}$
realizes	forces

Krivine's realizability models: $\beth 2$

$$\begin{array}{c}\text{First-order formula } A\\ \text{on Boolean Algebras}\end{array} \xrightarrow[\textit{translation}]{} \begin{array}{c}\text{Formula } (\beth 2 \models A) \text{ of the}\\ \text{realizability language}\end{array}$$

Krivine's realizability models: $ℶ2$

First-order formula A on Boolean Algebras $\xrightarrow[\textit{translation}]{}$ Formula ($ℶ2 \models A$) of the realizability language

$$\forall x \; \forall y \; \forall z \quad x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

The operation $\wedge$ is associative

Krivine's realizability models: $\gimel 2$

First-order formula A on Boolean Algebras $\xrightarrow[\textit{translation}]{}$ Formula ($\gimel 2 \models A$) of the realizability language

$$\gimel 2 \models \forall x\ \forall y\ \forall z \quad x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

The operation $\wedge$ on $\gimel 2$ is associative

Krivine's realizability models: $\gimel 2$

First-order formula A on Boolean Algebras $\xrightarrow[\textit{translation}]{}$ Formula ($\gimel 2 \models A$) of the realizability language

$\gimel 2 \models \forall x \, \forall y \, \forall z$
$x \wedge (y \wedge z) = (x \wedge y) \wedge z$

The operation $\wedge$ on $\gimel 2$ is associative

$\forall x \, (x = 0) \vee (x = 1)$ There are only two elements

Krivine's realizability models: $\gimel 2$

First-order formula A on Boolean Algebras $\xrightarrow[\textit{translation}]{}$ Formula ($\gimel 2 \models A$) of the realizability language

$$\gimel 2 \models \ \forall x\, \forall y\, \forall z \quad x \wedge (y \wedge z) = (x \wedge y) \wedge z$$

The operation $\wedge$ on $\gimel 2$ is associative

$$\gimel 2 \models \ \forall x\, (x = 0) \vee (x = 1)$$

$\gimel 2$ only has two elements

Krivine's realizability models: $\gimel 2$

$$\begin{array}{c}\text{First-order formula } A\\ \text{on Boolean Algebras}\end{array} \xrightarrow[\textit{translation}]{} \begin{array}{c}\text{Formula } (\gimel 2 \models A) \text{ of the}\\ \text{realizability language}\end{array}$$

$$|\gimel 2 \models A \to B|_{\perp\!\!\perp} \qquad = \qquad |(\gimel 2 \models A) \to (\gimel 2 \models B)|_{\perp\!\!\perp}$$

Krivine's realizability models: $\beth 2$

First-order formula A on Boolean Algebras $\xrightarrow[\textit{translation}]{}$ Formula $(\beth 2 \models A)$ of the realizability language

$$|\beth 2 \models A \to B|_{\perp\!\!\!\perp} \quad = \quad |(\beth 2 \models A) \to (\beth 2 \models B)|_{\perp\!\!\!\perp}$$

$$|\beth 2 \models \forall x\ A(x)|_{\perp\!\!\!\perp} \quad = \quad |\beth 2 \models A(0)|_{\perp\!\!\!\perp} \cap |\beth 2 \models A(1)|_{\perp\!\!\!\perp}$$

Krivine's realizability models: $\gimel 2$

$$\gimel 2 \models (0 = 0) \vee (0 = 1)$$

$$\gimel 2 \models (1 = 0) \vee (1 = 1)$$

$$\gimel 2 \models \forall x \; (x = 0) \vee (x = 1)$$

Krivine's realizability models: $\gimel 2$

$\gimel 2 \models (0 = 0) \vee (0 = 1)$ Realized by $\lambda x.\ \lambda y.\ x$

$\gimel 2 \models (1 = 0) \vee (1 = 1)$

$\gimel 2 \models \forall x\ (x = 0) \vee (x = 1)$

Krivine's realizability models: $\gimel 2$

$$\gimel 2 \models (0 = 0) \vee (0 = 1)$$ Realized by $\lambda x.\ \lambda y.\ x$

$$\gimel 2 \models (1 = 0) \vee (1 = 1)$$ Realized by $\lambda x.\ \lambda y.\ y$

$$\gimel 2 \models \forall x\ (x = 0) \vee (x = 1)$$

Krivine's realizability models: $\gimel 2$

$\gimel 2 \models (0 = 0) \vee (0 = 1)$ Realized by $\lambda x.\ \lambda y.\ x$

$\gimel 2 \models (1 = 0) \vee (1 = 1)$ Realized by $\lambda x.\ \lambda y.\ y$

$\gimel 2 \models \forall x\ (x = 0) \vee (x = 1)$ Not always realized (depends on $\perp\!\!\!\perp$)

Contributions

- Realizability structures & multi-evaluation relations,

Contributions

- Realizability structures & multi-evaluation relations,
- $\gimel 2$ can be elementarily equivalent to any Boolean algebra,

Contributions

- Realizability structures & multi-evaluation relations,
- $\gimel 2$ can be elementarily equivalent to any Boolean algebra,
- The problem of relative definability can be studied and solved with $\gimel 2$ in some models of computation,

Contributions

- Realizability structures & multi-evaluation relations,
- $\gimel 2$ can be elementarily equivalent to any Boolean algebra,
- The problem of relative definability can be studied and solved with $\gimel 2$ in some models of computation,
- For every λ, $DC_{\widehat{\lambda}}$ can be realized,

Contributions

- Realizability structures & multi-evaluation relations,
- $\beth 2$ can be elementarily equivalent to any Boolean algebra,
- The problem of relative definability can be studied and solved with $\beth 2$ in some models of computation,
- For every λ, $DC_{\widehat{\lambda}}$ can be realized,
- With Laura Fontanella: "$\widehat{\lambda}$ is a cardinal" can be realized.

Multiple poles: realizability structures

Limitation of single-pole models: lack of modularity

- If $⫫$ grows:

Limitation of single-pole models: lack of modularity

- If $\perp\!\!\!\perp$ grows: - $\|a \not\leq b\|_{\perp\!\!\!\perp}$ unchanged

Limitation of single-pole models: lack of modularity

- If $⫫$ grows: - $\|a \not\in b\|_{⫫}$ unchanged
  - $|a \not\in b|_{⫫}$ grows

Limitation of single-pole models: lack of modularity

- If $⫫$ grows:

-	$\|a \not\varepsilon\, b\|_{⫫}$	unchanged
-	$\lvert a \not\varepsilon\, b\rvert_{⫫}$	grows
-	$\|a \;\varepsilon\; b\|_{⫫}$	grows

Limitation of single-pole models: lack of modularity

- If $\perp\!\!\!\perp$ grows:
  - $\|a \not\mathcal{E}\, b\|_{\perp\!\!\!\perp}$ unchanged
  - $|a \not\mathcal{E}\, b|_{\perp\!\!\!\perp}$ grows
  - $\|a \,\varepsilon\, b\|_{\perp\!\!\!\perp}$ grows
  - $|a \,\varepsilon\, b|_{\perp\!\!\!\perp}$???

Limitation of single-pole models: lack of modularity

- If $\perp\!\!\!\perp$ grows:
  - $\|a \not\varepsilon b\|_{\perp\!\!\!\perp}$ unchanged
  - $|a \not\varepsilon b|_{\perp\!\!\!\perp}$ grows
  - $\|a \varepsilon b\|_{\perp\!\!\!\perp}$ grows
  - $|a \varepsilon b|_{\perp\!\!\!\perp}$???

- How to combine two poles $\perp\!\!\!\perp_1$ and $\perp\!\!\!\perp_2$?

Realizability structures

$$\begin{array}{ccc} \text{All sets of poles} & \longrightarrow & \text{Extensions of ZF} \\ \text{(realizability structures)} & & \\ & & \\ \mathcal{S} & \longmapsto & \text{Theory}(\mathcal{S}) \end{array}$$

Realizability structures

$$
\begin{array}{ccc}
\text{All sets of poles} & \longrightarrow & \text{Extensions of ZF} \\
\text{(realizability structures)} & & \\
& & \\
\mathcal{S} & \longmapsto & \text{Theory}(\mathcal{S}) \; = \\
& & \left\{ \begin{array}{c} A;\ \exists t \text{ proof-like}, \\ t \Vdash_{\mathcal{S}} A \end{array} \right\}
\end{array}
$$

Realizability structures

$$\begin{array}{ccc} \text{All sets of poles} & \longrightarrow & \text{Extensions of ZF} \\ \text{(realizability structures)} & & \\ & & \\ \mathcal{S} & \longmapsto & \text{Theory}(\mathcal{S}) \ = \\ & & \left\{ \begin{array}{c} A;\ \exists t \text{ proof-like}, \\ \forall \perp\!\!\!\perp \in \mathcal{S},\ t \in |A|_{\perp\!\!\!\perp} \end{array} \right\} \end{array}$$

Realizability structures

$$\begin{array}{ccc} \text{All sets of poles} & \longrightarrow & \text{Extensions of ZF} \\ \text{(realizability structures)} & & \\ & & \\ \mathcal{S} & \longmapsto & \text{Theory}(\mathcal{S}) \ = \\ & & \left\{ \begin{array}{c} A;\ \exists t \text{ proof-like}, \\ \forall \perp\!\!\!\perp \in \mathcal{S},\ t \in |A|_{\perp\!\!\!\perp} \end{array} \right\} \end{array}$$

If $\mathcal{S}_1 \subseteq \mathcal{S}_2$, then $\text{Theory}(\mathcal{S}_1) \supseteq \text{Theory}(\mathcal{S}_2)$.

Realizability structures

$$\begin{array}{ccc} \text{All sets of poles} & \longrightarrow & \text{Extensions of ZF} \\ \text{(realizability structures)} & & \\ & & \\ \mathcal{S} & \longmapsto & \text{Theory}(\mathcal{S}) \; = \\ & & \left\{ \begin{array}{c} A;\ \exists t \text{ proof-like}, \\ \forall \perp\!\!\!\perp \in \mathcal{S},\ t \in |A|_{\perp\!\!\!\perp} \end{array} \right\} \end{array}$$

If $\mathcal{S}_1 \subseteq \mathcal{S}_2$, then $\text{Theory}(\mathcal{S}_1) \supseteq \text{Theory}(\mathcal{S}_2)$.

Combining $\mathcal{S}_1$ and $\mathcal{S}_2$: $\mathcal{S}_1 \cap \mathcal{S}_2$.

Multi-evaluation relations

$$p \succ_{\mathcal{S}} q \quad \textit{iff} \ \ \forall \perp\!\!\!\perp \in \mathcal{S}, q \in \perp\!\!\!\perp \Rightarrow p \in \perp\!\!\!\perp$$

Multi-evaluation relations

$$p \succ_{\mathcal{S}} q \quad \textit{iff} \;\; \forall \perp\!\!\!\perp \in \mathcal{S}, q \in \perp\!\!\!\perp \Rightarrow p \in \perp\!\!\!\perp$$

$$p \succ_K q \Rightarrow p \succ_{\mathcal{S}} q$$

Multi-evaluation relations

$$p \succ_{\mathcal{S}} q \quad \textit{iff} \;\; \forall \perp\!\!\!\perp \in \mathcal{S}, q \in \perp\!\!\!\perp \Rightarrow p \in \perp\!\!\!\perp$$

$$p \succ_{K} q \Rightarrow p \succ_{\mathcal{S}} q$$

$\succ_{\mathcal{S}}$ reflexive and transitive

Multi-evaluation relations

$$p \succ_{\mathcal{S}} q \quad \textit{iff} \ \ \forall \perp\!\!\!\perp \in \mathcal{S}, q \in \perp\!\!\!\perp \Rightarrow p \in \perp\!\!\!\perp$$

$p \succ_K q \Rightarrow p \succ_{\mathcal{S}} q$

$\succ_{\mathcal{S}}$ reflexive and transitive

$\mathcal{S}$ not uniquely detemined by $\succ_{\mathcal{S}}$

Multi-evaluation relations

$$P \succ_{\mathcal{S}} Q \ \ \textit{iff} \ \ \forall \bot\!\!\!\bot \in \mathcal{S}, Q \subseteq \bot\!\!\!\bot \Rightarrow P \cap \bot\!\!\!\bot \neq \emptyset$$

Multi-evaluation relations

$$P \succ_{\mathcal{S}} Q \quad \textit{iff} \;\; \forall {\perp\!\!\!\perp} \in \mathcal{S}, Q \subseteq {\perp\!\!\!\perp} \Rightarrow P \cap {\perp\!\!\!\perp} \neq \emptyset$$

$\mathcal{S}$ is uniquely determined by $\succ_{\mathcal{S}}$

Multi-evaluation relations

Let $\mathcal{S}$ be a realizability structure:

Multi-evaluation relations

Let $\mathcal{S}$ be a realizability structure:

- For all p, q such that $p \succ_K q$, $\{p\} \succ_{\mathcal{S}} \{q\}$,

Multi-evaluation relations

Let $\mathcal{S}$ be a realizability structure:

- For all p, q such that $p \succ_K q$, $\{p\} \succ_{\mathcal{S}} \{q\}$,
- For all p, $\{p\} \succ_{\mathcal{S}} \{p\}$, *(identity)*

Multi-evaluation relations

Let $\mathcal{S}$ be a realizability structure:

- For all p, q such that $p \succ_K q$, $\{p\} \succ_{\mathcal{S}} \{q\}$,
- For all p, $\{p\} \succ_{\mathcal{S}} \{p\}$, *(identity)*
- For all P, Q, P', Q', for all r, *(cut)*
  if $P \succ_{\mathcal{S}} Q \cup \{r\}$ and $P' \cup \{r\} \succ_{\mathcal{S}} Q'$, then $P \cup P' \succ_{\mathcal{S}} Q \cup Q'$,

Multi-evaluation relations

Let $\mathcal{S}$ be a realizability structure:

- For all p, q such that $p \succ_K q$, $\{p\} \succ_{\mathcal{S}} \{q\}$,
- For all p, $\{p\} \succ_{\mathcal{S}} \{p\}$, *(identity)*
- For all P, Q, P', Q', for all r, *(cut)*
  if $P \succ_{\mathcal{S}} Q \cup \{r\}$ and $P' \cup \{r\} \succ_{\mathcal{S}} Q'$, then $P \cup P' \succ_{\mathcal{S}} Q \cup Q'$,
- For all P, Q, P', Q' such that $P \succ_{\mathcal{S}} Q$, *(weakening)*
  if $P \subseteq P'$ *and* $Q \subseteq Q'$, *then* $P' \succ_{\mathcal{S}} Q'$.

Multi-evaluation relations

A *multi-evaluation relation* is a binary relation $\succ$ between sets of processes such that:

- For all p, q such that $p \succ_K q$, $\{p\} \succ \{q\}$,
- For all p, $\{p\} \succ \{p\}$, *(identity)*
- For all P, Q, P', Q', for all r, *(cut)*
  if $P \succ Q \cup \{r\}$ and $P' \cup \{r\} \succ Q'$, then $P \cup P' \succ Q \cup Q'$,
- For all P, Q, P', Q' such that $P \succ Q$, *(weakening)*
  if $P \subseteq P'$ *and* $Q \subseteq Q'$*, then* $P' \succ Q'$.

Multi-evaluation relations

A *multi-evaluation relation* is a binary relation $\succ$ between sets of processes such that:

- For all p, q such that $p \succ_K q$, $\{p\} \succ \{q\}$,
- For all p, $\{p\} \succ \{p\}$, *(identity)*
- For all P, Q, P', Q', for all r, *(cut)*
  if $P \succ Q \cup \{r\}$ and $P' \cup \{r\} \succ Q'$, then $P \cup P' \succ Q \cup Q'$,
- For all P, Q, P', Q' such that $P \succ Q$, *(weakening)*
  if $P \subseteq P'$ *and* $Q \subseteq Q'$, *then* $P' \succ Q'$.

$$\succ \qquad \longrightarrow \mathcal{S}_{\succ} = \left\{ \perp\!\!\!\perp ; \quad \begin{array}{c} \forall P, Q \text{ s.t. } P \succ Q, \\ Q \subseteq \perp\!\!\!\perp \Rightarrow P \cap \perp\!\!\!\perp \neq \emptyset \end{array} \right\}$$

Taming ב2 (somewhat)

Adding instructions

$$\text{Infinitely many} \begin{cases} \text{- unrestricted instructions} \\ \text{- restricted instructions} \end{cases} \text{with no rules in } \succ_K.$$

Adding instructions

Infinitely many $\left\{ \begin{array}{l} \text{- unrestricted instructions} \\ \text{- restricted instructions} \end{array} \right.$ with no rules in $\succ_K$.

Proof-like term: a term in which no stack constant (k_π) or restricted instruction appears.

Realizing "$\beth 2$ has fewer than 2^n elements"

Proposition

$\text{Theory}(\mathcal{S}) \vdash \left(\beth 2 \models \forall x_1 \ \ldots \ \forall x_{2^n} \ \bigvee_{i \neq j} (x_i = x_j)\right)$ *iff*

Realizing "$\beth 2$ has fewer than 2^n elements"

Proposition

$\mathsf{Theory}(\mathcal{S}) \vdash \left(\beth 2 \models \mathsf{Fewer}_{2^n}\right)$ *iff*

Realizing "$\beth 2$ has fewer than 2^n elements"

Proposition

$\mathrm{Theory}(\mathcal{S}) \vdash \left(\beth 2 \models \mathsf{Fewer}_{2^n}\right)$ *iff*

$$\mathrm{Theory}(\mathcal{S}) \vdash \left\{ \begin{array}{cccccccccc} & \top & \to & \bot & \to & \dots & \to & \bot & \to & \bot \\ \cap & \bot & \to & \top & \to & \dots & \to & \bot & \to & \bot \\ \vdots & & & & & \ddots & & & & \vdots \\ \cap & \multicolumn{7}{c}{\underbrace{\bot \to \bot \to \dots \to \top}_{n \text{ arguments}}} & \to & \bot \end{array} \right. .$$

Realizing "$\gimel 2$ has fewer than 2^n elements"

Proposition

$\mathrm{Theory}(\mathcal{S}) \vdash \left(\gimel 2 \models \forall x_1 \ \ldots \ \forall x_{2^n} \ \bigvee_{i \neq j} (x_i = x_j)\right)$ *iff*

$$\mathrm{Theory}(\mathcal{S}) \vdash \left\{ \begin{array}{ccccccccccccc} & \top & \to & \bot & \to & \ldots & \to & \bot & \to & \bot \\ \cap & \bot & \to & \top & \to & \ldots & \to & \bot & \to & \bot \\ \vdots & & & & & \ddots & & & & \vdots \\ \cap & \underbrace{\bot \ \to \ \bot \ \to \ \ldots \ \to \ \top}_{n \text{ arguments}} & & & & & & & \to & \bot \end{array} \right. .$$

Realizing "$\beth 2$ has fewer than 2^n elements"

Let φ be any unrestricted instruction.

Realizing "ℶ2 has fewer than 2^n elements"

Let φ be any unrestricted instruction.

$$\text{Let } \mathcal{S}_{<2^n} = \left\{ \perp\!\!\!\perp; \begin{array}{l} \text{for all } u_1, \ldots, u_n, \pi, \\ \text{if all but one of the } u_i \star \pi \text{ are in } \perp\!\!\!\perp, \\ \text{then } \varphi \star u_1 \bullet \ldots \bullet u_n \bullet \pi \text{ is in } \perp\!\!\!\perp \end{array} \right\}.$$

Realizing "ℶ2 has fewer than 2^n elements"

Let φ be any unrestricted instruction.

Let $\mathcal{S}_{<2^n}$ be the largest structure such that for all $u_1, \ldots, u_n, \pi$,

$$\begin{array}{ccc}
\{\varphi \star u_1 \bullet \ldots \bullet u_n \bullet \pi\} & \succ_{\mathcal{S}_{<2^n}} & \{u_i \star \pi;\ i \neq 1\} \\
\vdots & \vdots & \vdots \\
\{\varphi \star u_1 \bullet \ldots \bullet u_n \bullet \pi\} & \succ_{\mathcal{S}_{<2^n}} & \{u_i \star \pi;\ i \neq n\}.
\end{array}$$

Realizing "$\beth 2$ has fewer than 2^n elements"

Let φ be any unrestricted instruction.

Let $\mathcal{S}_{<2^n}$ be the largest structure such that for all $u_1, \ldots, u_n, \pi$,

$$\begin{array}{ccc} \{\varphi \star u_1 \bullet \ldots \bullet u_n \bullet \pi\} & \succ_{\mathcal{S}_{<2^n}} & \{u_i \star \pi;\ i \neq 1\} \\ \vdots & \vdots & \vdots \\ \{\varphi \star u_1 \bullet \ldots \bullet u_n \bullet \pi\} & \succ_{\mathcal{S}_{<2^n}} & \{u_i \star \pi;\ i \neq n\}. \end{array}$$

Proposition

$\mathrm{Theory}(\mathcal{S}_{<2^n}) \vdash (\beth 2 \models \mathrm{Fewer}_{2^n})$.

Realizing "$\beth 2$ has at least 2^n elements"

Let ξ be any unrestricted instruction other than φ.

Realizing "$\beth 2$ has at least 2^n elements"

Let ξ be any unrestricted instruction other than φ.
Let $c_\top$ and $c_\perp$ be two restricted instructions.

Realizing "$\beth 2$ has at least 2^n elements"

Let ξ be any unrestricted instruction other than φ.
Let $c_\top$ and $c_\bot$ be two restricted instructions.

Let $\mathcal{S}_{\top\bot} = \{\ \perp\!\!\!\perp;\ \text{for all } \pi, c_\bot \star \pi \in \perp\!\!\!\perp\ \}$.

Realizing "$\beth 2$ has at least 2^n elements"

Let ξ be any unrestricted instruction other than φ.
Let $c_\top$ and $c_\perp$ be two restricted instructions.

Let $\mathcal{S}_{\top\perp} = \{\ \perp\!\!\!\perp;\ \text{for all } \pi, c_\perp \star \pi \in \perp\!\!\!\perp\ \}$.

$$\text{Let } \mathcal{S}_{\geq 2^n} = \mathcal{S}_{\top\perp} \cap \left\{ \perp\!\!\!\perp;\ \begin{array}{l} \text{for all } u, \pi, \\ \text{if} \quad u \star c_\top \bullet c_\perp \bullet \ldots \bullet c_\perp \star \pi \in \perp\!\!\!\perp \\ \text{and } u \star c_\perp \bullet c_\top \bullet \ldots \bullet c_\perp \star \pi \in \perp\!\!\!\perp \\ \qquad\qquad\qquad \ddots \\ \text{and } u \star c_\perp \bullet c_\perp \bullet \ldots \bullet c_\top \star \pi \in \perp\!\!\!\perp \\ \text{then } \xi \star u \bullet \pi \in \perp\!\!\!\perp \end{array} \right\}.$$

Realizing "ℶ2 has at least 2^n elements"

Let ξ be any unrestricted instruction other than φ.
Let $c_\top$ and $c_\bot$ be two restricted instructions.

Let $\mathcal{S}_{\geq 2^n}$ be the largest structure such that for all u, π:

- $\{c_\bot \star \pi\} \succ_{\mathcal{S}_{\geq 2^n}} \{\ \}$,

Realizing "ℶ2 has at least 2^n elements"

Let ξ be any unrestricted instruction other than φ.
Let $c_\top$ and $c_\bot$ be two restricted instructions.

Let $\mathcal{S}_{\geq 2^n}$ be the largest structure such that for all u, π:

- $\{c_\bot \star \pi\} \succ_{\mathcal{S}_{\geq 2^n}} \{\ \}$,
- $\{\xi \star u \bullet \pi\} \succ_{\mathcal{S}_{\geq 2^n}} \left\{ \begin{array}{c} u \star c_\top \bullet c_\bot \bullet \ldots \bullet c_\bot \star \pi, \\ u \star c_\bot \bullet c_\top \bullet \ldots \bullet c_\bot \star \pi, \\ \ddots \\ u \star c_\bot \bullet c_\bot \bullet \ldots \bullet c_\top \star \pi \end{array} \right\}$.

Realizing "$\gimel 2$ has at least 2^n elements"

Let ξ be any unrestricted instruction other than φ.
Let $c_\top$ and $c_\bot$ be two restricted instructions.

Let $\mathcal{S}_{\geq 2^n}$ be the largest structure such that for all u, π:

- $\{c_\bot \star \pi\} \succ_{\mathcal{S}_{\geq 2^n}} \{\ \}$,
- $\{\xi \star u \cdot \pi\} \succ_{\mathcal{S}_{\geq 2^n}} \left\{ \begin{array}{c} u \star c_\top \cdot c_\bot \cdot \ldots \cdot c_\bot \star \pi, \\ u \star c_\bot \cdot c_\top \cdot \ldots \cdot c_\bot \star \pi, \\ \ddots \\ u \star c_\bot \cdot c_\bot \cdot \ldots \cdot c_\top \star \pi \end{array} \right\}$.

Proposition

Theory$(\mathcal{S}_{\geq 2^n}) \vdash (\gimel 2 \models \text{Fewer}_{2^n} \to \bot)$.

Realizing "$\beth 2$ has exactly 2^n elements"

Let $\mathcal{S}_{=2^n} = \mathcal{S}_{<2^{n+1}} \cap \mathcal{S}_{\geq 2^n}$.

Realizing "$\beth 2$ has exactly 2^n elements"

Let $\mathcal{S}_{=2^n} = \mathcal{S}_{<2^{n+1}} \cap \mathcal{S}_{\geq 2^n}$.

Proposition

$\mathrm{Theory}(\mathcal{S}_{=2^n}) \vdash (\beth 2 \models \mathrm{Fewer}_{2^n+1} \ \wedge \neg\, \mathrm{Fewer}_{2^n})$.

Realizing "$\beth 2$ has exactly 2^n elements"

Let $\mathcal{S}_{=2^n} = \mathcal{S}_{<2^{n+1}} \cap \mathcal{S}_{\geq 2^n}$.

Proposition

$\mathrm{Theory}(\mathcal{S}_{=2^n}) \vdash (\beth 2 \models \mathrm{Fewer}_{2^n+1} \ \wedge \neg\, \mathrm{Fewer}_{2^n})$.

Proposition

$\mathrm{Theory}(\mathcal{S}_{=2^n}) \nvdash \bot$.

Realizing any consistent Boolean formula

Theorem

Let A be a Boolean formula. There exists a realizability structure $\mathcal{S}$ such that $\text{Theory}(\mathcal{S}) \vdash \daleth 2 \models A$ and $\text{Theory}(\mathcal{S}) \nvdash \bot$ *iff* A is satisfiable in Boolean algebras with at least two elements.

Realizing any consistent Boolean formula

Theorem

Let A be a Boolean formula. There exists a realizability structure $\mathcal{S}$ such that $\mathrm{Theory}(\mathcal{S}) \vdash \beth 2 \models A$ and $\mathrm{Theory}(\mathcal{S}) \nvdash \perp$ *iff* A is satisfiable in Boolean algebras with at least two elements.

Corollary

Let $\mathbb{B}$ be a Boolean algebra. There exists a realizability structure $\mathcal{S}$ such that $\mathrm{Theory}(\mathcal{S}) \vdash$ "$\beth 2$ is elementarily equivalent to $\mathbb{B}$" and $\mathrm{Theory}(\mathcal{S}) \nvdash \perp$.

Generalized dependent choices (restricted Zorn's lemma)

A well-ordering instruction

Let $<$ be a well-ordering on terms,

A well-ordering instruction

Let $<$ be a well-ordering on terms,
Let χ be any non-restricted instruction.

A well-ordering instruction

Let $<$ be a well-ordering on terms,
Let χ be any non-restricted instruction.

Let $\mathcal{S}$ be the largest structure such that for all a, b, t, u, v, π :

$$\{\chi \star a \bullet b \bullet t \bullet u \bullet v \bullet \pi\} \succ_{\mathcal{S}} \begin{array}{ll} \{t \star \pi\} & \text{if } a < b \\ \{u \star \pi\} & \text{if } a = b \\ \{v \star \pi\} & \text{if } a > b \end{array} .$$

Non-extensional choice

Proposition

Theory($\mathcal{S}$) $\vdash$ "Every family of non-empty sets has a (possibly non $\approx$-compatible) choice function".

Non-extensional choice

Proposition

Theory($\mathcal{S}$) $\vdash$ "Every family of non-empty sets has a (possibly non $\approx$-compatible) choice function".

$$a \longmapsto \mathrm{choice}(a) \,\varepsilon\, a$$

Non-extensional choice

Proposition

Theory($\mathcal{S}$) $\vdash$ "Every family of non-empty sets has a (possibly non $\approx$-compatible) choice function".

$$a \longmapsto \text{choice}(a) \; \varepsilon \; a$$

$$b \longmapsto \text{choice}(b) \; \varepsilon \; b$$

Non-extensional choice

Proposition

Theory($\mathcal{S}$) $\vdash$ "Every family of non-empty sets has a (possibly non $\approx$-compatible) choice function".

$$\begin{array}{ccc} a & \longmapsto & \text{choice}(a)\ \varepsilon\ a \\ = & & \\ b & \longmapsto & \text{choice}(b)\ \varepsilon\ b \end{array}$$

Non-extensional choice

Proposition

Theory($\mathcal{S}$) $\vdash$ "Every family of non-empty sets has a (possibly non $\approx$-compatible) choice function".

$$\begin{array}{ccc} a & \longmapsto & \mathrm{choice}(a) \; \varepsilon \; a \\ = & & = \\ b & \longmapsto & \mathrm{choice}(b) \; \varepsilon \; b \end{array}$$

Non-extensional choice

Proposition

Theory($\mathcal{S}$) $\vdash$ "Every family of non-empty sets has a (possibly non $\approx$-compatible) choice function".

$$\begin{array}{ccc} a & \longmapsto & \text{choice}(a)\ \varepsilon\ a \\ \approx & & \\ b & \longmapsto & \text{choice}(b)\ \varepsilon\ b \end{array}$$

Non-extensional choice

Proposition

Theory($\mathcal{S}$) $\vdash$ "Every family of non-empty sets has a (possibly non $\approx$-compatible) choice function".

$$\begin{array}{ccc} a & \longmapsto & \text{choice}(a) \; \varepsilon \; a \\ \approx & & \not\approx \\ b & \longmapsto & \text{choice}(b) \; \varepsilon \; b \end{array}$$

The generic ordinal $\widehat{\lambda}$

Let λ be the cardinal of Λ

The generic ordinal $\widehat{\lambda}$

Let λ be the cardinal of Λ
Let $(\tau_\alpha)_{\alpha<\lambda}$ be a list of terms s.t. $\beta < \alpha \Rightarrow \tau_\beta < \tau_\alpha$

The generic ordinal $\widehat{\lambda}$

Let λ be the cardinal of Λ
Let $(\tau_\alpha)_{\alpha<\lambda}$ be a list of terms s.t. $\beta < \alpha \Rightarrow \tau_\beta < \tau_\alpha$

For all $\alpha \leq \lambda$, let $\widehat{\alpha} = \left\{ \left(\widehat{\beta}, \tau_\beta \bullet \pi\right);\ \beta < \alpha, \pi \text{ stack} \right\}$.

Realizers of $\widehat{\beta} \,\varepsilon\, \widehat{\alpha}$: $\sim \begin{cases} \tau_\beta & \text{if } \beta < \alpha \\ \text{none} & \text{if } \beta \geq \alpha \end{cases}$.

Canonical representatives of ordinals

Proposition

Theory($\mathcal{S}$) $\vdash \forall \alpha \varepsilon \widehat{\lambda}\ \forall \beta \varepsilon \widehat{\lambda}\ \ (\alpha \varepsilon \beta \vee \alpha{=}\beta \vee \beta \varepsilon \alpha)$.

Canonical representatives of ordinals

Proposition

Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ \ (\alpha\varepsilon\beta \vee \alpha{=}\beta \vee \beta\varepsilon\alpha)$.

Corollary

Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ \ (\alpha{\neq}\beta \to \alpha\varepsilon\beta \vee \beta\varepsilon\alpha)$.

Canonical representatives of ordinals

Proposition

Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ \ (\alpha\varepsilon\beta \vee \alpha{=}\beta \vee \beta\varepsilon\alpha)$.

Corollary

Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ \ (\alpha{\neq}\beta \to \alpha{\in}\beta \vee \beta{\in}\alpha)$.

Canonical representatives of ordinals

Proposition

Theory($\mathcal{S}$) $\vdash \forall \alpha \varepsilon \widehat{\lambda}\ \forall \beta \varepsilon \widehat{\lambda}\ \ (\alpha \varepsilon \beta \vee \alpha{=}\beta \vee \beta \varepsilon \alpha)$.

Corollary

Theory($\mathcal{S}$) $\vdash \forall \alpha \varepsilon \widehat{\lambda}\ \forall \beta \varepsilon \widehat{\lambda}\ \ (\alpha{\neq}\beta \rightarrow \alpha{\not\approx}\beta)$.

Canonical representatives of ordinals

Proposition

Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ \ (\alpha\varepsilon\beta \vee \alpha{=}\beta \vee \beta\varepsilon\alpha)$.

Corollary

Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ \ (\alpha{\approx}\beta \rightarrow \alpha{=}\beta)$.

Canonical representatives of ordinals

Proposition

Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ \ (\alpha\varepsilon\beta \vee \alpha{=}\beta \vee \beta\varepsilon\alpha)$.

Corollary

Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ \ (\alpha{\approx}\beta \to \alpha{=}\beta)$.

Theorem

Theory($\mathcal{S}$) $\vdash$ "Every $\widehat{\lambda}$-indexed family of non-empty sets has a $\approx$-compatible choice function".

Canonical representatives of ordinals

Proposition
Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ (\alpha\varepsilon\beta \vee \alpha{=}\beta \vee \beta\varepsilon\alpha)$.

Corollary
Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ (\alpha{\approx}\beta \rightarrow \alpha{=}\beta)$.

Theorem
Theory($\mathcal{S}$) $\vdash$ $\mathsf{AC}_{\widehat{\lambda}}$.

Canonical representatives of ordinals

Proposition
Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ \ (\alpha\varepsilon\beta \vee \alpha{=}\beta \vee \beta\varepsilon\alpha)$.

Corollary
Theory($\mathcal{S}$) $\vdash \forall\alpha\varepsilon\widehat{\lambda}\ \forall\beta\varepsilon\widehat{\lambda}\ \ (\alpha{\approx}\beta \to \alpha{=}\beta)$.

Theorem
Theory($\mathcal{S}$) $\vdash$ $\mathsf{DC}_{\widehat{\lambda}}$ (dependent choice sequences up to length $\widehat{\lambda}$.)

Thank you!