# Higher-Order Logic and Set Theory: Stronger Together

Chad E. Brown

Czech Technical University in Prague

July 8, 2019

the European Research Council (ERC) grant nr. 649043 *AI4REASON*

# Outline

# Introduction

- **Egal** is a proof checker / interactive theorem prover for higher-order set theory.
- Specifically: Higher-Order Tarski-Grothendieck (HOTG)
  ZFC+universes

# Introduction

- **Egal** is a proof checker / interactive theorem prover for higher-order set theory.
- Specifically: Higher-Order Tarski-Grothendieck (HOTG)
  ZFC+universes
- Why another prover?

# Introduction

- **Egal** is a proof checker / interactive theorem prover for higher-order set theory.
- Specifically: Higher-Order Tarski-Grothendieck (HOTG)
  ZFC+universes
- Why another prover?
  - de Bruijn criteria: proofs easily checked by small independent proof checker

# Introduction

- **Egal** is a proof checker / interactive theorem prover for higher-order set theory.
- Specifically: Higher-Order Tarski-Grothendieck (HOTG)
  ZFC+universes
- Why another prover?
  - de Bruijn criteria: proofs easily checked by small independent proof checker
  - Quantifying over functions allows abstract statements (avoiding "fake theorems")

# Introduction

- **Egal** is a proof checker / interactive theorem prover for higher-order set theory.
- Specifically: Higher-Order Tarski-Grothendieck (HOTG)
  ZFC+universes
- Why another prover?
  - de Bruijn criteria: proofs easily checked by small independent proof checker
  - Quantifying over functions allows abstract statements (avoiding "fake theorems")
  - Most other libraries can be interpreted in HOTG, and so could be ported to Egal.

# Introduction

- **Egal** is a proof checker / interactive theorem prover for higher-order set theory.
- Specifically: Higher-Order Tarski-Grothendieck (HOTG)
  ZFC+universes
- Why another prover?
  - de Bruijn criteria: proofs easily checked by small independent proof checker
  - Quantifying over functions allows abstract statements (avoiding "fake theorems")
  - Most other libraries can be interpreted in HOTG, and so could be ported to Egal.
  - Some of the interpretations exploit "fake theorems"

# Theorem Proving in Set Theory

- Trybulec, et. al.: Mizar 1973-now
  - First-Order Tarski-Grothendieck
  - Scheme for Replacement
  - Interactive Theorem Prover / Proof Checker
  - Soft Typing System
  - Mathematical Input Style
- Quaife 1992 (JAR 1992)
  - von Neumann-Gödel-Bernays (Class Theory)
  - First Order Finitely Axiomatizable (even as clauses)
  - Modification of Boyer, et. al. 1986 (JAR 1986)
  - Using Otter: Automated Theorem Prover

# Theorem Proving in Set Theory

- Trybulec, et. al.: Mizar 1973-now
  - First-Order Tarski-Grothendieck
  - Scheme for Replacement
  - Interactive Theorem Prover / Proof Checker
  - Soft Typing System
  - Mathematical Input Style
- Quaife 1992 (JAR 1992)
  - von Neumann-Gödel-Bernays (Class Theory)
  - First Order Finitely Axiomatizable (even as clauses)
  - Modification of Boyer, et. al. 1986 (JAR 1986)
  - Using Otter: Automated Theorem Prover
- Isabelle-ZF (JAR 1996)
- Metamath

# Two Kinds of Pairs in Mizar

- $[x, y]$ : Kuratowski pair $\{\{x\}, \{x, y\}\}$
- $\langle x, y \rangle$ : Function from $\{1, 2\}$ with $1 \mapsto x, 2 \mapsto y$

Sometimes both are used.
Example: Definition in catalg_1:

```
func homsym(a,b) equals
[0,<*a,b*>];
```

# Two Kinds of Pairs in Mizar

- $[x, y]$ : Kuratowski pair $\{\{x\}, \{x, y\}\}$
- $\langle x, y \rangle$ : Function from $\{1, 2\}$ with $1 \mapsto x, 2 \mapsto y$

Sometimes both are used.
Example: Definition in catalg_1:

```
func homsym(a,b) equals
[0,<*a,b*>];
```

- Fake Theorem: $y \in \bigcup [x, y]$

# Two Kinds of Pairs in Mizar

- $[x, y]$ : Kuratowski pair $\{\{x\}, \{x, y\}\}$
- $\langle x, y \rangle$ : Function from $\{1, 2\}$ with $1 \mapsto x, 2 \mapsto y$

Sometimes both are used.
Example: Definition in catalg_1:

```
func homsym(a,b) equals
[0,<*a,b*>];
```

- Fake Theorem: $y \in \bigcup [x, y]$
- Fake Theorem: $[2, y] \in \langle x, y \rangle$

# Quaife's Pairs

- Quaife uses $\{\{x\}, \{x, \{y\}\}\}$
- Why not Kuratowski pairs?

# Quaife's Pairs

- Quaife uses $\{\{x\}, \{x, \{y\}\}\}$
- Why not Kuratowski pairs?
- Kuratowski pairs made the theory inconsistent.

# Quaife's Pairs

- Quaife uses $\{\{x\}, \{x, \{y\}\}\}$
- Why not Kuratowski pairs?
- Kuratowski pairs made the theory inconsistent.
- Let $V$ be the class of all sets
- Quaife simplified some of the Boyer, et. al., clauses
- preferring $(x, y) \in V \to \ldots$ over $x \in V, y \in V \to \ldots$.

# Quaife's Pairs

- Quaife uses $\{\{x\},\{x,\{y\}\}\}$
- Why not Kuratowski pairs?
- Kuratowski pairs made the theory inconsistent.
- Let $V$ be the class of all sets
- Quaife simplified some of the Boyer, et. al., clauses
- preferring $(x,y) \in V \to \ldots$ over $x \in V, y \in V \to \ldots$.
- Problem if a proper class is used in a pair.
- Kuratowski pairs give $(\emptyset, V) = (\emptyset, \emptyset)$ leading to $V \in V$

# Quaife's Pairs

- Quaife uses $\{\{x\},\{x,\{y\}\}\}$
- Why not Kuratowski pairs?
- Kuratowski pairs made the theory inconsistent.
- Let $V$ be the class of all sets
- Quaife simplified some of the Boyer, et. al., clauses
- preferring $(x,y) \in V \to \ldots$ over $x \in V, y \in V \to \ldots$.
- Problem if a proper class is used in a pair.
- Kuratowski pairs give $(\emptyset, V) = (\emptyset, \emptyset)$ leading to $V \in V$
- Quaife's pairs satisfy the "fake theorem" that $(x,y)$ is never equal to an ordered pair of sets if either $x$ or $y$ is a class.

# Fundamental Property of Pairing

- $P$ is a "pairing operator" if it takes two sets and returns a set such that

$$\forall xyzw.P\ x\ y = P\ z\ w\ \equiv\ x = z\ \wedge y = w$$

- If we have simple type theory over the set theory, we can define this a higher-order pairing predicate:

$$\lambda P : \iota\iota\iota.\forall xyzw.P\ x\ y = P\ z\ w\ \equiv\ x = z\ \wedge y = w$$

- A "real theorem" should work for any pairing:

$$\forall P.\text{pairing}\ P \rightarrow \Phi[P]$$

- Sometimes we may want to prove $\Phi[P]$ for a specific pairing operator $P$ and other times we may want the general case.

# Outline

# Higher-Order Logic (Quick Intro)

- Simple Type Theory (Church 1940)
- $\iota$ - base type
- $o$ - type of propositions
- $\sigma\tau$ - type of functions from $\sigma$ to $\tau$

Typed Terms:

- $\mathcal{V}_\sigma$ - variables $x$ of type $\sigma$
- $\mathcal{C}_\sigma$ - constants $c$ of type $\sigma$
- $\Lambda_\sigma$ - terms of type $\sigma$ generated by

$$s, t ::= x | c | st | \lambda x.s | s \to t | \forall x.s$$

  restricted to well-typed terms.
- $(\lambda x.s)$ has type $\sigma\tau$ where $x \in \mathcal{V}_\sigma$ and $s \in \Lambda_\tau$. It means the function sending $x$ to $s$.

# Higher-Order Logic (Quick Intro)

$$s, t ::= x|c|st|\lambda x.s|s \to t|\forall x.s$$

- Formula - term of type $o$
- Definable: $\wedge$, $\vee$, $\equiv$, $=$, $\exists$, $\exists!$ (Russell-Prawitz)
- Sometimes write $\lambda x : \sigma.s$ and $\forall x : \sigma.s$.
- $s \approx t$ means $s$ and $t$ are $\beta\eta$-convertible.

# Natural Deduction

$\Gamma$ ranges over finite sets of formulas.
Natural Deduction defines $\Gamma \vdash s$.

$$\frac{}{\Gamma \vdash s}\; s \text{ known} \qquad \frac{}{\Gamma \vdash s}\; s \in \Gamma \qquad \frac{\Gamma \vdash s}{\Gamma \vdash t}\; s \approx t$$

$$\frac{\Gamma \cup \{s\} \vdash t}{\Gamma \vdash s \to t} \qquad \frac{\Gamma \vdash s \to t \qquad \Gamma \vdash s}{\Gamma \vdash t}$$

$$\frac{\Gamma \vdash s^x_y}{\Gamma \vdash \forall x : \sigma . s}\; y \in \mathcal{V}_\sigma \text{ fresh} \qquad \frac{\Gamma \vdash \forall x : \sigma . s}{\Gamma \vdash s^x_t}\; t \in \Lambda_\sigma$$

## Proof Terms

Add names to assumptions. $\Gamma$ is $u_1 : s_1, \ldots, u_n : s_n$.
Proof term calculus for judgment $\Gamma \vdash \mathcal{D} : s$
meaning "$\mathcal{D}$ is a proof of $s$ under assumptions $\Gamma$."

$$\frac{}{\Gamma \vdash a : s}\ a : s \text{ known} \qquad \frac{}{\Gamma \vdash u : s}\ u : s \in \Gamma$$

$$\frac{\Gamma \vdash \mathcal{D} : s}{\Gamma \vdash \mathcal{D} : t}\ s \approx t$$

$$\frac{\Gamma \cup \{u : s\} \vdash \mathcal{D} : t}{\Gamma \vdash (\lambda u : s.\mathcal{D}) : s \to t} \qquad \frac{\Gamma \vdash \mathcal{D} : s \to t \qquad \Gamma \vdash \mathcal{E} : s}{\Gamma \vdash (\mathcal{D}\ \mathcal{E}) : t}$$

# Proof Terms

Add names to assumptions. $\Gamma$ is $u_1 : s_1, \ldots, u_n : s_n$.
Proof term calculus for judgment $\Gamma \vdash \mathcal{D} : s$
meaning "$\mathcal{D}$ is a proof of $s$ under assumptions $\Gamma$."

$$\frac{\Gamma \vdash \mathcal{D}^x_y : s^x_y}{\Gamma \vdash (\lambda x : \sigma.\mathcal{D}) : \forall x : \sigma.s}\ y \in \mathcal{V}_\sigma \text{ fresh}$$

$$\frac{\Gamma \vdash \mathcal{D} : \forall x : \sigma.s}{\Gamma \vdash (\mathcal{D}\ t) : s^x_t}\ t \in \Lambda_\sigma$$

- de Bruijn criteria: proofs easily checked by small independent proof checker

# Outline

# Higher-Order(ish) Set Theories

- Isabelle-ZF: Paulson JAR 1993 (FO, but $\lambda$'s)
- HOL with ZF: Gordon TPHOLs 1996
- Isabelle/HOLZF: Obua 2006

Why Higher-Order Tarski-Grothendieck?

- Mizar's MML can be translated into HOTG.
  (Brown Pąk CICM2019)
- HOL style libraries can be translated into HOTG.
- Dependent Type Theories (like Coq and Lean) can be translated into HOTG.

# Set Theory Constants

Take $\iota$ to mean the type of sets.

- $\varepsilon_\sigma : (\sigma o)\sigma$ — Choice Operator
- $\in : \iota\iota o$ — Membership
- $\emptyset : \iota$ — Empty Set
- $\bigcup : \iota\iota$ — Big Unions
- $\wp : \iota\iota$ — Power Sets
- $\mathsf{r} : \iota(\iota\iota)\iota$ — Replacement: $\{t | x \in s\}$ means $\mathsf{r}\ s\ (\lambda x.t)$
- $\mathcal{U} : \iota\iota$ — Universe Operator

# Axioms

Set of axioms:

- Choice for $\varepsilon_\sigma$ (scheme due to $\sigma$)
- Propositional Extensionality
- Functional Extensionality (scheme)
- Set Extensionality
- $\in$-Induction
- Empty
- Union
- Power
- Replacement
- Universes

ND system with axioms is Henkin complete for HOTG.
**Egal** is a proof checker for the ND system with proof terms.

# Relative Consistency

- Is HOTG too strong? Is it consistent?
- A standard model can be constructed given a 2-inaccessible cardinal (Brown Pąk Kaliszyk ITP 2019)
- As large cardinals go, 2-inaccessible is not very large.

# Basic Definitions

- If-then-else can be defined from $\varepsilon$.
- Unordered pairs $\{s, t\}$ can be defined as

$$\{\text{if } \emptyset \in X \text{ then } s \text{ else } t \mid X \in \wp(\wp\emptyset)\}$$

- Singletons $\{s\}$ are defined as $\{s, s\}$.
- $s \cup t$ is $\bigcup\{s, t\}$.

# Natural Numbers as Finite Ordinals

- 0 is $\emptyset$.
- $s^+$ is $s \cup \{s\}$.
- 1 is $0^+$, 2 is $1^+$, ...
- A predicate $\mathbf{N} : \iota o$ for the natural numbers is definable by higher-order quantification:

$$\lambda n : \iota . \forall p : \iota o . p\ 0 \wedge (\forall x . p\ x \rightarrow p\ (x \cup \{x\})) \rightarrow p\ n$$

- Theorem: $\forall n . \mathbf{N}\ n \rightarrow n \in \mathcal{U}\emptyset$

# Natural Numbers as Finite Ordinals

- 0 is $\emptyset$.
- $s^+$ is $s \cup \{s\}$.
- 1 is $0^+$, 2 is $1^+$, ...
- A predicate $\mathbf{N} : \iota o$ for the natural numbers is definable by higher-order quantification:

$$\lambda n : \iota . \forall p : \iota o . p\ 0 \wedge (\forall x . p\ x \ \rightarrow \ p\ (x \ \cup \ \{x\})) \rightarrow p\ n$$

- Theorem: $\forall n . \mathbf{N}\ n \rightarrow n \in \mathcal{U}\emptyset$
- Is this a fake theorem?

# Natural Numbers as Finite Ordinals

- 0 is $\emptyset$.
- $s^+$ is $s \cup \{s\}$.
- 1 is $0^+$, 2 is $1^+$, ...
- A predicate $\mathbf{N} : \iota o$ for the natural numbers is definable by higher-order quantification:

$$\lambda n : \iota.\forall p : \iota o.p\ 0 \wedge (\forall x.p\ x \rightarrow p\ (x \cup \{x\})) \rightarrow p\ n$$

- Theorem: $\forall n.\mathbf{N}\ n \rightarrow n \in \mathcal{U}\emptyset$
- Is this a fake theorem?
- "Real" abstract version:

$$\forall z : \iota.\forall S : \iota\iota.\forall n : \iota.$$
$$(\forall p : \iota o.p\ z \wedge (\forall x.p\ x \rightarrow p\ (S\ x)) \rightarrow p\ n) \rightarrow n \in \mathcal{U}\emptyset$$

# Natural Numbers as Finite Ordinals

- 0 is $\emptyset$.
- $s^+$ is $s \cup \{s\}$.
- 1 is $0^+$, 2 is $1^+$, ...
- A predicate $\mathbf{N} : \iota o$ for the natural numbers is definable by higher-order quantification:

$$\lambda n : \iota . \forall p : \iota o . p\ 0 \wedge (\forall x . p\ x \ \rightarrow \ p\ (x \ \cup \ \{x\})) \rightarrow p\ \ n$$

- Theorem: $\forall n . \mathbf{N}\ \ n \rightarrow n \in \mathcal{U}\emptyset$
- Is this a fake theorem?
- "Real" abstract version:

$$\forall z : \iota . \forall S : \iota\iota . \forall n : \iota .$$
$$(\forall p : \iota o . p\ z \wedge (\forall x . p\ x \ \rightarrow \ p\ (S\ \ x)) \rightarrow p\ \ n) \rightarrow n \in \mathcal{U}\emptyset$$

- Abstract version is not a theorem. Specific is fake.

# Definition by Epsilon (Membership) Recursion

Functions from sets to sets can be defined by $\in$-recursion.
Suppose $\Phi : \iota(\iota\iota)\iota$ satisfies

$$\forall XFG.(\forall x.x \in X \to Fx = Gx) \to \Phi XF = \Phi XG.$$

Under this condition, $\Phi$ defines a function $\mathbf{R}\Phi$ satisfying

$$\forall X.\mathbf{R}\Phi X = \Phi X(\lambda x.\mathbf{R}\Phi x)$$

Technique (JAR 2015):

- Define $\mathbf{G}\Phi : \iota\iota o$ to be the least relation $R$ such that if

$$\forall x.x \in X \to Rx(Fx)$$

  then $RX(\Phi XF)$.
- Prove $\mathbf{G}\Phi$ is a total, functional relation.
- Use $\varepsilon$ to define the function $\mathbf{R}\Phi : \iota\iota$.

# Outline

# Basic Properties

- $\mathsf{P} : \iota\iota\iota$ $(s,t)$ means $\mathsf{P}st$
- $\forall xywz.(x,y) = (w,z) \equiv x = w \wedge y = z$

# Basic Properties

- $\mathbf{P} : \iota\iota\iota$ $(s, t)$ means $\mathbf{P}st$
- $\forall xywz.(x, y) = (w, z) \equiv x = w \wedge y = z$
- $\mathbf{L} : \iota(\iota\iota)\iota$ $\lambda x \in s.t$ means $\mathbf{L}s(\lambda x.t)$

# Basic Properties

- $\mathbf{P} : \iota\iota\iota$ $\quad (s, t)$ means $\mathbf{P}st$
- $\forall xywz.(x, y) = (w, z) \equiv x = w \wedge y = z$
- $\mathbf{L} : \iota(\iota\iota)\iota$ $\quad \lambda x \in s.t$ means $\mathbf{L}s(\lambda x.t)$
- $\forall XFG.$

$$(\forall x.x \in X \to Fx = Gx) \equiv (\lambda x \in X.Fx) = \lambda x \in X.Gx$$

# Basic Properties

- $\mathbf{P} : \iota\iota\iota$ $\quad$ $(s,t)$ means $\mathbf{P}st$
- $\forall xywz.(x,y) = (w,z) \equiv x = w \wedge y = z$
- $\mathbf{L} : \iota(\iota\iota)\iota$ $\quad$ $\lambda x \in s.t$ means $\mathbf{L}s(\lambda x.t)$
- $\forall XFG.$

$$(\forall x.x \in X \to Fx = Gx) \equiv (\lambda x \in X.Fx) = \lambda x \in X.Gx$$

- $\mathbf{Q}^{\Sigma} : \iota(\iota\iota)\iota$ $\quad$ $\Sigma x \in s.t$ means $\mathbf{Q}^{\Sigma}s(\lambda x.t)$
- $\forall XYz.z \in (\Sigma x \in X.Yx) \equiv$

$$\exists x.x \in X \wedge \exists y.y \in Yx \wedge z = (x,y)$$

# Basic Properties

- $\mathbf{P} : \iota\iota\iota$ $(s,t)$ means $\mathbf{P}st$
- $\forall xywz.(x,y) = (w,z) \equiv x = w \wedge y = z$
- $\mathbf{L} : \iota(\iota\iota)\iota$ $\lambda x \in s.t$ means $\mathbf{L}s(\lambda x.t)$
- $\forall XFG.$

$$(\forall x.x \in X \to Fx = Gx) \equiv (\lambda x \in X.Fx) = \lambda x \in X.Gx$$

- $\mathbf{Q}^{\Sigma} : \iota(\iota\iota)\iota$ $\Sigma x \in s.t$ means $\mathbf{Q}^{\Sigma}s(\lambda x.t)$
- $\forall XYz.z \in (\Sigma x \in X.Yx) \equiv$

$$\exists x.x \in X \wedge \exists y.y \in Yx \wedge z = (x,y)$$

- $\mathbf{Q}^{\Pi} : \iota(\iota\iota)\iota$ $\Pi x \in s.t$ means $\mathbf{Q}^{\Pi}s(\lambda x.t)$
- $\forall XYf.f \in (\Pi x \in X.Yx) \equiv$

$$\exists F.(\forall x.x \in X \to Fx \in Yx) \wedge f = \lambda x \in X.Fx$$

# Properties of Application

- $\mathbf{A} : \iota\iota\iota$ $\qquad st$ means $\mathbf{A}st$ when $s, t : \iota$
- Beta:

$$\forall XFx.x \in X \to (\lambda x \in X.Fx)x = Fx$$

- A typing-like property:

$$\forall XYfx.f \in (\Pi x \in X.Yx) \to x \in X \to fx \in Yx$$

# Avoiding and Exploiting Fake Theorems

- Since we can quantify over higher types and the specifications are propositions...
- a proposition can be stated without giving an implementation of pairs, functions, etc.
- "For all pairing operators, for all lambda operators, etc., the property holds."

# Avoiding and Exploiting Fake Theorems

- Since we can quantify over higher types and the specifications are propositions...
- a proposition can be stated without giving an implementation of pairs, functions, etc.
- "For all pairing operators, for all lambda operators, etc., the property holds."
- Alternatively, we can prove a property using a specific implementation satisfying nice properties.
- This specific, potentially "fake" theorem, may still be useful to prove the abstract version.

# Translating from Dependent Type Theory

- Need representations of pairs, functions, dependent sums, dependent products and more.
- Each Type universe can be interpreted as a Grothendieck Universe $U$.
- Need to ensure that if $X \in U$ and $Yx \in U$ for $x \in X$, then $\Sigma x \in X.Yx$ and $\Pi x \in X.Yx$ are in $U$.
- Are these "fake theorems"?

# Translating from Dependent Type Theory

- Need representations of pairs, functions, dependent sums, dependent products and more.
- Each Type universe can be interpreted as a Grothendieck Universe $U$.
- Need to ensure that if $X \in U$ and $Yx \in U$ for $x \in X$, then $\Sigma x \in X.Yx$ and $\Pi x \in X.Yx$ are in $U$.
- Are these "fake theorems"?
- Yes, a bit fake.

# Translating from Dependent Type Theory

- Need representations of pairs, functions, dependent sums, dependent products and more.
- Each Type universe can be interpreted as a Grothendieck Universe $U$.
- Need to ensure that if $X \in U$ and $Yx \in U$ for $x \in X$, then $\Sigma x \in X.Yx$ and $\Pi x \in X.Yx$ are in $U$.
- Are these "fake theorems"?
- Yes, a bit fake.
- The universe Prop can be taken as $\{0, 1\}$, i.e. 2 or $\wp(1)$.
- Need to ensure that if $Yx \in \{0, 1\}$ for $x \in X$, then $\Pi x \in X.Yx$ is 0 or 1.
- Is this a "fake theorem"?

# Translating from Dependent Type Theory

- Need representations of pairs, functions, dependent sums, dependent products and more.
- Each Type universe can be interpreted as a Grothendieck Universe $U$.
- Need to ensure that if $X \in U$ and $Yx \in U$ for $x \in X$, then $\Sigma x \in X.Yx$ and $\Pi x \in X.Yx$ are in $U$.
- Are these "fake theorems"?
- Yes, a bit fake.
- The universe Prop can be taken as $\{0,1\}$, i.e. 2 or $\wp(1)$.
- Need to ensure that if $Yx \in \{0,1\}$ for $x \in X$, then $\Pi x \in X.Yx$ is 0 or 1.
- Is this a "fake theorem"?
- Yes. Not true for Graph representation of functions.

# Extra “Fake” Properties

- $\wp 1$ is closed under $\Pi$, for some $\Pi$.

$$\forall XY.(\forall x.x \in X \to Yx \in \wp 1) \to (\Pi x \in X.Yx) \in \wp 1$$

(This was Aczel's original motivation for his function representation.)

# Extra "Fake" Properties

- $\wp 1$ is closed under $\Pi$, for some $\Pi$.

$$\forall XY.(\forall x.x \in X \rightarrow Yx \in \wp 1) \rightarrow (\Pi x \in X.Yx) \in \wp 1$$

(This was Aczel's original motivation for his function representation.)

- Functions applied outside their domain give 0:

$$\forall XFx.x \notin X \rightarrow (\lambda x \in X.Fx)x = 0$$

# Extra "Fake" Properties

- $\wp 1$ is closed under $\Pi$, for some $\Pi$.

$$\forall XY.(\forall x.x \in X \to Yx \in \wp 1) \to (\Pi x \in X.Yx) \in \wp 1$$

  (This was Aczel's original motivation for his function representation.)
- Functions applied outside their domain give 0:

$$\forall XFx.x \notin X \to (\lambda x \in X.Fx)x = 0$$

- Pairs are functions with domain 2:

$$\forall F.(\lambda x \in 2.Fx) = (F0, F1)$$

# Extra "Fake" Properties

- $\wp 1$ is closed under $\Pi$, for some $\Pi$.

$$\forall XY.(\forall x.x \in X \to Yx \in \wp 1) \to (\Pi x \in X.Yx) \in \wp 1$$

  (This was Aczel's original motivation for his function representation.)
- Functions applied outside their domain give 0:

$$\forall XFx.x \notin X \to (\lambda x \in X.Fx)x = 0$$

- Pairs are functions with domain 2:

$$\forall F.(\lambda x \in 2.Fx) = (F0, F1)$$

- $\wp 1$ is closed under $\Sigma$.

$$\forall X.X \in \wp 1 \to \forall Y.(\forall x.x \in X \to Yx \in \wp 1) \to (\Sigma x \in X.Yx) \in \wp 1$$

# Consequences

The following are provable from the previous properties:

- $\forall X.X \times X = X^2$
  that is, $\forall X.(\Sigma x \in X.X) = \Pi x \in 2.X$
- $\forall xy.(x, y)0 = x$
- $\forall xy.(x, y)1 = y$

# Outline

# Pairs as Disjoint Sums

- Idea: $(X, Y)$ is $\{(0, x) | x \in X\} \cup \{(1, y) | y \in Y\}$
- Morse considered using disjoint sums for "class-level" pairs in 1965, but ultimately used a different implementation.

# Pairs as Disjoint Sums

- Idea: $(X, Y)$ is $\{(0, x) | x \in X\} \cup \{(1, y) | y \in Y\}$
- Morse considered using disjoint sums for "class-level" pairs in 1965, but ultimately used a different implementation.
- Problem: What are $(0, x)$ and $(1, y)$?

# Pairs as Disjoint Sums

- Idea: $(X, Y)$ is $\{(0, x)|x \in X\} \cup \{(1, y)|y \in Y\}$

- Morse considered using disjoint sums for "class-level" pairs in 1965, but ultimately used a different implementation.
- Problem: What are $(0, x)$ and $(1, y)$?
- We could use Kuratowski pairs inside the definition, but let's just have one kind of pair.

# Pairs as Disjoint Sums

- Idea: $(X, Y)$ is $\{(0, x)|x \in X\} \cup \{(1, y)|y \in Y\}$
- Morse considered using disjoint sums for "class-level" pairs in 1965, but ultimately used a different implementation.
- Problem: What are $(0, x)$ and $(1, y)$?
- We could use Kuratowski pairs inside the definition, but let's just have one kind of pair.
- Solution: First define $\mathbf{I}_0 : \iota\iota$ and $\mathbf{I}_1 : \iota\iota$ so that later $\mathbf{I}_0 x = (0, x)$ and $\mathbf{I}_1 y = (1, y)$.

# Pairs as Disjoint Sums

- Idea: $(X, Y)$ is $\{(0,x)|x \in X\} \cup \{(1,y)|y \in Y\}$

- Morse considered using disjoint sums for "class-level" pairs in 1965, but ultimately used a different implementation.
- Problem: What are $(0,x)$ and $(1,y)$?
- We could use Kuratowski pairs inside the definition, but let's just have one kind of pair.
- Solution: First define $\mathbf{I}_0 : \iota\iota$ and $\mathbf{I}_1 : \iota\iota$ so that later $\mathbf{I}_0 x = (0,x)$ and $\mathbf{I}_1 y = (1,y)$.
- Then: $(X, Y) := \{\mathbf{I}_0 x | x \in X\} \cup \{\mathbf{I}_1 y | y \in Y\}$

# Pairs as Disjoint Sums

- Define $\mathbf{I}_1$ by $\in$-recursion:

$$\mathbf{I}_1 X = \{0\} \cup \{\mathbf{I}_1 x | x \in X\}$$

# Pairs as Disjoint Sums

- Define $\mathbf{I}_1$ by $\in$-recursion:

$$\mathbf{I}_1 X = \{0\} \cup \{\mathbf{I}_1 x | x \in X\}$$

- Define $\mathbf{I}_0 : \iota\iota$ by $\lambda X.\{\mathbf{I}_1 x | x \in X\}$.

# Pairs as Disjoint Sums

- Define $\mathbf{I}_1$ by $\in$-recursion:

$$\mathbf{I}_1 X = \{0\} \cup \{\mathbf{I}_1 x | x \in X\}$$

- Define $\mathbf{I}_0 : \iota\iota$ by $\lambda X.\{\mathbf{I}_1 x | x \in X\}$.
- Easy: $\forall XY.\mathbf{I}_0 X \neq \mathbf{I}_1 Y$ and $\mathbf{I}_0 0 = 0$.

# Pairs as Disjoint Sums

- Define $\mathbf{I}_1$ by $\in$-recursion:

$$\mathbf{I}_1 X = \{0\} \cup \{\mathbf{I}_1 x | x \in X\}$$

- Define $\mathbf{I}_0 : \iota\iota$ by $\lambda X.\{\mathbf{I}_1 x | x \in X\}$.
- Easy: $\forall XY.\mathbf{I}_0 X \neq \mathbf{I}_1 Y$ and $\mathbf{I}_0 0 = 0$.
- Define a one-sided inverse $\mathbf{I}^- : \iota\iota$ recursively:

$$\mathbf{I}^- X = \{\mathbf{I}^- x | x \in X \setminus \{0\}\}$$

# Pairs as Disjoint Sums

- Define $\mathbf{I}_1$ by $\in$-recursion:

$$\mathbf{I}_1 X = \{0\} \cup \{\mathbf{I}_1 x | x \in X\}$$

- Define $\mathbf{I}_0 : \iota\iota$ by $\lambda X.\{\mathbf{I}_1 x | x \in X\}$.
- Easy: $\forall XY.\mathbf{I}_0 X \neq \mathbf{I}_1 Y$ and $\mathbf{I}_0 0 = 0$.
- Define a one-sided inverse $\mathbf{I}^- : \iota\iota$ recursively:

$$\mathbf{I}^- X = \{\mathbf{I}^- x | x \in X \setminus \{0\}\}$$

- $\forall X.\mathbf{I}^-(\mathbf{I}_1 X) = X$ by $\in$-induction.

# Pairs as Disjoint Sums

- Define $\mathsf{I}_1$ by $\in$-recursion:

$$\mathsf{I}_1 X = \{0\} \cup \{\mathsf{I}_1 x | x \in X\}$$

- Define $\mathsf{I}_0 : \iota\iota$ by $\lambda X.\{\mathsf{I}_1 x | x \in X\}$.
- Easy: $\forall XY.\mathsf{I}_0 X \neq \mathsf{I}_1 Y$ and $\mathsf{I}_0 0 = 0$.
- Define a one-sided inverse $\mathsf{I}^- : \iota\iota$ recursively:

$$\mathsf{I}^- X = \{\mathsf{I}^- x | x \in X \setminus \{0\}\}$$

- $\forall X.\mathsf{I}^-(\mathsf{I}_1 X) = X$ by $\in$-induction.
- $\forall X.\mathsf{I}^-(\mathsf{I}_0 X) = X$ follows.

# Pairs

- $(X, Y) := \{\mathbf{I}_0 x | x \in X\} \cup \{\mathbf{I}_1 y | y \in Y\}$

# Pairs

- $(X, Y) := \{\mathbf{I}_0 x | x \in X\} \cup \{\mathbf{I}_1 y | y \in Y\}$
- $(0, Y) = \emptyset \cup \{\mathbf{I}_1 y | y \in Y\} = \mathbf{I}_0 Y$
- $(1, Y) = \{\mathbf{I}_0 0\} \cup \{\mathbf{I}_1 y | y \in Y\} = \mathbf{I}_1 Y$

# Pairs

- $(X, Y) := \{\mathbf{I}_0 x | x \in X\} \cup \{\mathbf{I}_1 y | y \in Y\}$
- $(0, Y) = \emptyset \cup \{\mathbf{I}_1 y | y \in Y\} = \mathbf{I}_0 Y$
- $(1, Y) = \{\mathbf{I}_0 0\} \cup \{\mathbf{I}_1 y | y \in Y\} = \mathbf{I}_1 Y$
- $(0, 0) = 0$

# Functions

Usual Graph Representation:

$$\{(x,y)|y = Fx\}$$

Aczel Representation ("Trace" Representation, Lee-Werner):

$$\{(x,y)|y \in Fx\}$$

Define $\mathbf{L} : \iota(\iota\iota)\iota$ by

$$\lambda XF. \bigcup_{x \in X} \{(x,y)|y \in Fx\}$$

Define $\mathbf{A} : \iota\iota\iota$ by $\lambda fx.\{y|(x,y) \in f\}$

# Outline

# Properties

- $\forall XFxy.(x, y) \in (\lambda x \in X.Fx) \equiv x \in X \wedge y \in Fx$

# Properties

- $\forall XFxy.(x,y) \in (\lambda x \in X.Fx) \equiv x \in X \wedge y \in Fx$
- $\forall fxy.y \in fx \equiv (x,y) \in f$

# Properties

- $\forall XFxy.(x,y) \in (\lambda x \in X.Fx) \equiv x \in X \wedge y \in Fx$
- $\forall fxy.y \in fx \equiv (x,y) \in f$
- Beta: $\forall XFx.x \in X \rightarrow (\lambda x \in X.Fx)x = Fx$
- $\forall XFx.x \notin X \rightarrow (\lambda x \in X.Fx)x = 0$

# Properties

- $\forall XFxy.(x, y) \in (\lambda x \in X.Fx) \equiv x \in X \wedge y \in Fx$
- $\forall fxy.y \in fx \equiv (x, y) \in f$
- Beta: $\forall XFx.x \in X \rightarrow (\lambda x \in X.Fx)x = Fx$
- $\forall XFx.x \notin X \rightarrow (\lambda x \in X.Fx)x = 0$
- $\forall F.(\lambda z \in 2.Fz) = (F0, F1)$

# Properties

- $\forall XFxy.(x,y) \in (\lambda x \in X.Fx) \equiv x \in X \wedge y \in Fx$
- $\forall fxy.y \in fx \equiv (x,y) \in f$
- Beta: $\forall XFx.x \in X \rightarrow (\lambda x \in X.Fx)x = Fx$
- $\forall XFx.x \notin X \rightarrow (\lambda x \in X.Fx)x = 0$
- $\forall F.(\lambda z \in 2.Fz) = (F0, F1)$
- $\forall xy.(x,y)0 = x$ and $\forall xy.(x,y)1 = y$

# Properties

- $\forall XFxy.(x,y) \in (\lambda x \in X.Fx) \equiv x \in X \wedge y \in Fx$
- $\forall fxy.y \in fx \equiv (x,y) \in f$
- Beta: $\forall XFx.x \in X \rightarrow (\lambda x \in X.Fx)x = Fx$
- $\forall XFx.x \notin X \rightarrow (\lambda x \in X.Fx)x = 0$
- $\forall F.(\lambda z \in 2.Fz) = (F0, F1)$
- $\forall xy.(x,y)0 = x$ and $\forall xy.(x,y)1 = y$
- $\forall xyi.i \notin 2 \rightarrow (x,y)i = 0$

# Sums and Products

- Define $\mathbf{Q}^{\Sigma}$ to be $\mathbf{L}$ since $\forall XFz$.

$$z \in (\lambda x \in X.Fx) \equiv \exists x.x \in X \wedge \exists y.y \in Fx \wedge z = (x, y)$$

# Sums and Products

- Define $\mathbf{Q}^{\Sigma}$ to be $\mathbf{L}$ since $\forall XFz$.

$$z \in (\lambda x \in X.Fx) \equiv \exists x.x \in X \wedge \exists y.y \in Fx \wedge z = (x, y)$$

- "Sigma is lambda." $\Sigma x \in s.t$ is $\lambda x \in s.t$

# Sums and Products

- Define $\mathbf{Q}^{\Sigma}$ to be $\mathbf{L}$ since $\forall XFz$.

$$z \in (\lambda x \in X.Fx) \equiv \exists x.x \in X \wedge \exists y.y \in Fx \wedge z = (x, y)$$

- "Sigma is lambda." $\Sigma x \in s.t$ is $\lambda x \in s.t$

- Define $\mathbf{Q}^{\Pi}$ to be

$$\lambda XY.\{f \in \wp(\Sigma x \in X.\bigcup(Yx)) | \forall x.x \in X \rightarrow fx \in Yx\}$$

# Sums and Products

- Define $\mathbf{Q}^{\Sigma}$ to be $\mathbf{L}$ since $\forall X F z$.

$$z \in (\lambda x \in X.Fx) \equiv \exists x.x \in X \wedge \exists y.y \in Fx \wedge z = (x, y)$$

- "Sigma is lambda." $\Sigma x \in s.t$ is $\lambda x \in s.t$

- Define $\mathbf{Q}^{\Pi}$ to be

$$\lambda XY.\{f \in \wp(\Sigma x \in X.\bigcup(Yx)) | \forall x.x \in X \to fx \in Yx\}$$

- $s \times t$ means $\Sigma x : s.t$ where $x$ is not free in $t$.
- $t^s$ means $\Pi x : s.t$ where $x$ is not free in $t$.

# Sums and Products

- Define $\mathbf{Q}^{\Sigma}$ to be $\mathbf{L}$ since $\forall XFz$.

$$z \in (\lambda x \in X.Fx) \equiv \exists x.x \in X \wedge \exists y.y \in Fx \wedge z = (x, y)$$

- "Sigma is lambda." $\Sigma x \in s.t$ is $\lambda x \in s.t$
- Define $\mathbf{Q}^{\Pi}$ to be

$$\lambda XY.\{f \in \wp(\Sigma x \in X.\bigcup(Yx)) | \forall x.x \in X \rightarrow fx \in Yx\}$$

- $s \times t$ means $\Sigma x : s.t$ where $x$ is not free in $t$.
- $t^s$ means $\Pi x : s.t$ where $x$ is not free in $t$.
- The properties mentioned earlier follow.
- In particular: $X \times X = X^{\{0,1\}}$.

# Monotonicity Properties

- If $X \subseteq Y$ and $\forall x.x \in X \to Zx \subseteq Wx$, then

$$(\Sigma x \in X.Zx) \subseteq \Sigma y \in Y.Wy.$$

# Monotonicity Properties

- If $X \subseteq Y$ and $\forall x.x \in X \to Zx \subseteq Wx$, then
$$(\Sigma x \in X.Zx) \subseteq \Sigma y \in Y.Wy.$$
- If $X \subseteq W$ and $Y \subseteq Z$, then $(X, Y) \subseteq (W, Z)$.

# Monotonicity Properties

- If $X \subseteq Y$ and $\forall x.x \in X \to Zx \subseteq Wx$, then
$$(\Sigma x \in X.Zx) \subseteq \Sigma y \in Y.Wy.$$
- If $X \subseteq W$ and $Y \subseteq Z$, then $(X, Y) \subseteq (W, Z)$.
- Codomain Covariance: If $\forall x.x \in X \to Ax \subseteq Bx$, then
$$(\Pi x \in X.Ax) \subseteq \Pi x \in X.Bx.$$

# Monotonicity Properties

- If $X \subseteq Y$ and $\forall x.x \in X \to Zx \subseteq Wx$, then
$$(\Sigma x \in X.Zx) \subseteq \Sigma y \in Y.Wy.$$
- If $X \subseteq W$ and $Y \subseteq Z$, then $(X, Y) \subseteq (W, Z)$.
- Codomain Covariance: If $\forall x.x \in X \to Ax \subseteq Bx$, then
$$(\Pi x \in X.Ax) \subseteq \Pi x \in X.Bx.$$
- *Domain Covariance:* If $X \subseteq Y$ and $\forall y.y \in Y \to y \notin X \to 0 \in Ay$, then
$$(\Pi x \in X.Ax) \subseteq \Pi y \in Y.Ay$$

# Monotonicity Properties

- If $X \subseteq Y$ and $\forall x.x \in X \to Zx \subseteq Wx$, then

$$(\Sigma x \in X.Zx) \subseteq \Sigma y \in Y.Wy.$$

- If $X \subseteq W$ and $Y \subseteq Z$, then $(X, Y) \subseteq (W, Z)$.
- Codomain Covariance: If $\forall x.x \in X \to Ax \subseteq Bx$, then

$$(\Pi x \in X.Ax) \subseteq \Pi x \in X.Bx.$$

- *Domain Covariance:* If $X \subseteq Y$ and $\forall y.y \in Y \to y \notin X \to 0 \in Ay$, then

$$(\Pi x \in X.Ax) \subseteq \Pi y \in Y.Ay$$

- Combined Result: If $\forall x.x \in X \to Ax \subseteq Bx$, $X \subseteq Y$ and $\forall y.y \in Y \to y \notin X \to 0 \in By$, then

$$(\Pi x \in X.Ax) \subseteq \Pi y \in Y.By$$

# Monotonicity Properties

- $A^0 = \{\emptyset\} = 1$
- If $0 \in A$, $n$ is a natural number and $m \in n$, then

$$A^m \subseteq A^n$$

# Monotonicity Properties

- $A^0 = \{\emptyset\} = 1$
- If $0 \in A$, $n$ is a natural number and $m \in n$, then

$$A^m \subseteq A^n$$

# Monotonicity Properties

- $A^0 = \{\emptyset\} = 1$
- If $0 \in A$, $n$ is a natural number and $m \in n$, then

$$A^m \subseteq A^n$$

- If $0 \in A$, then

$$1 = A^0 \subseteq A^1 \subseteq A^2 \subseteq A^3 \subseteq A^4 \subseteq \cdots$$

# Monotonicity Properties

- $A^0 = \{\emptyset\} = 1$
- If $0 \in A$, $n$ is a natural number and $m \in n$, then

$$A^m \subseteq A^n$$

- If $0 \in A$, then

$$1 = A^0 \subseteq A^1 \subseteq A^2 \subseteq A^3 \subseteq A^4 \subseteq \cdots$$

- Don't get greedy: $A^1 \neq A$.

# Monotonicity Properties

- $A^0 = \{\emptyset\} = 1$
- If $0 \in A$, $n$ is a natural number and $m \in n$, then

$$A^m \subseteq A^n$$

- If $0 \in A$, then

$$1 = A^0 \subseteq A^1 \subseteq A^2 \subseteq A^3 \subseteq A^4 \subseteq \cdots$$

- Don't get greedy: $A^1 \neq A$.
- *Embrace the fake theorems.*

# Outline

# Conclusion

- Combining HOL with ZF allows us to state theorems generically, avoiding representation issues.
- Or...we can choose nonstandard representations, e.g.:
- Pairs and functions can be represented so that pairs are functions from 2 $X \times X = X^2$

# Conclusion

- Combining HOL with ZF allows us to state theorems generically, avoiding representation issues.
- Or...we can choose nonstandard representations, e.g.:
- Pairs and functions can be represented so that pairs are functions from 2 $X \times X = X^2$
- ...and other "fake theorems" / surprising properties.

# Conclusion

- Combining HOL with ZF allows us to state theorems generically, avoiding representation issues.
- Or...we can choose nonstandard representations, e.g.:
- Pairs and functions can be represented so that pairs are functions from 2 $X \times X = X^2$
- ...and other "fake theorems" / surprising properties.
- The representations may be more convenient for formalized mathematics than the usual Kuratowski pairs and "functions as graphs" representations.

# References


- Peter Aczel. The type theoretic interpretation of constructive set theory. 1978.
- Chad E. Brown. Reconsidering Pairs and Functions as Sets. 2015.
- Alonzo Church. A formulation of the simple theory of types. 1940.
- Grzegorz Bancerek. Algebra of morphisms. 1997, 2003.
- Mike Gordon. Set Theory, Higher Order Logic or Both? 1996
- Gyesik Lee, Benjamin Werner. Proof-irrelevant model of CC with predicative induction and judgmental equality. 2011.
- Anthony P. Morse. A Theory of Sets. 1965.
- Lawrence C. Paulson. Set theory for verification: I. from foundations to functions. 1993.
- Patrick Suppes. Axiomatic Set Theory. 1972.
- Zermelo. Über Grenzzahlen und Mengenbereiche. 1930.