Developments, Libraries and Automated Theorem Provers

Chad E. Brown

Czech Technical University in Prague

July 2016

Outline

Introduction

2009- SATALLAX

Higher Order Automated Theorem Prover

Introduction

1997-2004 TPS (Peter Andrews' HOATP)

2009- SATALLAX

Higher Order Automated Theorem Prover

Introduction

1997-2004 TPS (Peter Andrews' HOATP)

2009- SATALLAX

Higher Order Automated Theorem Prover

2012-2014 Egal

Proof Checker, Higher Order Tarski-Grothendieck Set Theory

Introduction

1997-2004 TPS (Peter Andrews' HOATP)

2005-2007 Scunak

Proof Checker, Interactive Prover

Dependently Typed Set Theory, Pf Terms, Pf Irrelevance

2009- SATALLAX

Higher Order Automated Theorem Prover

2012-2014 Egal

Proof Checker, Higher Order Tarski-Grothendieck Set Theory

Introduction

early 1990s — Math Student
Shouldn't computers be able to do this?

1997-2004 — TPS (Peter Andrews' HOATP)

2005-2007 — Scunak
Proof Checker, Interactive Prover
Dependently Typed Set Theory, Pf Terms, Pf Irrelevance

2009- — SATALLAX
Higher Order Automated Theorem Prover

2012-2014 — Egal
Proof Checker, Higher Order Tarski-Grothendieck Set Theory

Introduction

early 1990s
Math Student
Shouldn't computers be able to do this?

mid 1990s
Otter (McCune)? Foundation? NBG (Quaife)?

1997-2004 TPS (Peter Andrews' HOATP)

2005-2007 Scunak
Proof Checker, Interactive Prover
Dependently Typed Set Theory, Pf Terms, Pf Irrelevance

2009- SATALLAX
Higher Order Automated Theorem Prover

2012-2014 Egal
Proof Checker, Higher Order Tarski-Grothendieck Set Theory

Introduction

- What is the Universe of “Mathematics”?
- Formal Developments (Articles): Definitions, Theorems (with Proofs)

Introduction

- What is the Universe of "Mathematics"?
- Formal Developments (Articles):
  Definitions, Theorems (with Proofs)
- Global (Growing) Library of Definitions and Theorems

Introduction

- What is the Universe of “Mathematics”?
- Formal Developments (Articles):
  Definitions, Theorems (with Proofs)
- Global (Growing) Library of Definitions and Theorems
- Abstract Library as a Collection of Hashes

Introduction

- What is the Universe of "Mathematics"?
- Formal Developments (Articles):
  Definitions, Theorems (with Proofs)
- Global (Growing) Library of Definitions and Theorems
- Abstract Library as a Collection of Hashes
- Automation to Fill Proof Gaps

Outline

Iterative Conception of the Universe

- V_α sets as of stage α ("von Neumann hierarchy")

Iterative Conception of the Universe

- V_α sets as of stage α ("von Neumann hierarchy")
- $V_0 = \emptyset$
- $V_{\beta+1} = \mathscr{P}(V_\beta)$
- $V_\lambda = \bigcup_{\beta<\lambda} V_\beta$

Iterative Conception of the Universe

- V_α sets as of stage α ("von Neumann hierarchy")
- $V_0 = \emptyset$
- $V_{\beta+1} = \mathscr{P}(V_\beta)$
- $V_\lambda = \bigcup_{\beta<\lambda} V_\beta$
- $Y \subseteq V_\alpha \leftrightarrow Y \in V_{\alpha+1}$
- Rank of Y is the least α such that $Y \subseteq V_\alpha$.

Iterative Conception of the Universe

- V_α sets as of stage α ("von Neumann hierarchy")
- $V_0 = \emptyset$
- $V_{\beta+1} = \mathscr{P}(V_\beta)$
- $V_\lambda = \bigcup_{\beta<\lambda} V_\beta$
- $Y \subseteq V_\alpha \leftrightarrow Y \in V_{\alpha+1}$
- Rank of Y is the least α such that $Y \subseteq V_\alpha$.
- V_ω is the hereditarily finite sets

Iterative Conception of the Universe

- V_α sets as of stage α ("von Neumann hierarchy")
- $V_0 = \emptyset$
- $V_{\beta+1} = \mathscr{P}(V_\beta)$
- $V_\lambda = \bigcup_{\beta<\lambda} V_\beta$
- $Y \subseteq V_\alpha \leftrightarrow Y \in V_{\alpha+1}$
- Rank of Y is the least α such that $Y \subseteq V_\alpha$.
- V_ω is the hereditarily finite sets
- V_κ is a Grothendieck universe (satisfying ZF) if κ is strongly inaccessable

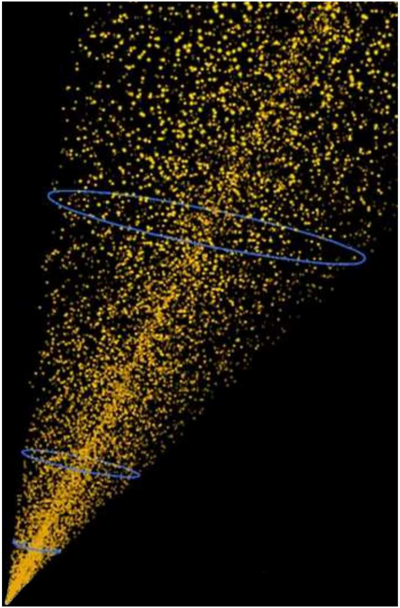
Iterative Conception of the Universe

- V_α sets as of stage α ("von Neumann hierarchy")
- $V_0 = \emptyset$
- $V_{\beta+1} = \mathscr{P}(V_\beta)$
- $V_\lambda = \bigcup_{\beta<\lambda} V_\beta$
- $Y \subseteq V_\alpha \leftrightarrow Y \in V_{\alpha+1}$
- Rank of Y is the least α such that $Y \subseteq V_\alpha$.
- V_ω is the hereditarily finite sets
- V_κ is a Grothendieck universe (satisfying ZF) if κ is strongly inaccessable
- *"Everything is a set."* E.g., pairs are sets: $(x, y) = \{\{x\}, \{x, y\}\}$

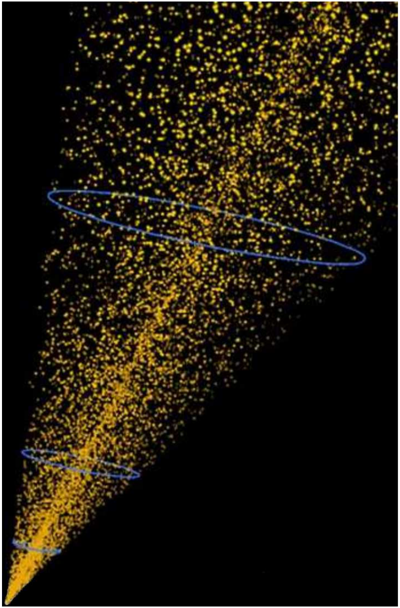
Iterative Conception of the Universe

- V_α sets as of stage α ("von Neumann hierarchy")
- $V_0 = \emptyset$
- $V_{\beta+1} = \mathscr{P}(V_\beta)$
- $V_\lambda = \bigcup_{\beta<\lambda} V_\beta$
- $Y \subseteq V_\alpha \leftrightarrow Y \in V_{\alpha+1}$
- Rank of Y is the least α such that $Y \subseteq V_\alpha$.
- V_ω is the hereditarily finite sets
- V_κ is a Grothendieck universe (satisfying ZF) if κ is strongly inaccessable
- *"Everything is a set."* E.g., pairs are sets: $(x, y) = \{\{x\}, \{x, y\}\}$
- A model of, e.g., Tarski-Grothendieck is a kind of "monster model" of mathematics.

Iterative Conception of the Universe

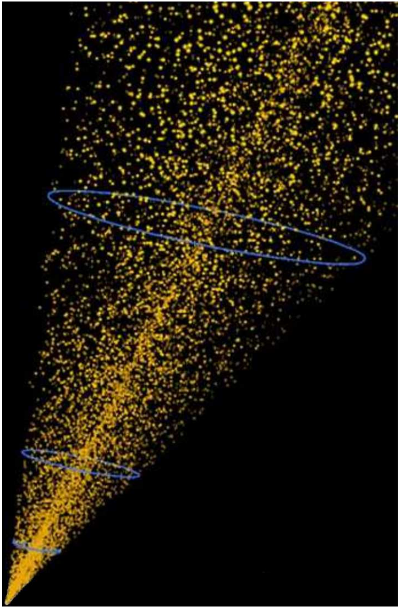


Iterative Conception of the Universe



$\emptyset$

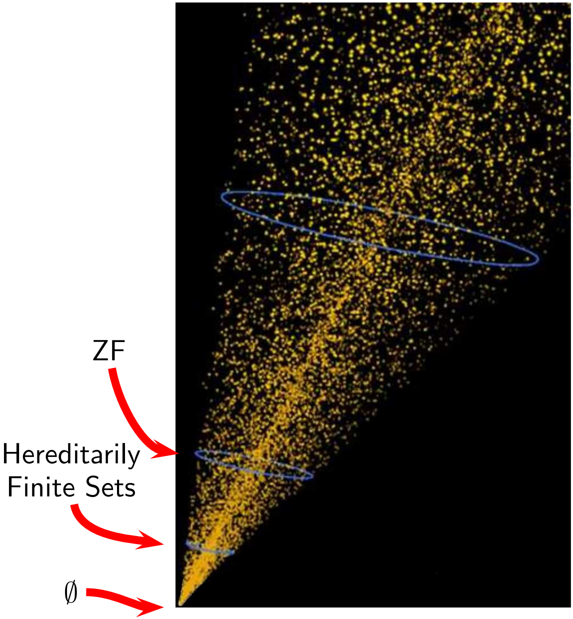
Iterative Conception of the Universe



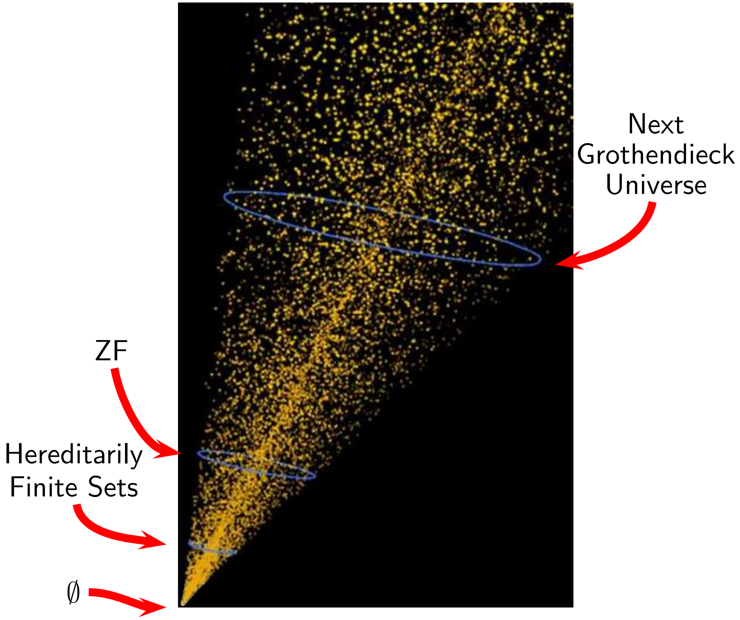
Hereditarily Finite Sets

$\emptyset$

Iterative Conception of the Universe



Iterative Conception of the Universe



Hedge

- Well, not everything is a set:
- The relation $\lambda xy.x \in y$ is not a set.
- The pairing operation $\lambda xy.(x, y)$ is not a set.

Hedge

- Well, not everything is a set:
- The relation $\lambda xy.x \in y$ is not a set.
- The pairing operation $\lambda xy.(x, y)$ is not a set.
- ι : sets

Hedge

- Well, not everything is a set:
- The relation $\lambda xy.x \in y$ is not a set.
- The pairing operation $\lambda xy.(x, y)$ is not a set.
- ι : sets
- $\iota \to \cdots \to \iota \to o$: predicates/classes
- $\iota \to \cdots \to \iota \to \iota$: metafunctions

Outline

A Small Mathematical Development

- Define an operation I on sets such that

$$I(Y) = \{I(x) | x \in Y\} \cup \{\emptyset\}$$

A Small Mathematical Development

- Define an operation I on sets such that

$$I(Y) = \{I(x) | x \in Y\} \cup \{\emptyset\}$$

- Define an operation U on sets such that

$$U(Y) = \{U(x) | x \in Y \setminus \{\emptyset\}\}$$

A Small Mathematical Development

- Define an operation I on sets such that

$$I(Y) = \{I(x) | x \in Y\} \cup \{\emptyset\}$$

- Define an operation U on sets such that

$$U(Y) = \{U(x) | x \in Y \setminus \{\emptyset\}\}$$

- Both operations can be defined by $\in$-recursion.

A Small Mathematical Development

- Define an operation I on sets such that

$$I(Y) = \{I(x) | x \in Y\} \cup \{\emptyset\}$$

- Define an operation U on sets such that

$$U(Y) = \{U(x) | x \in Y \setminus \{\emptyset\}\}$$

- Both operations can be defined by $\in$-recursion.
- Prove $U(I(Y)) = Y$ for all sets Y.

A Small Mathematical Development

- Define an operation I on sets such that

$$I(Y) = \{I(x)|x \in Y\} \cup \{\emptyset\}$$

- Define an operation U on sets such that

$$U(Y) = \{U(x)|x \in Y \setminus \{\emptyset\}\}$$

- Both operations can be defined by $\in$-recursion.
- Prove $U(I(Y)) = Y$ for all sets Y.
- Proof: $\in$-induction.

A Small Mathematical Development

- Define an operation I on sets such that

$$I(Y) = \{I(x) | x \in Y\} \cup \{\emptyset\}$$

- Define an operation U on sets such that

$$U(Y) = \{U(x) | x \in Y \setminus \{\emptyset\}\}$$

- Both operations can be defined by $\in$-recursion.
- Prove $U(I(Y)) = Y$ for all sets Y.
- Proof: $\in$-induction.
- Goal: Formalize as a Mizar article.

A Small Mathematical Development

- Define an operation I on sets such that

$$I(Y) = \{I(x) | x \in Y\} \cup \{\emptyset\}$$

- Define an operation U on sets such that

$$U(Y) = \{U(x) | x \in Y \setminus \{\emptyset\}\}$$

- Both operations can be defined by $\in$-recursion.
- Prove $U(I(Y)) = Y$ for all sets Y.
- Proof: $\in$-induction.
- Goal: Formalize as a Mizar article.
- Question: Is this already in the MML?
  (Mizar Mathematical Library)

Local Developments vs a Global Library

Library

Developments
(Documents/Articles/Theories)
Local

Local Developments vs a Global Library

Global
(Theorems, Definitions)
Library

Developments
(Documents/Articles/Theories)
Local

Local Developments vs a Global Library

Global
(Theorems, Definitions)
Library
search

Developments
(Documents/Articles/Theories)
Local

Searching the Library

- MPTP (Urban) – FOF TPTP version of the MML
- Primary purpose: help users automatically complete Mizar proofs

Searching the Library

- MPTP (Urban) – FOF TPTP version of the MML
- Primary purpose: help users automatically complete Mizar proofs
- New HO version: – THF TPTP version of the MML (135K props [thms, defs, type hierarchy,. . .])

Searching the Library

- MPTP (Urban) – FOF TPTP version of the MML
- Primary purpose: help users automatically complete Mizar proofs
- New HO version: – THF TPTP version of the MML (135K props [thms, defs, type hierarchy,. . .])
- HO version can handle schemes and Fraenkel operators (See MKM Talk)

Searching the Library

- MPTP (Urban) – FOF TPTP version of the MML
- Primary purpose: help users automatically complete Mizar proofs
- New HO version: – THF TPTP version of the MML (135K props [thms, defs, type hierarchy,...])
- HO version can handle schemes and Fraenkel operators (See MKM Talk)
- HO provers (e.g., Satallax) can help Mizar users

Searching the Library

- MPTP (Urban) – FOF TPTP version of the MML
- Primary purpose: help users automatically complete Mizar proofs
- New HO version: – THF TPTP version of the MML (135K props [thms, defs, type hierarchy,...])
- HO version can handle schemes and Fraenkel operators (See MKM Talk)
- HO provers (e.g., Satallax) can help Mizar users
- Side effect: can search the THF version of the MML for theorems of a certain form

Searching the Library

- MPTP (Urban) – FOF TPTP version of the MML
- Primary purpose: help users automatically complete Mizar proofs
- New HO version: – THF TPTP version of the MML (135K props [thms, defs, type hierarchy,. . .])
- HO version can handle schemes and Fraenkel operators (See MKM Talk)
- HO provers (e.g., Satallax) can help Mizar users
- Side effect: can search the THF version of the MML for theorems of a certain form
- Satallax extended to search for "similar" theorems to a conjecture
- Search for all theorems of the form

$$\forall Y : \mathtt{set}. * (*(Y)) = Y$$

Searching the MML for U(I(Y))=Y

```
include('allofmizar-preface-types-axioms').

thf(invset,conjecture,
   (? [I: ($i>$i)] : (? [U: ($i>$i)] :
       (! [Y:$i] : ((U @ (I @ Y)) = Y))))).
```

Searching the MML for U(I(Y))=Y

```
include('allofmizar-preface-types-axioms').

thf(invset,conjecture,
   (? [I: ($i>$i)] : (? [U: ($i>$i)] :
       (! [Y:$i] : ((U @ (I @ Y)) = Y))))).
```

```
~/satallax/satallax.native -similar invset
```

Searching the MML for U(I(Y))=Y

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Number of conclusion variants: 3
Total partial matches: 2376
t2_jgraph_1 similarity measure 0
rd2_relat_1 similarity measure 0
t5_pcomps_1 similarity measure 0
t19_cohsp_1 similarity measure 0
t25_zfmisc_1 similarity measure 0
t19_yellow_1 similarity measure 0
t55_modelc_3 similarity measure 0
t10_setfam_1 similarity measure 0
t5_catalg_1 similarity measure 0
t2_cohsp_1 similarity measure 0
t80_exchsort similarity measure 0
rd1_relat_1 similarity measure 0
t81_zfmisc_1 similarity measure 0
Total of 13 of measure 0
t37_ordinal1 similarity measure 1
d26_fomodel0 similarity measure 1
t1_toler_1 similarity measure 1
d1_algstr_1 similarity measure 1
d3_subset_1 similarity measure 1
t8_card_3 similarity measure 1
rd2_xtuple_0 similarity measure 1
Total of 7 of measure 1
involutiveness_k4_measure6 similarity measure 2
t22_xboole_1 similarity measure 2
```

Searching the MML for U(I(Y))=Y

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Number of conclusion variants: 3
Total partial matches: 2376
t2_jgraph_1 similarity measure 0
rd2_relat_1 similarity measure 0
t5_pcomps_1 similarity measure 0
t19_cohsp_1 similarity measure 0
t25_zfmisc_1 similarity measure 0
t19_yellow_1 similarity measure 0
t55_modelc_3 similarity measure 0
t10_setfam_1 similarity measure 0
t5_catalg_1 similarity measure 0
t2_cohsp_1 similarity measure 0
t80_exchsort similarity measure 0
rd1_relat_1 similarity measure 0
t81_zfmisc_1 similarity measure 0
Total of 13 of measure 0
t37_ordinal1 similarity measure 1
d26_fomodel0 similarity measure 1
t1_toler_1 similarity measure 1
d1_algstr_1 similarity measure 1
d3_subset_1 similarity measure 1
t8_card_3 similarity measure 1
rd2_xtuple_0 similarity measure 1
Total of 7 of measure 1
involutiveness_k4_measure6 similarity measure 2
t22_xboole_1 similarity measure 2
```

13 have this form

Searching the MML for U(I(Y))=Y

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Number of conclusion variants: 3
Total partial matches: 2376
t2_jgraph_1 similarity measure 0
rd2_relat_1 similarity measure 0
t5_pcomps_1 similarity measure 0
t19_cohsp_1 similarity measure 0
t25_zfmisc_1 similarity measure 0
t19_yellow_1 similarity measure 0
t55_modelc_3 similarity measure 0
t10_setfam_1 similarity measure 0
t5_catalg_1 similarity measure 0
t2_cohsp_1 similarity measure 0
t80_exchsort similarity measure 0
rd1_relat_1 similarity measure 0
t81_zfmisc_1 similarity measure 0
Total of 13 of measure 0
t37_ordinal1 similarity measure 1
d26_fomodel0 similarity measure 1
t1_toler_1 similarity measure 1
d1_algstr_1 similarity measure 1
d3_subset_1 similarity measure 1
t8_card_3 similarity measure 1
rd2_xtuple_0 similarity measure 1
Total of 7 of measure 1
involutiveness_k4_measure6 similarity measure 2
t22_xboole_1 similarity measure 2
```

$$\bigcup\{x\} = x$$

13 have this form

Searching the MML for U(I(Y))=Y

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Number of conclusion variants: 3
Total partial matches: 2376
t2_jgraph_1 similarity measure 0
rd2_relat_1 similarity measure 0
t5_pcomps_1 similarity measure 0
t19_cohsp_1 similarity measure 0
t25_zfmisc_1 similarity measure 0
t19_yellow_1 similarity measure 0
t55_modelc_3 similarity measure 0
t10_setfam_1 similarity measure 0
t5_catalg_1 similarity measure 0
t2_cohsp_1 similarity measure 0
t80_exchsort similarity measure 0
rd1_relat_1 similarity measure 0
t81_zfmisc_1 similarity measure 0
Total of 13 of measure 0
t37_ordinal1 similarity measure 1
d26_fomodel0 similarity measure 1
t1_toler_1 similarity measure 1
d1_algstr_1 similarity measure 1
d3_subset_1 similarity measure 1
t8_card_3 similarity measure 1
rd2_xtuple_0 similarity measure 1
Total of 7 of measure 1
involutiveness_k4_measure6 similarity measure 2
t22_xboole_1 similarity measure 2
```

$$\bigcup\{x\} = x$$

13 have this form

$$\bigcup \mathscr{P} X = X$$

Searching the MML for U(I(Y))=Y

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Number of conclusion variants: 3
Total partial matches: 2376
t2_jgraph_1 similarity measure 0
rd2_relat_1 similarity measure 0
t5_pcomps_1 similarity measure 0
t19_cohsp_1 similarity measure 0
t25_zfmisc_1 similarity measure 0
t19_yellow_1 similarity measure 0
t55_modelc_3 similarity measure 0
t10_setfam_1 similarity measure 0
t5_catalg_1 similarity measure 0
t2_cohsp_1 similarity measure 0
t80_exchsort similarity measure 0
rd1_relat_1 similarity measure 0
t81_zfmisc_1 similarity measure 0
Total of 13 of measure 0
t37_ordinal1 similarity measure 1
d26_fomodel0 similarity measure 1
t1_toler_1 similarity measure 1
d1_algstr_1 similarity measure 1
d3_subset_1 similarity measure 1
t8_card_3 similarity measure 1
rd2_xtuple_0 similarity measure 1
Total of 7 of measure 1
involutiveness_k4_measure6 similarity measure 2
```

$$\bigcup\{x\} = x$$

13 have this form

$$\bigcup \mathscr{P} X = X$$

None of these 13 correspond to $U(I(Y)) = Y$ for my U and I.

Epsilon Induction in Mizar

$U(I(Y)) = Y$ can be proven by $\in$-induction.
$\in$-induction: For every property P of sets,

$$(\forall Y.(\forall x \in Y . P(x)) \to P(Y)) \to \forall Y.P(Y)$$

Epsilon Induction in Mizar

$U(I(Y)) = Y$ can be proven by $\in$-induction.
$\in$-induction: For every property P of sets,

$$(\forall Y.(\forall x \in Y . P(x)) \to P(Y)) \to \forall Y.P(Y)$$

In Mizar this needs to be a "scheme":

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
```

Epsilon Induction in Mizar

$U(I(Y)) = Y$ can be proven by $\in$-induction.
$\in$-induction: For every property P of sets,

$$(\forall Y.(\forall x \in Y\,.\,P(x)) \to P(Y)) \to \forall Y.P(Y)$$

In Mizar this needs to be a “scheme”:

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
```

- Is this already in the MML?
- Does it easily follow from the MML?

Searching the MML for ∈-induction

```
include('allofmizar-preface-types-axioms').

thf(epsind,conjecture,
   (! [P:($i>$o)] :
       ((! [Y:$i] : ((! [X:$i] : ((r2_hidden @ X @ Y) => (P @ X))) => (P @ Y)))
     => (! [Y:$i] : (P @ Y))))).
```

Searching the MML for $\in$-induction

```
include('allofmizar-preface-types-axioms').

thf(epsind,conjecture,
   (! [P:($i>$o)] :
        ((! [Y:$i] : ((! [X:$i] : ((r2_hidden @ X @ Y) => (P @ X))) => (P @ Y)))
      => (! [Y:$i] : (P @ Y))))).
```

```
~/satallax/satallax.native -similar epsind
```

Searching the MML for ∈-induction

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Number of conclusion variants: 2
Total partial matches: 92
s2_finset_1 similarity measure 3
s17_ordinal2 similarity measure 3
Total of 2 of measure 3
s1_card_1 similarity measure 4
s1_uniroots similarity measure 4
s1_fib_num2 similarity measure 4
s3_finseq_1 similarity measure 4
s2_fib_num2 similarity measure 4
s2_pre_poly similarity measure 4
s2_nat_d similarity measure 4
s5_funct_7 similarity measure 4
s3_afinsq_1 similarity measure 4
s1_card_fil similarity measure 4
s1_fib_num similarity measure 4
s1_zf_lang1 similarity measure 4
s1_modal_1 similarity measure 4
s2_ordinal1 similarity measure 4
s4_nat_1 similarity measure 4
s1_nat_1 similarity measure 4
s1_modelc_2 similarity measure 4
s2_nat_1 similarity measure 4
s10_nat_1 similarity measure 4
s1_glib_000 similarity measure 4
s1_ordinal2 similarity measure 4
s2_hilbert2 similarity measure 4
s3_cgames_1 similarity measure 4
s2_nat_2 similarity measure 4
s1_zf_lang similarity measure 4
s1_chord similarity measure 4
Total of 26 of measure 4
s4_int_1 similarity measure 5
```

Searching the MML for $\in$-induction

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Number of conclusion variants: 2
Total partial matches: 92
s2_finset_1 similarity measure 3
s17_ordinal2 similarity measure 3
Total of 2 of measure 3
s1_card_1 similarity measure 4
s1_uniroots similarity measure 4
s1_fib_num2 similarity measure 4
s3_finseq_1 similarity measure 4
s2_fib_num2 similarity measure 4
s2_pre_poly similarity measure 4
s2_nat_d similarity measure 4
s5_funct_7 similarity measure 4
s3_afinsq_1 similarity measure 4
s1_card_fil similarity measure 4
s1_fib_num similarity measure 4
s1_zf_lang1 similarity measure 4
s1_modal_1 similarity measure 4
s2_ordinal1 similarity measure 4
s4_nat_1 similarity measure 4
s1_nat_1 similarity measure 4
s1_modelc_2 similarity measure 4
s2_nat_1 similarity measure 4
s10_nat_1 similarity measure 4
s1_glib_000 similarity measure 4
s1_ordinal2 similarity measure 4
s2_hilbert2 similarity measure 4
s3_cgames_1 similarity measure 4
s2_nat_2 similarity measure 4
s1_zf_lang similarity measure 4
s1_chord similarity measure 4
Total of 26 of measure 4
s4_int_1 similarity measure 5
```

No Exact Matches, 92 Partial Matches

Searching the MML for ∈-induction

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Number of conclusion variants: 2
Total partial matches: 92
s2_finset_1 similarity measure 3
s17_ordinal2 similarity measure 3
Total of 2 of measure 3
s1_card_1 similarity measure 4
s1_uniroots similarity measure 4
s1_fib_num2 similarity measure 4
s3_finseq_1 similarity measure 4
s2_fib_num2 similarity measure 4
s2_pre_poly similarity measure 4
s2_nat_d similarity measure 4
s5_funct_7 similarity measure 4
s3_afinsq_1 similarity measure 4
s1_card_fil similarity measure 4
s1_fib_num similarity measure 4
s1_zf_lang1 similarity measure 4
s1_modal_1 similarity measure 4
s2_ordinal1 similarity measure 4
s4_nat_1 similarity measure 4
s1_nat_1 similarity measure 4
s1_modelc_2 similarity measure 4
s2_nat_1 similarity measure 4
s10_nat_1 similarity measure 4
s1_glib_000 similarity measure 4
s1_ordinal2 similarity measure 4
s2_hilbert2 similarity measure 4
s3_cgames_1 similarity measure 4
s2_nat_2 similarity measure 4
s1_zf_lang similarity measure 4
s1_chord similarity measure 4
Total of 26 of measure 4
s4_int_1 similarity measure 5
```

No Exact Matches, 92 Partial Matches

Finite Sets, No Help

Searching the MML for ∈-induction

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Number of conclusion variants: 2
Total partial matches: 92
s2_finset_1 similarity measure 3
s17_ordinal2 similarity measure 3
Total of 2 of measure 3
s1_card_1 similarity measure 4
s1_uniroots similarity measure 4
s1_fib_num2 similarity measure 4
s3_finseq_1 similarity measure 4
s2_fib_num2 similarity measure 4
s2_pre_poly similarity measure 4
s2_nat_d similarity measure 4
s5_funct_7 similarity measure 4
s3_afinsq_1 similarity measure 4
s1_card_fil similarity measure 4
s1_fib_num similarity measure 4
s1_zf_lang1 similarity measure 4
s1_modal_1 similarity measure 4
s2_ordinal1 similarity measure 4
s4_nat_1 similarity measure 4
s1_nat_1 similarity measure 4
s1_modelc_2 similarity measure 4
s2_nat_1 similarity measure 4
s10_nat_1 similarity measure 4
s1_glib_000 similarity measure 4
s1_ordinal2 similarity measure 4
s2_hilbert2 similarity measure 4
s3_cgames_1 similarity measure 4
s2_nat_2 similarity measure 4
s1_zf_lang similarity measure 4
s1_chord similarity measure 4
Total of 26 of measure 4
s4_int_1 similarity measure 5
```

No Exact Matches, 92 Partial Matches

Finite Sets, No Help

Natural Number Induction, No Help

Searching the MML for $\in$-induction

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Number of conclusion variants: 2
Total partial matches: 92
s2_finset_1 similarity measure 3
s17_ordinal2 similarity measure 3
Total of 2 of measure 3
s1_card_1 similarity measure 4
s1_uniroots similarity measure 4
s1_fib_num2 similarity measure 4
s3_finseq_1 similarity measure 4
s2_fib_num2 similarity measure 4
s2_pre_poly similarity measure 4
s2_nat_d similarity measure 4
s5_funct_7 similarity measure 4
s3_afinsq_1 similarity measure 4
s1_card_fil similarity measure 4
s1_fib_num similarity measure 4
s1_zf_lang1 similarity measure 4
s1_modal_1 similarity measure 4
s2_ordinal1 similarity measure 4
s4_nat_1 similarity measure 4
s1_nat_1 similarity measure 4
s1_modelc_2 similarity measure 4
s2_nat_1 similarity measure 4
s10_nat_1 similarity measure 4
s1_glib_000 similarity measure 4
s1_ordinal2 similarity measure 4
s2_hilbert2 similarity measure 4
s3_cgames_1 similarity measure 4
s2_nat_2 similarity measure 4
s1_zf_lang similarity measure 4
s1_chord similarity measure 4
Total of 26 of measure 4
s4_int_1 similarity measure 5
```

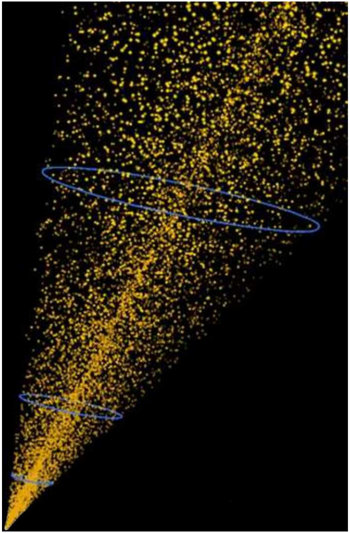
No Exact Matches, 92 Partial Matches

Finite Sets, No Help

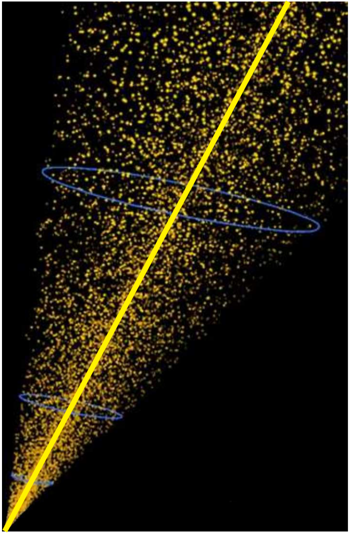
Natural Number Induction, No Help

Transfinite Induction!

Transfinite Induction

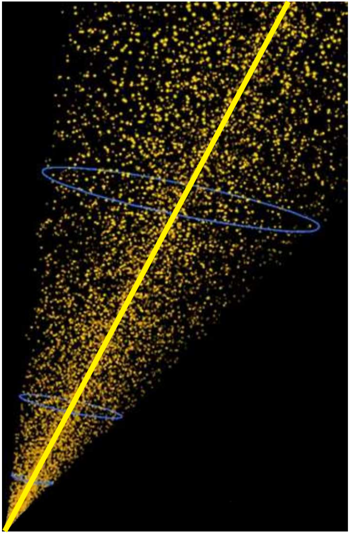


Transfinite Induction



Ordinals

Transfinite Induction

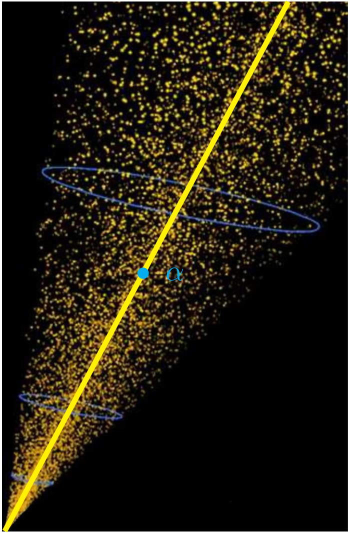


Ordinals
$P[Ordinal]$

$\rightarrow$

$\forall \alpha . P[\alpha]$

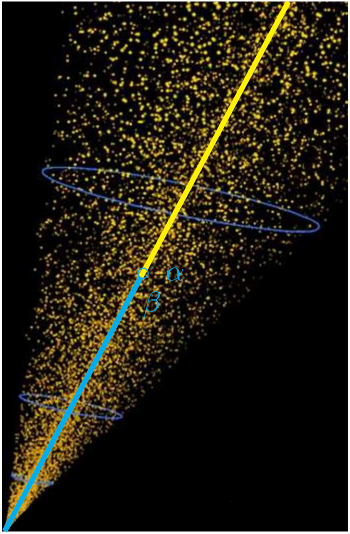
Transfinite Induction



Ordinals
$P[Ordinal]$

$$(\forall \alpha. \qquad \rightarrow P[\alpha])$$
$$\rightarrow$$
$$\forall \alpha . P[\alpha]$$

Transfinite Induction

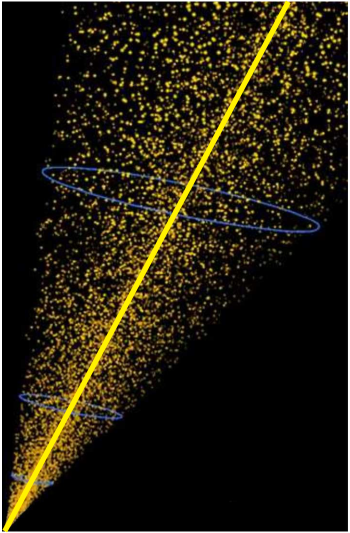


Ordinals

$P[Ordinal]$

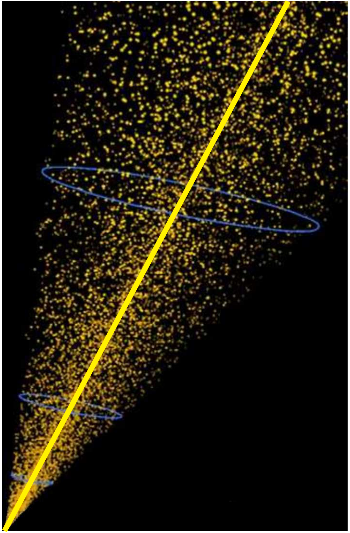
$$(\forall\alpha.(\forall\beta \in \alpha \,.\, P[\beta]) \to P[\alpha])$$
$$\to$$
$$\forall\alpha.P[\alpha]$$

Transfinite Induction and ∈-Induction



$P[set]$

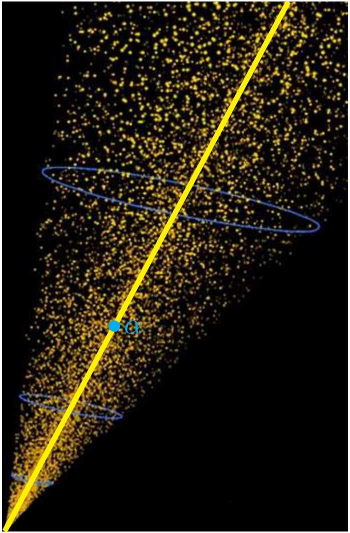
Transfinite Induction and $\in$-Induction



$$P[set]$$

Let $Q[\alpha]$ be $\forall Y. Y \subseteq V_\alpha \to P[Y]$.

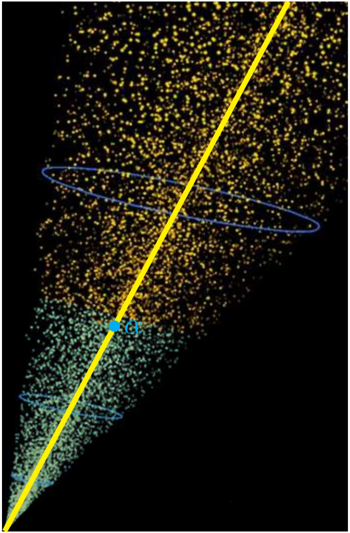
Transfinite Induction and $\in$-Induction



$$P[set]$$

Let $Q[\alpha]$ be $\forall Y. Y \subseteq V_\alpha \rightarrow P[Y]$.

Transfinite Induction and $\in$-Induction



$$P[set]$$

Let $Q[\alpha]$ be $\forall Y. Y \subseteq V_\alpha \to P[Y]$.

Transfinite Induction and $\in$-Induction

- To prove $\in$-induction we need:
- Transfinite Induction and two facts:

Transfinite Induction and $\in$-Induction

- To prove $\in$-induction we need:
- Transfinite Induction and two facts:
  1. If $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.

Transfinite Induction and $\in$-Induction

- To prove $\in$-induction we need:
- Transfinite Induction and two facts:
  1. If $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
  2. For every set Y, there is an α such that $Y \subseteq V_\alpha$.

Transfinite Induction and $\in$-Induction

- To prove $\in$-induction we need:
- Transfinite Induction and two facts:
  1. If $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
  2. For every set Y, there is an α such that $Y \subseteq V_\alpha$.
- Does the MML include V_α? (Yes...)

Transfinite Induction and $\in$-Induction

- To prove $\in$-induction we need:
- Transfinite Induction and two facts:
  1. If $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
  2. For every set Y, there is an α such that $Y \subseteq V_\alpha$.
- Does the MML include V_α? (Yes...)
- Search for theorems of the form $\forall Y.\exists \alpha. Y \subseteq *(\alpha)$.

Searching the MML for V

```
include('allofmizar-preface-types-axioms').

thf(invset,conjecture,
       (? [V: ($i>$i)] :
          (! [Y:$i] : (? [A:$i] : ((v3_ordinal1 @ A) & (r1_tarski @ Y @ (V @ A))))))).
```

Searching the MML for V

```
include('allofmizar-preface-types-axioms').

thf(invset,conjecture,
       (? [V: ($i>$i)] :
          (! [Y:$i] : (? [A:$i] : ((v3_ordinal1 @ A) & (r1_tarski @ Y @ (V @ A))))))).
```

```
~/satallax/satallax.native -similar rankprop2
```

Searching the MML for V

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Conjunctions (2)
imp (v3_ordinal1 *3) False
r1_tarski __2 (*1 *3)
Number of conclusion variants: 5
Total partial matches: 96
t62_classes1 similarity measure 0
Total of 1 of measure 0
eps_ax similarity measure 2
Total of 1 of measure 2
t11_ordinal3 similarity measure 3
t4_ordinal3 similarity measure 3
s2_finset_1 similarity measure 3
s17_ordinal2 similarity measure 3
Total of 4 of measure 3
s1_card_1 similarity measure 4
```

96 Partial Matches

Searching the MML for V

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Conjunctions (2)
imp (v3_ordinal1 *3) False
r1_tarski __2 (*1 *3)
Number of conclusion variants: 5
Total partial matches: 96
t62_classes1 similarity measure 0
Total of 1 of measure 0
eps_ax similarity measure 2
Total of 1 of measure 2
t11_ordinal3 similarity measure 3
t4_ordinal3 similarity measure 3
s2_finset_1 similarity measure 3
s17_ordinal2 similarity measure 3
Total of 4 of measure 3
s1_card_1 similarity measure 4
```

96 Partial Matches
Closest: CLASSES1:62

Searching the MML for V

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Conjunctions (2)
imp (v3_ordinal1 *3) False
r1_tarski __2 (*1 *3)
Number of conclusion variants: 5
Total partial matches: 96
t62_classes1 similarity measure 0
Total of 1 of measure 0
eps_ax similarity measure 2
Total of 1 of measure 2
t11_ordinal3 similarity measure 3
t4_ordinal3 similarity measure 3
s2_finset_1 similarity measure 3
s17_ordinal2 similarity measure 3
Total of 4 of measure 3
s1_card_1 similarity measure 4
```

96 Partial Matches

Closest: CLASSES1:62

```
theorem :: CLASSES1:62
  ex A st X c= Rank A;
```

Searching the MML for V

```
Read it, initial_branch has 135002 elts
Searching for similar formulas.
Conjunctions (2)
imp (v3_ordinal1 *3) False
r1_tarski __2 (*1 *3)
Number of conclusion variants: 5
Total partial matches: 96
t62_classes1 similarity measure 0
Total of 1 of measure 0
eps_ax similarity measure 2
Total of 1 of measure 2
t11_ordinal3 similarity measure 3
t4_ordinal3 similarity measure 3
s2_finset_1 similarity measure 3
s17_ordinal2 similarity measure 3
Total of 4 of measure 3
s1_card_1 similarity measure 4
```

96 Partial Matches
Closest: CLASSES1:62

```
theorem :: CLASSES1:62
  ex A st X c= Rank A;
```

Rank A corresponds to V_α.

Importing

Global
(Theorems, Definitions)
Library

Developments
(Documents/Articles/Theories)
Local

Importing

```
environ

 vocabularies TARSKI, ORDINAL1, CLASSES1;
 notations TARSKI, ORDINAL1, CLASSES1;
 constructors CLASSES1;
 theorems CLASSES1;
 schemes ORDINAL1;
```

Global
(Theorems, Definitions)
Library

import

Developments
(Documents/Articles/Theories)
Local

Importing

```
environ

  vocabularies TARSKI, ORDINAL1, CLASSES1;
  notations TARSKI, ORDINAL1, CLASSES1;
  constructors CLASSES1;
  theorems CLASSES1;
  schemes ORDINAL1;
```

Global
(Theorems, Definitions)
Library

import

Developments
(Documents/Articles/Theories)
Local

TARSKI: $\subseteq$
ORDINAL1: Ordinals, Transfinite Induction Scheme
CLASSES1: Rank, $\forall Y.\exists\alpha.Y \subseteq V_\alpha$

Partial Proof

scheme

$$\forall Y . P[Y]$$

$$\forall Y . (\forall x \in Y . P[x]) \to P[Y]$$

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
```

$\forall Y . P[Y]$

$$\forall Y . (\forall x \in Y . P[x]) \to P[Y]$$

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
```

Definition: $Q[\alpha] := \forall Y. Y \subseteq V_\alpha \to P[Y]$

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
```

Definition: $Q[\alpha] := \forall Y. Y \subseteq V_\alpha \rightarrow P[Y]$

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
```

$$\forall \alpha \,.\, Q[\alpha]$$

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
```

$$\forall \alpha \, . \, Q[\alpha]$$

```
  A3: for A holds Q[A]
```

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];




  A3: for A holds Q[A]
  let Y;
```

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
```

```
  A3: for A holds Q[A]
  let Y;
```

$\exists \alpha . Y \subseteq V_\alpha$ by "Fact 2"

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];




  A3: for A holds Q[A]
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
```

$\exists\alpha.Y \subseteq V_\alpha$ by "Fact 2"

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
```

```
  A3: for A holds Q[A]
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
```

$P[Y]$ since $Q[\alpha]$ and $Y \subseteq V_\alpha$

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];



  A3: for A holds Q[A]
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
  thus P[Y] by A3,A4;
end;
```

$P[Y]$ since $Q[\alpha]$ and $Y \subseteq V_\alpha$

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];




  A3: for A holds Q[A]
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
  thus P[Y] by A3,A4;
end;
```

(by Transfinite Induction)

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
```

A2: $\forall\alpha\,.\,(\forall\beta\in\alpha\,.\,Q[\beta])\rightarrow Q[\alpha]$

```
  A3: for A holds Q[A] from ORDINAL1:sch 2(A2);
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
  thus P[Y] by A3,A4;
end;
```

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
  A2: for A st for B st B in A holds Q[B] holds Q[A]
  proof
      :: something to prove
  end;
  A3: for A holds Q[A] from ORDINAL1:sch 2(A2);
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
  thus P[Y] by A3,A4;
end;
```

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
  A2: for A st for B st B in A holds Q[B] holds Q[A]
  proof
      :: something to prove
  end;
  A3: for A holds Q[A] from ORDINAL1:sch 2(A2);
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
  thus P[Y] by A3,A4;
end;
```

- Remaining subproof: By A1 and "Fact 1."

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
  A2: for A st for B st B in A holds Q[B] holds Q[A]
  proof
      :: something to prove
  end;
  A3: for A holds Q[A] from ORDINAL1:sch 2(A2);
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
  thus P[Y] by A3,A4;
end;
```

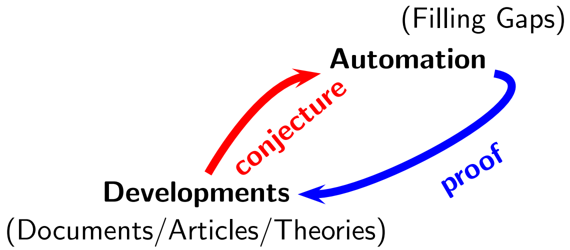
- Remaining subproof: By A1 and "Fact 1."
- Fact 1 can be proven as a lemma by another transfinite induction.

Partial Proof

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
  A2: for A st for B st B in A holds Q[B] holds Q[A]
  proof
      :: something to prove
  end;
  A3: for A holds Q[A] from ORDINAL1:sch 2(A2);
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
  thus P[Y] by A3,A4;
end;
```

- Remaining subproof: By A1 and "Fact 1."
- Fact 1 can be proven as a lemma by another transfinite induction.
- Given Fact 1, the gap can be completed by local reasoning. Automation?

Automation


(Filling Gaps)
Automation
conjecture
proof
Developments
(Documents/Articles/Theories)

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y \,.\, P[x]) \to P[Y]$
- Definition: $Q[\alpha] := \forall Y.Y \subseteq V_\alpha \to P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \to Q[\beta]) \to Q[\alpha]$

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y . P[x]) \to P[Y]$
- Definition: $Q[\alpha] := \forall Y.Y \subseteq V_\alpha \to P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \to Q[\beta]) \to Q[\alpha]$
- MPTP interface can try to automatically prove goals:

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y . P[x]) \rightarrow P[Y]$
- Definition: $Q[\alpha] := \forall Y.Y \subseteq V_\alpha \rightarrow P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \rightarrow Q[\beta]) \rightarrow Q[\alpha]$
- MPTP interface can try to automatically prove goals:

```
A2: for A st for B st B in A holds Q[B] holds Q[A] by;
```

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y . P[x]) \to P[Y]$
- Definition: $Q[\alpha] := \forall Y.Y \subseteq V_\alpha \to P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \to Q[\beta]) \to Q[\alpha]$
- MPTP interface can try to automatically prove goals:

```
A2: for A st for B st B in A holds Q[B] holds Q[A] ; :: ATP asked ...
```

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y . P[x]) \to P[Y]$
- Definition: $Q[\alpha] := \forall Y.Y \subseteq V_\alpha \to P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \to Q[\beta]) \to Q[\alpha]$
- MPTP interface can try to automatically prove goals:

```
A2: for A st for B st B in A holds Q[B] holds Q[A] by EPSIND3E:1,A1,A2;
```

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y \,.\, P[x]) \to P[Y]$
- Definition: $Q[\alpha] := \forall Y.Y \subseteq V_\alpha \to P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \to Q[\beta]) \to Q[\alpha]$
- MPTP interface can try to automatically prove goals:

```
A2: for A st for B st B in A holds Q[B] holds Q[A] by EPSIND3E:1,A1,A2;
```

```
  A2: for A st for B st B in A holds Q[B] holds Q[A] by Th1,A1;
::>                                                       *4
```

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y . P[x]) \to P[Y]$
- Definition: $Q[\alpha] := \forall Y.Y \subseteq V_\alpha \to P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \to Q[\beta]) \to Q[\alpha]$

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y . P[x]) \rightarrow P[Y]$
- Definition: $Q[\alpha] := \forall Y . Y \subseteq V_\alpha \rightarrow P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \rightarrow Q[\beta]) \rightarrow Q[\alpha]$
- Satallax can prove the corresponding HO THF problem in $< 0.1s$.

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y . P[x]) \rightarrow P[Y]$
- Definition: $Q[\alpha] := \forall Y . Y \subseteq V_\alpha \rightarrow P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \rightarrow Q[\beta]) \rightarrow Q[\alpha]$
- Satallax can prove the corresponding HO THF problem in $< 0.1s$.
- Satallax can produce various kinds of proofs.
  - Coq scripts (using tactics to simulate tableau rules)

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y\,.\,P[x]) \to P[Y]$
- Definition: $Q[\alpha] := \forall Y.Y \subseteq V_\alpha \to P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \to Q[\beta]) \to Q[\alpha]$
- Satallax can prove the corresponding HO THF problem in $< 0.1s$.
- Satallax can produce various kinds of proofs.
  - Coq scripts (using tactics to simulate tableau rules)
  - Coq proof terms (using lemmas corresponding to tableau rules)

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y . P[x]) \to P[Y]$
- Definition: $Q[\alpha] := \forall Y.Y \subseteq V_\alpha \to P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \to Q[\beta]) \to Q[\alpha]$
- Satallax can prove the corresponding HO THF problem in $< 0.1s$.
- Satallax can produce various kinds of proofs.
  - Coq scripts (using tactics to simulate tableau rules)
  - Coq proof terms (using lemmas corresponding to tableau rules)
  - But not Mizar proofs (yet?)

Filling the Gap

- Lemma: For Y and α, if $Y \subseteq V_\alpha$ and $x \in Y$, then there is some $\beta \in \alpha$ such that $x \subseteq V_\beta$.
- Assumption: $\forall Y.(\forall x \in Y . P[x]) \to P[Y]$
- Definition: $Q[\alpha] := \forall Y.Y \subseteq V_\alpha \to P[Y]$
- Goal: $\forall \alpha.(\forall \beta.\beta \in \alpha \to Q[\beta]) \to Q[\alpha]$
- Satallax can prove the corresponding HO THF problem in $< 0.1s$.
- Satallax can produce various kinds of proofs.
  - Coq scripts (using tactics to simulate tableau rules)
  - Coq proof terms (using lemmas corresponding to tableau rules)
  - But not Mizar proofs (yet?)
  - Mizar could be extended to allow "byproofterm" justifications with a small Curry-Howard-de Bruijn style proof checker.

Filling the Gap (Pf Term)

Satallax Produced Proof Term for the Gap:

```
exact (NNPP (forall (X1:set), v3_ordinal1 X1 -> (forall (X2:set), v3_ordinal1 X2 -> r2_hidden X2 X1 -> (forall (X3:set), r1_tars
ki X3 (k4_classes1 X2) -> p X3)) -> (forall (X2:set), r1_tarski X2 (k4_classes1 X1) -> p X2)) (fun H0 => TNAll set (fun (X1:set)
 => v3_ordinal1 X1 -> (forall (X2:set), v3_ordinal1 X2 -> r2_hidden X2 X1 -> q X2) -> q X1) H0 (fun __0 H1 => TNImp (v3_ordinal1
 __0) ((forall (X1:set), v3_ordinal1 X1 -> r2_hidden X1 __0 -> q X1) -> q __0) H1 (fun H2 H3 => TNImp (forall (X1:set), v3_ordin
al1 X1 -> r2_hidden X1 __0 -> q X1) (q __0) H3 (fun H4 H5 => TNAll set (fun (X1:set) => r1_tarski X1 (k4_classes1 __0) -> p X1)
H5 (fun __1 H6 => TNImp (r1_tarski __1 (k4_classes1 __0)) (p __1) H6 (fun H7 H8 => TAll set (fun (X1:set) => forall (X2:set), v3
_ordinal1 X2 -> r1_tarski X1 (k4_classes1 X2) -> (forall (X3:set), r2_hidden X3 X1 -> (exists X4:set, v3_ordinal1 X4 /\ r2_hidde
n X4 X2 /\ r1_tarski X3 (k4_classes1 X4)))) th1 __1 (fun H9 => TAll set (fun (X1:set) => v3_ordinal1 X1 -> r1_tarski __1 (k4_cla
sses1 X1) -> (forall (X2:set), r2_hidden X2 __1 -> (exists X3:set, v3_ordinal1 X3 /\ r2_hidden X3 X1 /\ r1_tarski X2 (k4_classes
1 X3)))) H9 __0 (fun H10 => TAll set (fun (X1:set) => (forall (X2:set), r2_hidden X2 X1 -> p X2) -> p X1) a1 __1 (fun H11 => TIm
p (v3_ordinal1 __0) (r1_tarski __1 (k4_classes1 __0) -> (forall (X1:set), r2_hidden X1 __1 -> (exists X2:set, v3_ordinal1 X2 /\
r2_hidden X2 __0 /\ r1_tarski X1 (k4_classes1 X2)))) H10 (fun H12 => H12 H2) (fun H12 => TImp (r1_tarski __1 (k4_classes1 __0))
(forall (X1:set), r2_hidden X1 __1 -> (exists X2:set, v3_ordinal1 X2 /\ r2_hidden X2 __0 /\ r1_tarski X1 (k4_classes1 X2))) H12
(fun H13 => H13 H7) (fun H13 => TImp (forall (X1:set), r2_hidden X1 __1 -> p X1) (p __1) H11 (fun H14 => TNAll set (fun (X1:set)
 => r2_hidden X1 __1 -> p X1) H14 (fun __4 H15 => TNImp (r2_hidden __4 __1) (p __4) H15 (fun H16 H17 => TAll set (fun (X1:set) =
> r2_hidden X1 __1 -> (exists X2:set, v3_ordinal1 X2 /\ r2_hidden X2 __0 /\ r1_tarski X1 (k4_classes1 X2))) H13 __4 (fun H18 =>
TImp (r2_hidden __4 __1) (exists X1:set, v3_ordinal1 X1 /\ r2_hidden X1 __0 /\ r1_tarski __4 (k4_classes1 X1)) H18 (fun H19 => H
19 H16) (fun H19 => TEx set (fun (X1:set) => v3_ordinal1 X1 /\ r2_hidden X1 __0 /\ r1_tarski __4 (k4_classes1 X1)) H19 (fun __5
H20 => TAnd (v3_ordinal1 __5) (r2_hidden __5 __0 /\ r1_tarski __4 (k4_classes1 __5)) H20 (fun H21 H22 => TAnd (r2_hidden __5 __0
) (r1_tarski __4 (k4_classes1 __5)) H22 (fun H23 H24 => TAll set (fun (X1:set) => v3_ordinal1 X1 -> r2_hidden X1 __0 -> q X1) H4
 __5 (fun H25 => TImp (v3_ordinal1 __5) (r2_hidden __5 __0 -> q __5) H25 (fun H26 => H26 H21) (fun H26 => TImp (r2_hidden __5 __
0) (q __5) H26 (fun H27 => H27 H23) (fun H27 => TAll set (fun (X1:set) => r1_tarski X1 (k4_classes1 __5) -> p X1) H27 __4 (fun H
28 => TImp (r1_tarski __4 (k4_classes1 __5)) (p __4) H28 (fun H29 => H29 H24) (fun H29 => H17 H29))))))))))))) (fun H14 => H8 H1
4))))))))))))).
```

If Mizar Accepted Proof Terms...

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
  A2: for A st for B st B in A holds Q[B] holds Q[A]
         byproofterm
```

```
exact (NNPP (forall (X1:set), v3_ordinal1 X1 -> (forall (X2:set), v3_ordinal1 X2 -> r2_hidden X2 X1 -> (forall (X3:set), r1_tars
ki X3 (k4_classes1 X2) -> p X3)) -> (forall (X2:set), r1_tarski X2 (k4_classes1 X1) -> p X2)) (fun H0 => TNAll set (fun (X1:set)
 => v3_ordinal1 X1 -> (forall (X2:set), v3_ordinal1 X2 -> r2_hidden X2 X1 -> q X2) -> q X1) H0 (fun __0 H1 => TNImp (v3_ordinal1
 __0) ((forall (X1:set), v3_ordinal1 X1 -> r2_hidden X1 __0 -> q X1) -> q __0) H1 (fun H2 H3 => TNImp (forall (X1:set), v3_ordin
all X1 -> r2_hidden X1 __0 -> q X1) (q __0) H3 (fun H4 H5 => TNAll set (fun (X1:set) => r1_tarski X1 (k4_classes1 __0) -> p X1)
H5 (fun __1 H6 => TNImp (r1_tarski __1 (k4_classes1 __0)) (p __1) H6 (fun H7 H8 => TAll set (fun (X1:set) => forall (X2:set), v3
_ordinal1 X2 -> r1_tarski X1 (k4_classes1 X2) -> (forall (X3:set), r2_hidden X3 X1 -> (exists X4:set, v3_ordinal1 X4 /\ r2_hidde
n X4 X2 /\ r1_tarski X3 (k4_classes1 X4)))) th1 __1 (fun H9 => TAll set (fun (X1:set) => v3_ordinal1 X1 -> r1_tarski __1 (k4_cla
sses1 X1) -> (forall (X2:set), r2_hidden X2 __1 -> (exists X3:set, v3_ordinal1 X3 /\ r2_hidden X3 X1 /\ r1_tarski X2 (k4_classes
1 X3)))) H9 __0 (fun H10 => TAll set (fun (X1:set) => (forall (X2:set), r2_hidden X2 X1 -> p X2) -> p X1) a1 __1 (fun H11 => TIm
p (v3_ordinal1 __0) (r1_tarski __1 (k4_classes1 __0) -> (forall (X1:set), r2_hidden X1 __1 -> (exists X2:set, v3_ordinal1 X2 /\
r2_hidden X2 __0 /\ r1_tarski X1 (k4_classes1 X2)))) H10 (fun H12 => H12 H2) (fun H12 => TImp (r1_tarski __1 (k4_classes1 __0))
(forall (X1:set), r2_hidden X1 __1 -> (exists X2:set, v3_ordinal1 X2 /\ r2_hidden X2 __0 /\ r1_tarski X1 (k4_classes1 X2))) H12
(fun H13 => H13 H7) (fun H13 => TImp (forall (X1:set), r2_hidden X1 __1 -> p X1) (p __1) H11 (fun H14 => TNAll set (fun (X1:set)
 => r2_hidden X1 __1 -> p X1) H14 (fun __4 H15 => TNImp (r2_hidden __4 __1) (p __4) H15 (fun H16 H17 => TAll set (fun (X1:set) =
> r2_hidden X1 __1 -> (exists X2:set, v3_ordinal1 X2 /\ r2_hidden X2 __0 /\ r1_tarski X1 (k4_classes1 X2))) H13 __4 (fun H18 =>
TImp (r2_hidden __4 __1) (exists X1:set, v3_ordinal1 X1 /\ r2_hidden X1 __0 /\ r1_tarski __4 (k4_classes1 X1)) H18 (fun H19 => H
19 H16) (fun H19 => TEx set (fun (X1:set) => v3_ordinal1 X1 /\ r2_hidden X1 __0 /\ r1_tarski __4 (k4_classes1 X1)) H19 (fun __5
H20 => TAnd (v3_ordinal1 __5) (r2_hidden __5 __0 /\ r1_tarski __4 (k4_classes1 __5)) H20 (fun H21 H22 => TAnd (r2_hidden __5 __0
) (r1_tarski __4 (k4_classes1 __5)) H22 (fun H23 H24 => TAll set (fun (X1:set) => v3_ordinal1 X1 -> r2_hidden X1 __0 -> q X1) H4
 __5 (fun H25 => TImp (v3_ordinal1 __5) (r2_hidden __5 __0 -> q __5) H25 (fun H26 => H26 H21) (fun H26 => TImp (r2_hidden __5 __
0) (q __5) H26 (fun H27 => H27 H23) (fun H27 => TAll set (fun (X1:set) => r1_tarski X1 (k4_classes1 __5) -> p X1) H27 __4 (fun H
28 => TImp (r1_tarski __4 (k4_classes1 __5)) (p __4) H28 (fun H29 => H29 H24) (fun H29 => H17 H29)))))))))))) (fun H14 => H8 H1
4))))))))))))).
```

```
        ;
  A3: for A holds Q[A] from ORDINAL1:sch 2(A2);
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
  thus P[Y] by A3,A4;
end;
```

If Mizar Accepted Proof Terms...

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
  A2: for A st for B st B in A holds Q[B] holds Q[A]
          byproofterm
```

```
exact (NNPP (forall (X1:set), v3_ordinal1 X1 -> (forall (X2:set), v3_ordinal1 X2 -> r2_hidden X2 X1 -> (forall (X3:set), r1_tars
ki X3 (k4_classes1 X2) -> p X3)) -> (forall (X2:set), r1_tarski X2 (k4_classes1 X1) -> p X2)) (fun H0 => TNAll set (fun (X1:set)
 => v3_ordinal1 X1 -> (forall (X2:set), v3_ordinal1 X2 -> r2_hidden X2 X1 -> q X2) -> q X1) H0 (fun __0 H1 => TNImp (v3_ordinal1
 __0) ((forall (X1:set), v3_ordinal1 X1 -> r2_hidden X1 __0 -> q X1) -> q __0) H1 (fun H2 H3 => TNImp (forall (X1:set), v3_ordin
all X1 -> r2_hidden X1 __0 -> q X1) (q __0) H3 (fun H4 H5 => TNAll set (fun (X1:set) => r1_tarski X1 (k4_classes1 __0) -> p X1)
H5 (fun __1 H6 => TNImp (r1_tarski __1 (k4_classes1 __0)) (p __1) H6 (fun H7 H8 => TAll set (fun (X1:set) => forall (X2:set), v3
_ordinal1 X2 -> r1_tarski X1 (k4_classes1 X2) -> (forall (X3:set), r2_hidden X3 X1 -> (exists X4:set, v3_ordinal1 X4 /\ r2_hidde
n X4 X2 /\ r1_tarski X3 (k4_classes1 X4)))) th1 __1 (fun H9 => TAll set (fun (X1:set) => v3_ordinal1 X1 -> r1_tarski __1 (k4_cla
sses1 X1) -> (forall (X2:set), r2_hidden X2 __1 -> (exists X3:set, v3_ordinal1 X3 /\ r2_hidden X3 X1 /\ r1_tarski X2 (k4_classes
1 X3)))) H9 __0 (fun H10 => TAll set (fun (X1:set) => (forall (X2:set), r2_hidden X2 X1 -> p X2) -> p X1) a1 __1 (fun H11 => TIm
p (v3_ordinal1 __0) (r1_tarski __1 (k4_classes1 __0) -> (forall (X1:set), r2_hidden X1 __1 -> (exists X2:set, v3_ordinal1 X2 /\
r2_hidden X2 __0 /\ r1_tarski X1 (k4_classes1 X2)))) H10 (fun H12 => H12 H2) (fun H12 => TImp (r1_tarski __1 (k4_classes1 __0))
(forall (X1:set), r2_hidden X1 __1 -> (exists X2:set, v3_ordinal1 X2 /\ r2_hidden X2 __0 /\ r1_tarski X1 (k4_classes1 X2))) H12
(fun H13 => H13 H7) (fun H13 => TImp (forall (X1:set), r2_hidden X1 __1 -> p X1) (p __1) H11 (fun H14 => TNAll set (fun (X1:set)
 => r2_hidden X1 __1 -> p X1) H14 (fun __4 H15 => TNImp (r2_hidden __4 __1) (p __4) H15 (fun H16 H17 => TAll set (fun (X1:set) =
> r2_hidden X1 __1 -> (exists X2:set, v3_ordinal1 X2 /\ r2_hidden X2 __0 /\ r1_tarski X1 (k4_classes1 X2))) H13 __4 (fun H18 =>
TImp (r2_hidden __4 __1) (exists X1:set, v3_ordinal1 X1 /\ r2_hidden X1 __0 /\ r1_tarski __4 (k4_classes1 X1)) H18 (fun H19 => H
19 H16) (fun H19 => TEx set (fun (X1:set) => v3_ordinal1 X1 /\ r2_hidden X1 __0 /\ r1_tarski __4 (k4_classes1 X1)) H19 (fun __5
H20 => TAnd (v3_ordinal1 __5) (r2_hidden __5 __0 /\ r1_tarski __4 (k4_classes1 __5)) H20 (fun H21 H22 => TAnd (r2_hidden __5 __0
) (r1_tarski __4 (k4_classes1 __5)) H22 (fun H23 H24 => TAll set (fun (X1:set) => v3_ordinal1 X1 -> r2_hidden X1 __0 -> q X1) H4
 __5 (fun H25 => TImp (v3_ordinal1 __5) (r2_hidden __5 __0 -> q __5) H25 (fun H26 => H26 H21) (fun H26 => TImp (r2_hidden __5 __
0) (q __5) H26 (fun H27 => H27 H23) (fun H27 => TAll set (fun (X1:set) => r1_tarski X1 (k4_classes1 __5) -> p X1) H27 __4 (fun H
28 => TImp (r1_tarski __4 (k4_classes1 __5)) (p __4) H28 (fun H29 => H29 H24) (fun H29 => H17 H29)))))))))))) (fun H14 => H8 H1
4))))))))))))).
```

```
          ;
 A3: for A holds Q[A] from ORDINAL1:sch 2(A2);
 let Y;
 consider A such that A4: Y c= Rank A by CLASSES1:62;
 thus P[Y] by A3,A4;
end;
```

- Problem: Not Robust.

Hand-Written Mizar Subproof

```
A2: for A st for B st B in A holds Q[B] holds Q[A]
proof
  let A be Ordinal;
  assume B1: for B st B in A holds Q[B];
  let Y;
  assume B2: Y c= Rank A;
  for x st x in Y holds P[x]
  proof
    let x;
    assume x in Y;
    then consider B such that B3: B in A & x c= Rank B by Th1, B2;
    thus P[x] by B1,B3;
  end;
  hence P[Y] by A1;
end;
```

Hand-Written Mizar Subproof

```
tab_start H0.
tab_negall H0 __0 H1.
tab_negimp H1 H2 H3.
tab_negimp H3 H4 H5.
tab_negall H5 __1 H6.
tab_negimp H6 H7 H8.
tab_all th1 (__1) H9.
tab_all H9 (__0) H10.
tab_all a1 (__1) H11.
tab_imp H10 H12.
 tab_conflict H2 H12.
 tab_imp H12 H13.
  tab_conflict H7 H13.
  tab_imp H11 H14.
   tab_negall H14 __4 H15.
   tab_negimp H15 H16 H17.
   tab_all H13 (__4) H18.
   tab_imp H18 H19.
    tab_conflict H16 H19.
    tab_ex H19 __5 H20.
    tab_and H20 H21 H22.
    tab_and H22 H23 H24.
    tab_all H4 (__5) H25.
    tab_imp H25 H26.
     tab_conflict H21 H26.
     tab_imp H26 H27.
      tab_conflict H23 H27.
      tab_all H27 (__4) H28.
      tab_imp H28 H29.
       tab_conflict H24 H29.
       tab_conflict H29 H17.
   tab_conflict H14 H8.
```

```
A2: for A st for B st B in A holds Q[B] holds Q[A]
proof
  let A be Ordinal;
  assume B1: for B st B in A holds Q[B];
  let Y;
  assume B2: Y c= Rank A;
  for x st x in Y holds P[x]
  proof
    let x;
    assume x in Y;
    then consider B such that B3: B in A & x c= Rank B by Th1, B2;
    thus P[x] by B1,B3;
  end;
  hence P[Y] by A1;
end;
```

Hand-Written Mizar Subproof

```
tab_start negA2.
tab_negall negA2 A H1.
tab_negimp H1 Aord H3.
tab_negimp H3 B1 H5.
tab_negall H5 Y H6.
tab_negimp H6 B2 negPY.
tab_all Th1 (Y) H9.
tab_all H9 (A) H10.
tab_all A1 (Y) H11.
tab_imp H10 H12.
 tab_conflict Aord H12.
 tab_imp H12 H13.
  tab_conflict B2 H13.
  tab_imp H11 H14.
   tab_negall H14 x H15.
   tab_negimp H15 x_in_Y negPx.
   tab_all H13 (x) H18.
   tab_imp H18 H19.
    tab_conflict x_in_Y H19.
    tab_ex H19 B H20.
    tab_and H20 Bord B3.
    tab_and B3 H23 H24.
    tab_all B1 (B) H25.
    tab_imp H25 H26.
     tab_conflict Bord H26.
     tab_imp H26 H27.
      tab_conflict H23 H27.
      tab_all H27 (x) H28.
      tab_imp H28 H29.
       tab_conflict H24 H29.
       tab_conflict H29 negPx.
   tab_conflict H14 negPY.
```

```
A2: for A st for B st B in A holds Q[B] holds Q[A]
proof
   let A be Ordinal;
   assume B1: for B st B in A holds Q[B];
   let Y;
   assume B2: Y c= Rank A;
   for x st x in Y holds P[x]
   proof
      let x;
      assume x in Y;
      then consider B such that B3: B in A & x c= Rank B by Th1, B2;
      thus P[x] by B1,B3;
   end;
   hence P[Y] by A1;
end;
```

Proof of Epsilon Induction

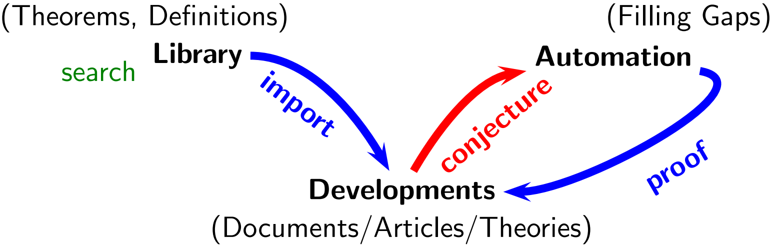
```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
  A2: for A st for B st B in A holds Q[B] holds Q[A]
  proof
    let A be Ordinal;
    assume B1: for B st B in A holds Q[B];
    let Y;
    assume B2: Y c= Rank A;
    for x st x in Y holds P[x]
    proof
      let x;
      assume x in Y;
      then consider B such that B3: B in A & x c= Rank B by Th1, B2;
      thus P[x] by B1,B3;
    end;
    hence P[Y] by A1;
  end;
  A3: for A holds Q[A] from ORDINAL1:sch 2(A2);
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
  thus P[Y] by A3,A4;
end;
```

Proof of Epsilon Induction

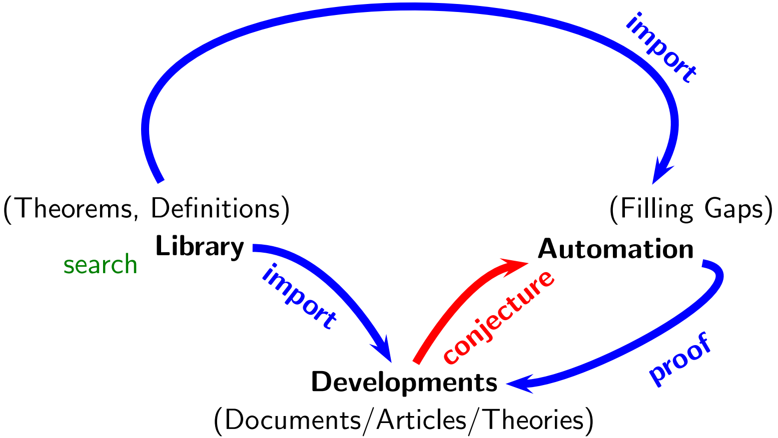
```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
proof
  defpred Q[Ordinal] means for Y st Y c= Rank $1 holds P[Y];
  A2: for A st for B st B in A holds Q[B] holds Q[A]
  proof
    let A be Ordinal;
    assume B1: for B st B in A holds Q[B];
    let Y;
    assume B2: Y c= Rank A;
    for x st x in Y holds P[x]
    proof
      let x;
      assume x in Y;
      then consider B such that B3: B in A & x c= Rank B by Th1, B2;
      thus P[x] by B1,B3;
    end;
    hence P[Y] by A1;
  end;
  A3: for A holds Q[A] from ORDINAL1:sch 2(A2);
  let Y;
  consider A such that A4: Y c= Rank A by CLASSES1:62;
  thus P[Y] by A3,A4;
end;
```

Actually, MPTP could prove this too, finding the reference.

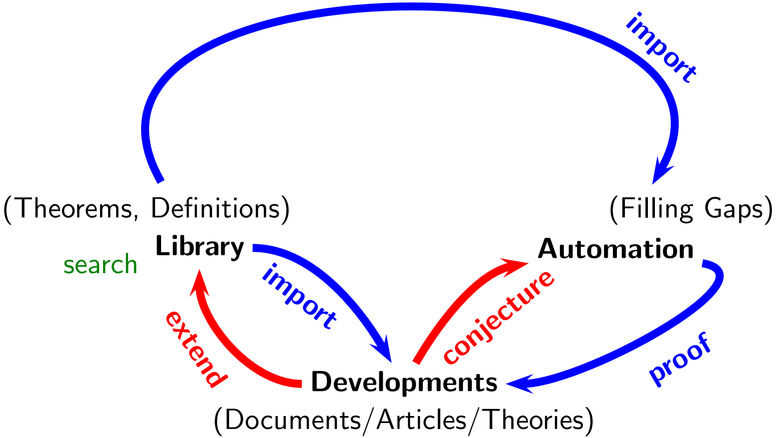
Three Legs Supporting the Formalization Process



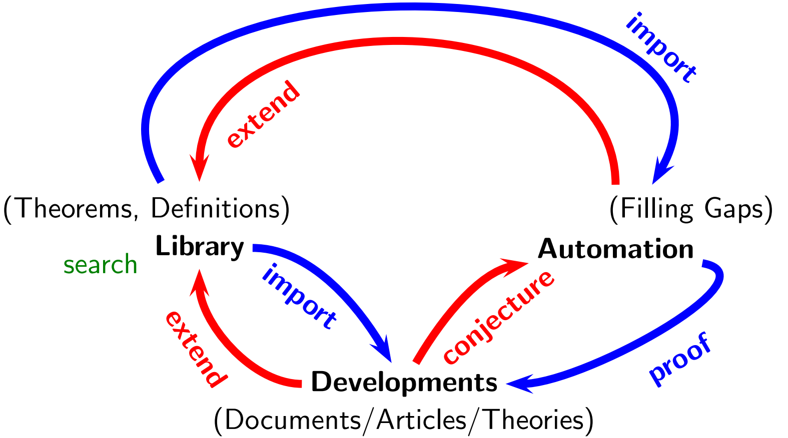
Three Legs Supporting the Formalization Process



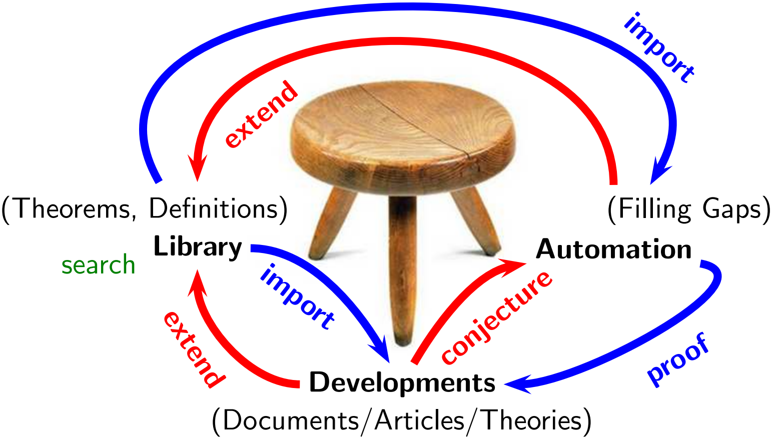
Three Legs Supporting the Formalization Process



Three Legs Supporting the Formalization Process



Three Legs Supporting the Formalization Process



Outline

Ideal Mathematical Library

- Given a closed term t, library tells you a constant d defined by $d = t$.

Ideal Mathematical Library

- Given a closed term t, library tells you a constant d defined by $d = t$.
- For each closed proposition P, library tells you if P is provable.

Ideal Mathematical Library

- Given a closed term t, library tells you a constant d defined by $d = t$.
- For each closed proposition P, library tells you if P is provable.
- In practice, there is a finite approximation of this ideal library, which becomes better after each development.

Ideal Mathematical Library

- Given a closed term t, library tells you a constant d defined by $d = t$.
- For each closed proposition P, library tells you if P is provable.
- In practice, there is a finite approximation of this ideal library, which becomes better after each development.
- After the incorporating the previous development:
  - $\in$-induction would be known to be provable.
  - U and I each would have a unique entry (of type $\iota \to \iota$) as a definition.
  - $\forall Y.U(I(Y)) = Y$ would be known to be provable.

Ideal Mathematical Library

- Given a closed term t, library tells you a constant d defined by $d = t$.
- For each closed proposition P, library tells you if P is provable.
- In practice, there is a finite approximation of this ideal library, which becomes better after each development.
- After the incorporating the previous development:
  - $\in$-induction would be known to be provable.
  - U and I each would have a unique entry (of type $\iota \to \iota$) as a definition.
  - $\forall Y.U(I(Y)) = Y$ would be known to be provable.
- Queries for exact matches (of definitions or theorems) can be immediate (hashing).

Ideal Mathematical Library

- Given a closed term t, library tells you a constant d defined by $d = t$.
- For each closed proposition P, library tells you if P is provable.
- In practice, there is a finite approximation of this ideal library, which becomes better after each development.
- After the incorporating the previous development:
  - $\in$-induction would be known to be provable.
  - U and I each would have a unique entry (of type $\iota \to \iota$) as a definition.
  - $\forall Y.U(I(Y)) = Y$ would be known to be provable.
- Queries for exact matches (of definitions or theorems) can be immediate (hashing).
- Bad for searching, but fine for importing.

Library via Hashing: Knowns

- $\in$-induction:

$$\forall P.(\forall Y.(\forall x \in Y\,.\,P[x]) \to P[Y]) \to \forall Y.P[Y]$$

Convert to nameless representation, serialize and hash.

Library via Hashing: Knowns

- $\in$-induction:

$$\forall P.(\forall Y.(\forall x \in Y \,.\, P[x]) \to P[Y]) \to \forall Y.P[Y]$$

  Convert to nameless representation, serialize and hash.
- Remember the hash is "known" so that every proposition hashing to it can be used in future developments without proof.

Library via Hashing: Knowns

- $\in$-induction:

$$\forall P.(\forall Y.(\forall x \in Y\,.\,P[x]) \to P[Y]) \to \forall Y.P[Y]$$

  Convert to nameless representation, serialize and hash.
- Remember the hash is "known" so that every proposition hashing to it can be used in future developments without proof.

```
scheme
  EpsilonInduction { P[set] } : for Y holds P[Y]
provided
A1: for Y st for x st x in Y holds P[x] holds P[Y]
by known;
```

Library via Hashing: Knowns

- $\in$-induction:

$$\forall P.(\forall Y.(\forall x \in Y \,.\, P[x]) \to P[Y]) \to \forall Y.P[Y]$$

  Convert to nameless representation, serialize and hash.

- Remember the hash is "known" so that every proposition hashing to it can be used in future developments without proof.

```
scheme
  OtherEpsInduction { Q[set] } : for Z holds Q[Z]
provided
for Z st for y st y in Z holds Q[y] holds Q[Z]
by known;
```

Library via Hashing: Knowns

- $\in$-induction:

$$\forall P.(\forall Y.(\forall x \in Y \,.\, P[x]) \to P[Y]) \to \forall Y.P[Y]$$

  Convert to nameless representation, serialize and hash.

- Remember the hash is "known" so that every proposition hashing to it can be used in future developments without proof.

```
scheme
  OtherEpsInduction { Q[set] } : for Z holds Q[Z]
provided
for Z st for y st y in Z holds Q[y] holds Q[Z]
by known;
```

- Don't even need to know the article where it was proven.

Library via Hashing: Definitions

- $I : \iota \to \iota$ defined by a term d. Serialize nameless rep and hash to obtain $\sharp d$.

Library via Hashing: Definitions

- $l : \iota \to \iota$ defined by a term d. Serialize nameless rep and hash to obtain $\sharp d$.
- Remember $\sharp d$ is the hash of a term of type $\iota \to \iota$.

Library via Hashing: Definitions

- $I : \iota \to \iota$ defined by a term d. Serialize nameless rep and hash to obtain $\sharp d$.
- Remember $\sharp d$ is the hash of a term of type $\iota \to \iota$.
- Allow constants of type $\iota \to \iota$ to be declared to correspond to $\sharp d$.

```
definition
 let Y be set;
 func I(Y) -> set
:: #d
end;
```

Library via Hashing: Definitions

- $I : \iota \to \iota$ defined by a term d. Serialize nameless rep and hash to obtain $\sharp d$.
- Remember $\sharp d$ is the hash of a term of type $\iota \to \iota$.
- Allow constants of type $\iota \to \iota$ to be declared to correspond to $\sharp d$.

```
definition
 let Y be set;
 func I(Y) -> set
:: #d
end;
```

- $U : \iota \to \iota$ defined by a term e – obtain $\sharp e$.

```
definition
 let Y be set;
 func U(Y) -> set
:: #e
end;
```

Library via Hashing: Definitions

- $I : \iota \to \iota$ defined by a term d. Serialize nameless rep and hash to obtain $\sharp d$.
- Remember $\sharp d$ is the hash of a term of type $\iota \to \iota$.
- Allow constants of type $\iota \to \iota$ to be declared to correspond to $\sharp d$.

```
definition
 let Y be set;
 func I(Y) -> set
:: #d
end;
```

- $U : \iota \to \iota$ defined by a term e – obtain $\sharp e$.

```
definition
 let Y be set;
 func U(Y) -> set
:: #e
end;
```

- Opaque importation of definitions

Library via Hashing

- Represent the theorem $\forall Y. U(I(Y)) = Y$ as $\forall Y. \sharp e(\sharp d(Y))$.
- Serialize nameless rep of $\forall Y. \sharp e(\sharp d(Y))$ and hash.
- Remember hash is "known." In new developments this should be accepted:

```
theorem UIThm: for Y holds U(I(Y)) = Y by known;
```

Abstract Library

An *abstract library* is $(\mathcal{D}, \mathcal{K})$ where

- $\mathcal{D}$ is a partial function from hashes to types. (definitions)
- $\mathcal{K}$ is a set of hashes. (knowns)

Abstract Library

An *abstract library* is $(\mathcal{D}, \mathcal{K})$ where

- $\mathcal{D}$ is a partial function from hashes to types. (definitions)
- $\mathcal{K}$ is a set of hashes. (knowns)

Importing a development into the library increases $\mathcal{D}$ and/or $\mathcal{K}$.

Outline

Automation

- ATP: Usually First-Order (Vampire, E, . . .)
- Sometimes Higher-Order (TPS, LEO, Satallax)

Automation

- ATP: Usually First-Order (Vampire, E, ...)
- Sometimes Higher-Order (TPS, LEO, Satallax)
- Mizar is approximately FO, but not in the ATP sense.
  - MPTP can't send Vampire a scheme.
  - MPTP can send Satallax a scheme.

Automation

- ATP: Usually First-Order (Vampire, E, ...)
- Sometimes Higher-Order (TPS, LEO, Satallax)
- Mizar is approximately FO, but not in the ATP sense.
  - MPTP can't send Vampire a scheme.
  - MPTP can send Satallax a scheme.
- Other Interactive Provers aren't FO at all.
  Instead of using general, complete procedures – try a variety of incomplete ones.
  - HOL-light, MESON, techniques for instantiating type vars
  - Isabelle-HOL, Sledgehammer
  - Coq, tactics

Automation

- ATP: Usually First-Order (Vampire, E, . . .)
- Sometimes Higher-Order (TPS, LEO, Satallax)
- Mizar is approximately FO, but not in the ATP sense.
  - MPTP can't send Vampire a scheme.
  - MPTP can send Satallax a scheme.
- Other Interactive Provers aren't FO at all.
  Instead of using general, complete procedures – try a variety of incomplete ones.
  - HOL-light, MESON, techniques for instantiating type vars
  - Isabelle-HOL, Sledgehammer
  - Coq, tactics
- Problems for having complete ATPs:
  - Polymorphism (type variables)
  - Type Definitions using Predicates

Frameworks for Leveraging Automation

- Unrealistic Proposal: Use NBG (Quaife, 1992)

Frameworks for Leveraging Automation

- Unrealistic Proposal: Use NBG (Quaife, 1992)
- More Realistic Proposal:
  Simple Type Theory+Set Theory

Frameworks for Leveraging Automation

- Unrealistic Proposal: Use NBG (Quaife, 1992)
- More Realistic Proposal:
  Simple Type Theory+Set Theory
- Finite axiomatization of HO Tarski-Grothendieck (HOTG)

Frameworks for Leveraging Automation

- Unrealistic Proposal: Use NBG (Quaife, 1992)
- More Realistic Proposal:
  Simple Type Theory+Set Theory
- Finite axiomatization of HO Tarski-Grothendieck (HOTG)
- Support for term level binders like $\{x \in A | P[x]\}$

Frameworks for Leveraging Automation

- Unrealistic Proposal: Use NBG (Quaife, 1992)
- More Realistic Proposal:
  Simple Type Theory+Set Theory
- Finite axiomatization of HO Tarski-Grothendieck (HOTG)
- Support for term level binders like $\{x \in A | P[x]\}$
- The MML (and other libraries) could be translated into HOTG
- No need for polymorphism or type definitions

Frameworks for Leveraging Automation

- Unrealistic Proposal: Use NBG (Quaife, 1992)
- More Realistic Proposal:
  Simple Type Theory+Set Theory
- Finite axiomatization of HO Tarski-Grothendieck (HOTG)
- Support for term level binders like $\{x \in A | P[x]\}$
- The MML (and other libraries) could be translated into HOTG
- No need for polymorphism or type definitions
- (Henkin) Complete ATPs for STT Exist

SATALLAX

Outline

Conclusion

- Write Formal Developments (Local)
- Contribute to a Growing Global Library

Conclusion

- Write Formal Developments (Local)
- Contribute to a Growing Global Library
- Library as a Collection of Hashes

Conclusion

- Write Formal Developments (Local)
- Contribute to a Growing Global Library
- Library as a Collection of Hashes
- Automation to Fill Gaps

Conclusion

- Write Formal Developments (Local)
- Contribute to a Growing Global Library
- Library as a Collection of Hashes
- Automation to Fill Gaps
- Automation: More than FO *...but not too much more*