FAbstracts:

How can we get the best of all systems?

Cezary Kaliszyk Karol Pąk

AITP 2018

FAbstracts project

How do we understand it?

- Provide statements and definitions
- For all known results in Math++ (and major assumptions)
- Computer understandable

We also want: language close to standard mathematics

- Extensible notation language
- High level of disambiguation

We also want: expressible foundations

- No need to “deep embed”
- Definitions should not need too much reformulation

We also want: consistent library

FAbstracts inspired by many proof assistants

HOL and Isabelle/HOL

- Simplicity of the foundations
- Ease of use
- Notations (ML Parse translations)

Type Theory-based systems

- Powerful foundations allowing reasoning in many domains
- Reflection

Set theory based systems?

- Not so much?

FAbstracts inspired by many proof assistants

HOL and Isabelle/HOL

- Simplicity of the foundations
- Ease of use
- Notations (ML Parse translations)

Type Theory-based systems

- Powerful foundations allowing reasoning in many domains
- Reflection

Set theory based systems?

- Foundations maybe more familiar for mathematicians
- Soft types
- Structures with (multi-)inheritance

Could FAbstracts take inspiration from Mizar?

Proof Assistant

- Many features quite different from the usual
- Developed by mathematicians for mathematicians
- Initially as a type-setting system

Could FAbstracts take inspiration from Mizar?

Proof Assistant

- Many features quite different from the usual
- Developed by mathematicians for mathematicians
- Initially as a type-setting system

Math type-setting system (1971)

- Extended to check proofs (in 1973)
- Consistent library of formalized Math (1980s)

Natural deduction

- Stays as long as possible in first-order logic

Foundations

- Set Theory (with universes, rarely used)
- Dependent soft type system and type inference mechanism

Other Mizar features

Rich input language and $\LaTeX$ generation

- Contextual parsing: more than 100 meanings of "+"
- Journal of Formalized Mathematics

Focus on mathematics

- A lot not covered elsewhere (lattices)
- Much less computer related proofs (random access Turing machines)

The system has evolved

- unfortunately many features have not changed since the 1980s...

Can we express it all in a modern logical framework?

1. SQUARE ROOTS OF PRIMES ARE IRRATIONAL

For simplicity, we adopt the following convention: k, n, p, K, N are natural numbers, x, y, e_1 are real numbers, s_1, s_2, s_3 are sequences of real numbers, and s_4 is a finite sequence of elements of $\mathbb{R}$.

Let us consider x. We introduce x is irrational as an antonym of x is rational.

Let us consider x, y. We introduce x^y as a synonym of x^y.

One can prove the following propositions:

(1) If p is prime, then $\sqrt{p}$ is irrational.

(2) There exist x, y such that x is irrational and y is irrational and x^y is rational.

1. Square Roots of Primes are Irrational

For simplicity, we adopt the following convention: k, n, p, K, N are natural numbers, x, y, e_1 are real numbers, s_1, s_2, s_3 are sequences of real numbers, and s_4 is a finite sequence of elements of $\mathbb{R}$.

Let us consider x. We introduce x is irrational as an antonym of x is rational.

Let us consider x, y. We introduce x^y as a synonym of x^y.

One can prove the following propositions:

(1) If p is prime, then $\sqrt{p}$ is irrational.

(2) There exist x, y such that x is irrational and y is irrational and x^y is rational.

```
:: The Irrationality of the Square Root of 2
theorem Th1:
 for p being Element of NAT st p is prime holds
 sqrt p is irrational
proof
 let p be Element of NAT ;
 assume A1: p is prime ;
 then A2: p > 1 by INT_2:def 4;
 assume sqrt p is rational ;
 then consider i being Integer, n being Element of NAT such that
 A3: n <> 0 and
 A4: sqrt p = i / n and
 A5: for i1 being Integer
```

Encoding the Mizar foundations in Isabelle

We can use Isabelle/FOL

- Features beyond first-order can be encoded in the logical framework

Define the meta-types

typedecl *Set*
typedecl *Ty*
$$inhabited(D) \longleftrightarrow (\exists_M x.\ x\ is\ D)$$

Example Mizar types

- even natural number
- bijective Function of A,B

Meta-Level Constants

Construct Types

Construct Meta-level functions

User-level typing rules

consts

$$
\begin{array}{ll}
\quad \mathit{ty\text{-}membership} :: \mathit{Set} \Rightarrow \mathit{Ty} \Rightarrow o & (\textbf{infix}\ \mathit{be}\ 90) \\
\quad \mathit{define\text{-}ty} :: \mathit{Ty} \Rightarrow (\mathit{Set} \Rightarrow o) \Rightarrow (\mathit{Set} \Rightarrow o) \Rightarrow \mathit{Ty} & \\
\quad \mathit{choice} :: \mathit{Ty} \Rightarrow \mathit{Set} & (\mathit{the}\ \text{-})
\end{array}
$$

Axiomatization

Axiom of choice

What does it mean to define a type?

$$\text{it be parent} \wedge (\text{cond(it)} \longrightarrow \text{property(it)})$$

"non-" (e.g. non-negative)

$$\textit{non } A \equiv \textit{define-ty}(\textit{object}, \lambda\text{-}.\ \textit{True}, \lambda x.\ \neg\ x \textit{ is } A)$$

$$x \textit{ is non } A \longleftrightarrow \neg\ x \textit{ is } A$$

Intersection types

$$x \textit{ is t1} \mid \textit{t2} \longleftrightarrow x \textit{ is t1} \wedge x \textit{ is t2}$$

Mizar Notations

Types and adjectives

term x *is set*

term x *is empty* | *set*

term x *is non empty* | *Subset–of NAT*

term *onto*(*NAT*)

term x *is empty* | *non onto*(*NAT*) | *Function*

term *the empty* | *set*

Formulas

term $P \wedge Q \longleftrightarrow W \longrightarrow (\mathit{not}\ P \longrightarrow R \vee W)$

term *for x,y being set,z being object holds P(x,y,z)*

term *ex y being even* | *Element–of NAT st Q(y)*

reserve a **for** *Function*

reserve b **for** *non empty* | *set*

term *for a,b holds Q(a,b)*

Part of Mizar, that is expressible but not standard set theory

- type of sets
- (set) equality
- set membership

axiomatization
  object **and**
  prefix-in :: *Set* ⇒ *Set* ⇒ *o* (**infixl** *in 50*)
where
  object-root: *x be object* **and**
  object-exists: *inhabited*(*object*)

Tarski-Grothendieck Set Theory

```
reserve x,y,z,u,a for object;
reserve M,N,X,Y,Z for set;
:: Set axiom
theorem :: TARSKI_0:1
  for x holds x is set;

:: Extensionality axiom
theorem :: TARSKI_0:2
  (for x holds x in X iff x in Y)
    implies X = Y;

:: Axiom of pair
theorem :: TARSKI_0:3
  for x,y ex Z st for a holds
    a in Z iff a = x or a = y;

:: Axiom of union
theorem :: TARSKI_0:4
  for X ex Z st for x holds
    x in Z iff ex Y st x in Y & Y in X;

:: Axiom of regularity
theorem :: TARSKI_0:5
  x in X implies ex Y st Y in X &
    not ex x st x in X & x in Y;
```

reserve *x,y,z,u,a* **for** *object*
reserve *M,N,X,Y,Z* **for** *set*

— Set axiom
theorem *tarski-0-1*:
$\forall x.\ x\ \mathit{be\ set}$ **using** *SET-def* **by** *simp*

— Extensionality axiom
axiomatization where *tarski-0-2*:
$\forall X.\ \forall Y.\ (\forall x.\ x\ \mathit{in}\ X \longleftrightarrow x\ \mathit{in}\ Y)$
$\longrightarrow X = Y$

— Axiom of pair
axiomatization where *tarski-0-3*:
$\forall x.\ \forall y.\ \exists Z.\ \forall a.$
$a\ \mathit{in}\ Z \longleftrightarrow a = x \vee a = y$

— Axiom of union
axiomatization where *tarski-0-4*:
$\forall X.\ \exists Z.\ \forall x.$
$x\ \mathit{in}\ Z \longleftrightarrow (\exists Y.\ x\ \mathit{in}\ Y \wedge Y\ \mathit{in}\ X)$

— Axiom of regularity
axiomatization where *tarski-0-5*:
$\forall x.\ \forall X.\ x\ \mathit{in}\ X \longrightarrow (\exists Y.\ Y\ \mathit{in}\ X \wedge$
$\neg(\exists z.\ z\ \mathit{in}\ X \wedge z\ \mathit{in}\ Y))$

Tarski-Grothendieck Set Theory

```
reserve x,y,z,u,a for object;
reserve M,N,X,Y,Z for set;
:: Set axiom
theorem :: TARSKI_0:1
  for x holds x is set;

:: Extensionality axiom
theorem :: TARSKI_0:2
  (for x holds x in X iff x in Y)
     implies X = Y;

:: Axiom of pair
theorem :: TARSKI_0:3
  for x,y ex Z st for a holds
    a in Z iff a = x or a = y;

:: Axiom of union
theorem :: TARSKI_0:4
  for X ex Z st for x holds
    x in Z iff ex Y st x in Y & Y in X;

:: Axiom of regularity
theorem :: TARSKI_0:5
  x in X implies ex Y st Y in X &
    not ex x st x in X & x in Y;
```

reserve *x,y,z,u,a* **for** *object*
reserve *M,N,X,Y,Z* **for** *set*

— Set axiom
theorem *tarski-0-1*:
$\forall x.\ x$ *be set* **using** *SET-def* **by** *simp*

— Extensionality axiom
axiomatization where *tarski-0-2*:
$\forall X.\ \forall Y.\ (\forall x.\ x \ in\ X \longleftrightarrow x \ in\ Y) \longrightarrow X = Y$

— Axiom of pair
axiomatization where *tarski-0-3*:
$\forall x.\ \forall y.\ \exists Z.\ \forall a.\ a \ in\ Z \longleftrightarrow a = x \vee a = y$

— Axiom of union
axiomatization where *tarski-0-4*:
$\forall X.\ \exists Z.\ \forall x.\ x \ in\ Z \longleftrightarrow (\exists Y.\ x \ in\ Y \wedge Y \ in\ X)$

— Axiom of regularity
axiomatization where *tarski-0-5*:
$\forall x.\ \forall X.\ x \ in\ X \longrightarrow (\exists Y.\ Y \ in\ X \wedge \neg(\exists z.\ z \ in\ X \wedge z \ in\ Y))$

differences: quantification, types, parentheses, schemes

Support for Mizar Definitions

Conditional Definitions

Definitions by “means”

Type definitions

Structures

Simple definition package

- Core definitions
- User obligations
- Derived properties

Definitions

```
  let y be object;
  func { y } → set means
:: TARSKI:def 1
    for x holds x in it iff x = y;
```

mdef *tarski-def-1* ({-}) **where**
mlet *y be object*
func2 $\{y\} \rightarrow$ *set means* $\lambda it.$
$\forall x.\ x\ in\ it \longleftrightarrow x = y$

```
  let y, z be object;
  func { y, z } → set  means
:: TARSKI:def 2
    x in it iff x = y or x = z;
```

mdef *tarski-def-2* ({- , -}) **where**
mlet *y be object*, *z be object*
func2 $\{y, z\} \rightarrow$ *set means* $\lambda it.$
$\forall x.\ x\ in\ it \longleftrightarrow (x = y \vee x = z)$

```
  let X be set;
  func union X → set means
:: TARSKI:def 4
    x in it iff ex Y st x in Y & Y in X;
```

mdef *tarski-def-4* (*union* -) **where**
mlet *X be set*
func union $X \rightarrow$ *set means* $\lambda it.$
$\forall x.\ x\ in\ it \longleftrightarrow (\exists Y.\ x\ in\ Y \wedge Y\ in\ X)$

```
  func {} → set equals
:: XBOOLE_0:def 2
    the empty set;
```

mdef *xboole-0-def-2* ({}) **where**
func $\{\} \rightarrow$ *set equals*
the empty|set

Tuples: Consider the ring structure: $\langle R, +, \mathbf{0}, \cdot, \mathbf{1} \rangle$

```
struct doubleLoopStr (#
  carrier → set,
  addF → BinOp of the carrier,
  ZeroF → Element of the carrier,
  multF → BinOp of the carrier,
  OneF → Element of the carrier
#)
```

Tuples: Consider the ring structure: $\langle R, +, \mathbf{0}, \cdot, \mathbf{1} \rangle$

```
struct doubleLoopStr (#
  carrier → set,
  addF → BinOp of the carrier,
  ZeroF → Element of the carrier,
  multF → BinOp of the carrier,
  OneF → Element of the carrier
#)
```

Modeled as partial functions:

mdefinition *doubleLoopStr-d*(*doubleLoopStr*) **where**
struct doubleLoopStr (#
  carrier → (λ*S*. *set*);
  addF → (λ*S*. *BinOp–of the carrier of S*);
  ZeroF → (λ*S*. *Element–of the carrier of S*);
  multF → (λ*S*. *BinOp–of the carrier of S*);
  OneF → (λ*S*. *Element–of the carrier of S*)
#) : *struct-well-defined...*

Tuples: Consider the ring structure: $\langle R, +, \mathbf{0}, \cdot, \mathbf{1} \rangle$

```
struct doubleLoopStr (#
  carrier → set,
  addF → BinOp of the carrier,
  ZeroF → Element of the carrier,
  multF → BinOp of the carrier,
  OneF → Element of the carrier
#)
```

Modeled as partial functions:

mdefinition *doubleLoopStr-d*(*doubleLoopStr*) **where**
struct doubleLoopStr (#
carrier → (λ*S. set*);
addF → (λ*S. BinOp—of the carrier of S*);
ZeroF → (λ*S. Element—of the carrier of S*);
multF → (λ*S. BinOp—of the carrier of S*);
OneF → (λ*S. Element—of the carrier of S*)
#) : *struct-well-defined...*

abbreviation
Ring ≡ *Abelian* | *add—associative* | *right—zeroed* |
right—complementable | *associative* |
well—unital | *distributive* |
non empty—struct | *doubleLoopStr*

Example: Algebra

```
reserve G for Group;
reserve h, g for Element of G;

theorem Th16:
 (h * g) " = g" * h"
proof
 (g" * h") * (h * g)
  = g" * h" * h * g
   by Def3
 .= g" * (h" * h) * g
   by Def3
 .= g" * 1_G * g
   by Def5
 .= g" * g
   by Def4
 .= 1_G
   by Def5;
 hence thesis
   by Th11;

end;
```

Example: Algebra

```
reserve G for Group;
reserve h, g for Element of G;

theorem  Th16:
 (h * g)" = g" * h"
proof
 (g" * h") * (h * g)
  = g" * h" * h * g
    by Def3
 .= g" * (h" * h) * g
    by Def3
 .= g" * 1_G * g
    by Def5
 .= g" * g
    by Def4
 .= 1_G
    by Def5;
 hence thesis
    by Th11;

end;
```

reserve *G* **for** *Group*
reserve *h,g* **for** *Element–of–struct G*

mtheorem *group-1-th-16*:
$(h \otimes_G g)^{-1}{}_G = g^{-1}{}_G \otimes_G h^{-1}{}_G$
proof–
have $(g^{-1}{}_G \otimes_G h^{-1}{}_G) \otimes_G (h \otimes_G g) = (g^{-1}{}_G \otimes_G h^{-1}{}_G) \otimes_G h \otimes_G g$
using *group-1-def-3E*[*of - - h*] **by** *mauto*
also have ... $= g^{-1}{}_G \otimes_G (h^{-1}{}_G \otimes_G h) \otimes_G g$
using *group-1-def-3E* **by** *mty auto*
also have ... $= g^{-1}{}_G \otimes_G 1._G \otimes_G g$
using *group-1-def-5* **by** *mauto*
also have ... $= (g^{-1}{}_G) \otimes_G g$
using *group-1-def-4* **by** *mauto*
also have ... $= 1._G$
using *group-1-def-5* **by** *mauto*
finally show *?thesis*
using *group-1-th-11*[*of - h* $\otimes_G$ *g, THEN conjunct1*] **by** *mauto*
qed

Type Inference

User can state and prove inference rules

cluster

$$Y\ be\ set \implies F\ be\ Y\text{—}valued \mid Function \implies$$
$$F\ be\ Y\text{—}onto \mid one\text{—}to\text{—}one \implies F\ be\ Y\text{—}bijective$$

cluster

$$f\ be\ Function \wedge g\ be\ Function \implies$$
$$(g \circ f)\ is\ Function\text{-}like$$

Type Inference

User can state and prove inference rules

cluster

$$Y\ be\ set \Longrightarrow F\ be\ Y\text{—}valued \mid Function \Longrightarrow$$
$$F\ be\ Y\text{—}onto \mid one\text{—}to\text{—}one \Longrightarrow F\ be\ Y\text{—}bijective$$

cluster

$$f\ be\ Function \wedge g\ be\ Function \Longrightarrow$$
$$(g \circ f)\ is\ Function\text{-}like$$

Type inference derives all the derivable properties of an object

- In the previous proof, 35 judgements automatically derived, e.g.

$$\mathsf{G\ is\ unital}$$

$$\mathsf{(h} \otimes_G \mathsf{g)}^{-1}{}_G\ \mathsf{is\ Element\text{-}of\ G}$$

$$\mathsf{the\ carrier\ of\ G\ is\ non\ empty}$$

Examples (2/2)

Ordinals

theorem *ordinal-2-sch-19*:
  assumes [*ty*]: *a is Nat*
    and *A1*: $P(\{\})$
    and *A2*: $\forall\, n : Nat.\ P(n) \longrightarrow P(succ\ n)$
  shows $P(a)$

Examples (2/2)

Ordinals

theorem *ordinal-2-sch-19*:
  assumes $[ty]$: a *is* Nat
    and $A1$: $P(\{\})$
    and $A2$: $\forall\, n : Nat.\ P(n) \longrightarrow P(succ\ n)$
  shows $P(a)$

Turing Machines

theorem *extpro-1*:
  assumes $[ty]$: N *be with-zero* $|$ *set*
  shows $halt_{Trivial-AMI\ N}$ *is halting* $Trivial-AMI\ N,\ N$

Semi-Automated Translation

Export combined syntactic-semantic Mizar

Isabelle can import first 100 MML articles

All definitions, theorems, user typing rules

- So far the proofs assumed in the import

Usable environment for proof development

- Type inference

Summary: Mizar features could be useful for FAbstracts?

Familiar mathematical foundations

Convenient proof style

Actual support for Mathematicians

Committee taking care of the library

In a modern logical framework