

the Mizar type system

Freek Wiedijk

Radboud University Nijmegen

TPHOLs 2007

University of Kaiserslautern

2007 09 12, 10:00

this talk

- Mizar
- Mizar types
- the paper
 - the Mizar type system in the form of **typing rules**
 - **correctness** with respect to first order predicate logic



Mizar versus the HOLs

Mizar

HOL

Isabelle

PVS

Coq

⋮

batch checking

interactive

first order logic

higher order logic

readable proofs

tactic scripts

untyped set theory

typed foundations

very nice type system

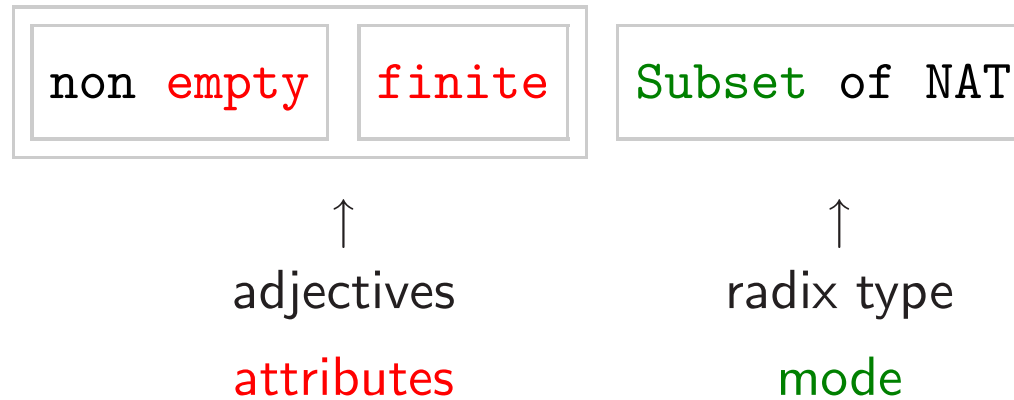
Mizar types

features

- dependent types
- **no** types built from types
no function types $A \rightarrow B$ for types A and B
(`'Ordinal \rightarrow Ordinal'`?)
- subtyping
- 'attributes'
- structure types (records)

Alternative Aggregates in Mizar
Gilbert Lee and Piotr Rudnicki
MKM 2007, LNAI 4573

attributes



overloading

meaning of an attribute depends on the radix type:

- **connected** Relation
- **connected** Graph
- **connected** TopSpace

example

definition let d be `non zero Element of NAT` ;

func cyclotomic_poly d -> `Polynomial of F_Complex` means

`ex s being non empty finite Subset of F_Complex`

`st s = { y where y is Element of MultGroup F_Complex : ord y = d } &`

`it = poly_with_roots((s,1)-bag);`

end;

UNIROOTS **Primitive Roots of Unity and Cyclotomic Polynomials**

Broderick Arneson and Piotr Rudnicki

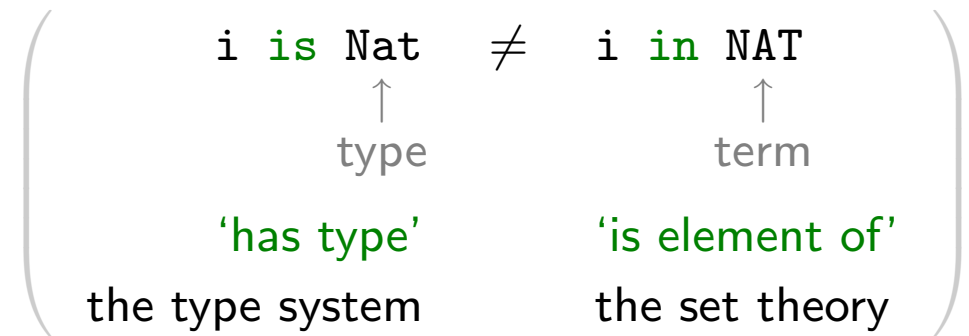
3293 lines, 135K

full Mizar library 985 'articles', 1.97 million lines, 68.9M

typings also are formulas

for `i` being Integer holds `i >= 0` iff `i is Nat`

$$\forall i : \mathbb{Z}. i \geq 0 \Leftrightarrow (i : \mathbb{N})$$



‘coerce’ a term to a more informative type by giving a proof

...

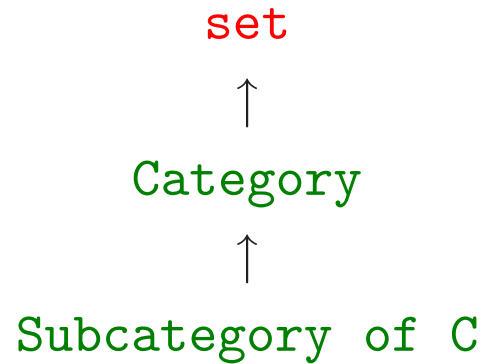
```
then n' - n >= 0 by XREAL_1:50;
```

```
then reconsider d = n' - n as Nat by INT_1:16;
```

...

subtyping

```
definition let C be Category;  
  mode Subcategory of C -> Category means  
  :: CAT_2:def 4  
  ...  
end;
```



supertype may depend on the types of the arguments of the type

clusters

for X being set holds X is **empty** implies X is **finite**



cluster empty -> **finite set**

any term with attribute **empty** **automatically** also gets attribute **finite**

typed or untyped logic?

what does it all mean?

foundations of Mizar:

Tarski-Grothendieck set theory

=

ZFC + 'there are arbitrarily large inaccessible cardinals'

set of axioms on top of **untyped** first order predicate logic

the axioms of Mizar

TARSKI: def 3
$$X \subseteq Y \Leftrightarrow (\forall x. x \in X \Rightarrow x \in Y)$$

TARSKI: def 5
$$\langle x, y \rangle = \{\{x, y\}, \{x\}\}$$

TARSKI: def 6
$$X \sim Y \Leftrightarrow \exists Z. (\forall x. x \in X. \Rightarrow \exists y. y \in Y \wedge \langle x, y \rangle \in Z) \wedge$$
$$(\forall y. y \in Y. \Rightarrow \exists x. x \in X \wedge \langle x, y \rangle \in Z) \wedge$$
$$(\forall x \forall y \forall z \forall u. \langle x, y \rangle \in Z \wedge \langle z, u \rangle \in Z \Rightarrow (x = z \Leftrightarrow y = u))$$

TARSKI: def 1
$$x \in \{y\} \Leftrightarrow x = y$$

TARSKI: def 2
$$x \in \{y, z\} \Leftrightarrow x = y \vee x = z$$

TARSKI: def 4
$$x \in \bigcup X \Leftrightarrow \exists Y. x \in Y \wedge Y \in X$$

TARSKI: 2
$$(\forall x. x \in X \Leftrightarrow x \in Y) \Rightarrow X = Y$$

TARSKI: 7
$$x \in X \Rightarrow \exists Y. Y \in X \wedge \neg \exists x. x \in X \wedge x \in Y$$

TARSKI: sch 1
$$(\forall x \forall y \forall z. P[x, y] \wedge P[x, z] \Rightarrow y = z) \Rightarrow$$
$$(\exists X. \forall x. x \in X \Leftrightarrow \exists y. y \in A \wedge P[y, x])$$

TARSKI: 9
$$\exists M. N \in M \wedge (\forall X \forall Y. X \in M \wedge Y \subseteq X \Rightarrow Y \in M) \wedge$$
$$(\forall X. X \in M \Rightarrow \exists Z. Z \in M \wedge \forall Y. Y \subseteq X \Rightarrow Y \in Z) \wedge$$
$$(\forall X. X \subseteq M \Rightarrow X \sim M \vee X \in M)$$

translation to untyped logic

for i being Integer holds $i \geq 0$ iff i is Nat

$$\forall i : \text{Integer}. i \geq 0 \Leftrightarrow (i : \text{Nat})$$

↓

$$\forall i. \text{Integer}(i) \Rightarrow [i \geq 0 \Leftrightarrow \text{Nat}(i)]$$

types are just **predicates** that the system manages automatically

dependent types with n arguments are predicates with $n + 1$ arguments

the type system as typing rules

symbolic notation

x		term variables
f		function symbols
M		mode symbols
α		attribute symbols
t	$::= x \mid f(\vec{t})$	terms
R	$::= \star \mid M(\vec{t})$	radix types
a	$::= \alpha \mid \bar{\alpha}$	adjectives
T	$::= \vec{a} R$	types
J	$::= \cdot \mid t : T \mid T \leq T \mid \exists T \mid \alpha/T$	judgment elements
D	$::= x : T$	declarations
Δ	$::= \vec{D}$	
Γ	$::= \overrightarrow{[\Delta](J)}$	
	$\Gamma; \Delta \vdash J$	judgments

rules

twenty-two typing rules

three examples:

$$\frac{}{; \vdash \cdot}$$

mode definition:

$$\frac{\Gamma; \vec{x} : \vec{T} \vdash \exists T'}{\Gamma, [\vec{x} : \vec{T}](M(\vec{x}) \leq T'), [\vec{x} : \vec{T}](\exists M(\vec{x})); \vdash \cdot} M \notin \Gamma$$

conditional cluster:

$$\frac{\Gamma; \Delta \vdash \vec{a} T' \leq T' \quad \Gamma; \Delta \vdash \vec{a}' T' \leq T'}{\Gamma, [\Delta](\vec{a} T' \leq \vec{a}' T'); \vdash \cdot}$$

correctness

translating judgments

type judgment \rightarrow first order sequent

$\text{int} \leq \star, \exists \text{int}, \text{pos}/\text{int}, \exists \text{pos int}; x : \text{pos int} \vdash x : \star$

\rightarrow

$(\forall x. \text{int}(x) \Rightarrow \top), (\exists x. \text{int}(x)), \top, (\exists x. \text{pos}(x) \wedge \text{int}(x)), (\text{pos}(x) \wedge \text{int}(x)) \vdash \top$

main theorem

'the type system is correct'

derivable judgment \rightarrow provable sequent

outlook

system in the paper is an idealization

```
definition let n be Nat;
```

```
  redefine mode Element of n -> Element of n + 1;
```

```
end;
```

according to the rules from the paper we have

Element of n \rightarrow **Element of n + 1** \rightarrow **Element of n + 2** \rightarrow ...

in the actual Mizar system we just have

Element of n \rightarrow **Element of n + 1**

why not have something like the Mizar type system yourself?

- the Mizar proof language is well-known to be nice
- the Mizar type system is less known, but very nice too
- every system can have the Mizar type system as a layer on top

