

Strengthening the Computational Power of the Mizar Checker

Czesław Byliński
University of Białystok
bylinski@math.uwb.edu.pl

The Mizar Project

- The Mizar Project is an attempt to create a computer system for writing, checking and accumulating whole mathematics. The original goal of the Mizar project was to design and implement a software environment that supports writing traditional mathematics papers.
- Mizar is the name of the language that is both rigorous and close to mathematical vernacular, designed by Andrzej Trybulec.
- Mizar is the computer system for checking and collecting articles written in Mizar language.
- *Mizared* mathematical knowledge is collected in *the Mizar Mathematical Library* (MML).

A bit of history

- The project Mizar started 30 years ago (1973) in Poland under the leadership of Andrzej Trybulec.
- Its original goal was to design and implement of software environment to assist the process of preparing mathematical papers. The project can be seen as an attempt to develop software environment for writing traditional mathematical papers, where classical logic and set theory form the basis of all future developments.
- After several year of experiments with Mizar Languages and its implementations (1980 - Mizar 2, Mizar MSE – 1981, 1987 - Mizar 4) in 1988 the design of the language was completed by A. Trybulec. This language is named simply Mizar and its implementation on PC is called PC-Mizar.

The Mizar System

- The Mizar system is implemented on several platforms: Intel x86 based computers with MS Windows, Linux, FreeBSD, Solaris and currently on non Intel platform: Darwin/Mac OS.
- The whole Mizar system including verifier is coded in Pascal using Free Pascal compiler. It is compatible with GNU Pascal Compiler and Delphi/Kylix.
- Current version of the system: 7.0.8
- The Mizar System for download is available at:
<http://www.mizar.org/system/>

Mizar Mathematical Library

- The main effort in the Mizar Project has been in building the mathematical library of *mizared* articles (a scripts in mizar language).
- *Mizared* mathematical knowledge is collected in the Mizar Mathematical Library (MML).
- The development of MML started with axiomatics of Tarski-Grothendieck set theory and axiomatic description of built-in notions.
- We collected till now more than 870 Mizar articles.
- The cross-references allowed in Mizar enable to use proved mathematical facts and defined notions in new articles.

The Mizar Mathematical Library

- Most of the library texts formalizes the fundamentals of introductory mathematics. Some of the texts contribute more advanced results.
- The proof of the Jordan curve theory is being continued
- The mizar formalization of a book : *A Compendium of Continuous Lattices* by G. Giers, K.H. Hoffman, K. Keimel, j.D. Lawson, M. Mislove, and D. S. Scott, Springer Verlag, 1980. The work is not completed. The current state of formalization is described at: <http://megrez.mizar.org/ccl/>. The work was done by 16 author in 57 mizar articles.

The Mizar Mathematical Library

Statistics

- A large body of mathematics has been formalized in the Mizar Language and proof-checked by the Mizar verifier
- 35 524 theorems
- 5630 cluster registrations (existential, conditional, functor)
- 6 854 definitions
- 709 schemes
- 870 articles (65 MB)
- 424 628 cross-references between articles in inferences
- 120 authors, about 20 of them active on a long term basis

The Mizar inference checker

- The Mizar inference checker is a classical logic prover. The checker works well as proof assistant for Mizared article. The inference checker uses model elimination with stress on processing speed, not power.
- It is not powerful enough with as it cannot help to do „high school problems” by themselves. Mathematical practice shows that even in formal proofs some easy background reasoning should be reduced.
- We would like to see similar to computer algebra systems technics incorporated into the Mizar system.

The Mizar inference checker extensions

- The power of the Mizar inference checker is extended by two ways.
 - the properties that can be associated with Mizar definitions
 - the requirements directive.
- Both these features influence generating of equality classes in the EQUALIZER - the Mizar checker's module responsible for the equality calculus .
- They provide some extra equalities used in the process of inference justification. It extensively reduces the number of references in the Mizar Mathematical Library.

Preproperties

- The properties in Mizar are implemented for predicates (constructors of formulae) and functors (constructors of terms).
- The predicate properties include:
 - symmetry, antisymmetry, asymmetry, reflexivity, irreflexivity and connectedness.
- The functor properties are:
 - commutativity, idempotence for binary functors and
 - involutiveness, projectivity for unary functors.
- The properties are paired with a justification of suitable correctness conditions.

Predicate Properties

:: Axiomatic definition

definition let x, X be set;

pred x in X ;

antisymmetry;

end;

:: BOOLE

definition

let X, Y be set;

pred X meets Y means :Def5:

ex x being set st x in X & x in Y ;

symmetry;

antonym X misses Y ;

end;

Predicate Properties

:: NAT_1

Definition

let k,l be natural number;

pred k divides l means :Def3:

ex t being natural number st $l = k * t$;

reflexivity

proof

let i be natural number;

$i = i * 1$;

hence thesis;

end;

end;

Predicate Properties

:: BOOLE

```
definition let X,Y be set;  
  pred X c< Y means :Def9:  
    X c= Y & X <> Y;  
  irreflexivity;  
end;
```

Predicate Properties

:: ORDINAL1

definition

let A,B be Ordinal;

redefine pred A c= B;

connectedness

proof

let A,B be Ordinal;

A in B or A = B or B in A by Th24;

hence thesis by Def2;

end;

end;

Functor Properties

:: BOOLE

definition

let X,Y be set;

func $X \vee Y \rightarrow$ set means :Def2:

x in it iff x in X or x in Y;

existence proof

...

end;

uniqueness proof

...

end;

commutativity;

idempotence;

end;

Functor Properties

:: RELAT_1

definition

let R be Relation;

func R~ -> Relation means :Def7:

[x,y] in it iff [y,x] in R;

existence proof

...

end;

uniqueness proof

...

end;

involutiveness;

end;

Functor Properties

:: ABSVALUE

definition

let x be real number;

func abs x -> real number equals :Def1:

 x if $0 \leq x$

 otherwise -x;

coherence;

consistency;

projectivity by REAL_1:66;

end;

REAL_1:66 refers to theorem from MML:

theorem :: REAL_1:66

for x being real number holds $x < 0$ iff $0 < -x$

Requirements

- The requirements directive, that is comparatively new in Mizar allows for special processing of selected constructors. Unlike the properties, the following concerns the *environ* part of a Mizar article. During the accomodation stage of processing an article with the requirements directive, the accomodator program imports some built-in concepts for selected constructors.
- They are contained in special files associated with suitable articles where the constructors are defined.
- As yet, the special files are:
 - HIDDEN
 - BOOLE
 - SUBSET
 - ARYTM
 - REAL.

Requirements HIDDEN

- This directive is automatically included during accomodation of every article and therefore shouldn't be used explicitly.
- It identifies the objects defined in axiomatic file HIDDEN, i.e.
 - mode *set* - the most general \M mode and any other mode widens to it
 - predicate '=' - The fundamental equality predicate '=' is extensional which means that two objects of the same kind (atomic formulae, types, functors, attributes) are equal when their arguments are equal
 - predicate in – membership of a set
- Thanks to the identification provided by requirements HIDDEN it is used internally wherever the most basic type is needed, ex. while generating various correctness conditions.

Requirements BOOLE

- Processing an article with requirements BOOLE Mizar checker treats specially the constructors provided by the definitions: the empty set and set theoretical join, meet, difference and symmetric difference.
- It allows the following often used equations to be accepted without any external justification:
 - $X \vee \{\} = X$
 - $X \wedge \{\} = \{\}$
 - $X \setminus \{\} = X!$,
 - $\{\} \setminus X = \{\}$
 - $\{\} \setminus + \setminus X = \{\}$
- The empty set gets also additional 'natural' properties:
 - $x \text{ in } \{\}$ implies contradiction
 - $x \text{ in } X$ yields X is not equal to $\{\}$.

Requirements SUBSET

- This requirements directive concerns the definition of predicate inclusion, power set, attribute empty, mode 'Element of' and mode 'Subset of'.
- It strongly interacts with requirements BOOLE by unifying functor $\{ \}$ and attribute empty.
- Moreover,
 - the formula $x \text{ in } X$ is equivalent to $x \text{ is Element of } X \ \& \ X \text{ is non empty}$,
 - formula $X \text{ c= } Y$ automatically yields also $X \text{ is Subset of } Y$ and vice versa,
 - the property of the form $x \text{ in } X \ \& \ X \text{ c= } Y$ implies $x \text{ in } Y$ is incorporated as well - Formerly it was an extensively used MML theorem BOOLE:11. By the above, it also holds for 'the Element of' and 'Subset of' modes.

Requirements REAL

- This directive identifies the ordering relation of real numbers (\leq) and the attributes of real numbers: positive and negative. Each occurrence of $x \leq y$ is processed to the following attribute calculus:
 - If x is positive then y is positive
 - If x is negative then y is negative
 - If x is non negative then y is non negative
 - If x is non positive then y is non positive
- Any negated occurrence of $x \leq y$ yields the following:
 - If x is non positive then y is negative
 - If y is non negative then x is positive

Requirements REAL

- When the directive BOOLE is also used the checker accepts additionally the following clauses:
 - $x \leq y$ & y is non zero & x is non negative
implies y is positive
 - $x \leq y$ & y is non zero & x is non positive
implies y is negative
- The attribute zero is a synonym for empty which can be used for numbers
- The directive requirements REAL is also required by Mizar checker to automatically set the order between numerals.

Requirements NUMBERS

- This directive enables special processing of the successor operator. For example
$$\text{succ}(\text{succ}(2)) = \text{succ}(3)$$
are accepted automatically.
- Another function of this directive is to generate the type *Element of omega* for any numeral, so in fact to provides the correspondence between numerals and numbers defined in MML.
- With an empty requirements REAL numerals are just names for (not fixed) sets.
- This directive requires also SUBSET to be present in requirements directive in order to understand the mode 'Element of'.

Requirements ARITHM

- This directive identifies the basic operations on complex numbers: addition (+), multiplication (*), the negative element (-), the inverse element (⁻¹), subtraction (-) and division (/) and also imaginary unit (<i>).
(Note: The original text contains a typo "substraction" which has been corrected to "subtraction".)
- The above functor are used in the Mizar checker to calculate result of operations on rational numerals (constants). For example: $2*2 = 4$ and $3*2 = 6$ etc.
- The newest algorithm in Mizar checker is based on polynomials as a representation of values of result of above listed operations on complex numbers (inverse element and division are not applied). For example the checker accept the equalities:

:: REAL_2:101

$$(a - b) * (e - d) = a * e - a * d - b * e + b * d;$$

Requirements ARITHM

$a-(b+c)=a-b-c$

proof

thus $a-(b+c) = a+ -(b+c) + 0$ by XCMPLX_0:def 8
. $= a+ -(b+c) + (b+ -b)$ by XCMPLX_0:def 6
. $= a+ -(b+c) + b+ -b + 0$ by AXIOMS:13
. $= a+ -(b+c) + b+ -b + (c + -c)$ by XCMPLX_0:def 6
. $= a+ -(b+c) + b+ -b + c + -c$ by AXIOMS:13
. $= a+(-(b+c) + b)+ -b + c + -c$ by AXIOMS:13
. $= a+ -b + (-(b+c) + b) + c + -c$ by AXIOMS:13
. $= a+ -b + (-(b+c) + b + c) + -c$ by AXIOMS:13
. $= a+ -b + -c + (-(b+c) + b + c)$ by AXIOMS:13
. $= a+ -b + -c + (-(b+c)+ (b + c))$ by AXIOMS:13
. $= a+ -b + -c + 0$ by XCMPLX_0:def 6
. $= a-b + -c$ by XCMPLX_0:def 8
. $= a-b-c$ by XCMPLX_0:def 8;

end;

Requirements ARITHM

- The equality from above example with directive requirements ARITHM is obvious for Mizar checker. The checker accept as true:

$$a-(b+c)=a-b-c;$$

- This implementation allows for significant reduction of users input in reasonnings involving numbers.
- Quite complicated equalities are now accepted by the checker wit no additional justification, like the following example:

$$z*2*(x+y)/2-z*z = ((z*x-(z+-1))+z*(x-z-y))*2''$$

Conclusions

- The idea of implementing such features was based on statistical observations showing very extensive usage of special constructs.
- Introduction of these technics has a strong influence on the maintaining of the MML.
- However, the distinction between properties and requirements is not clear. In some cases it is hard to decide which of these technics could be more efficient.
- Requirements are much more flexible, but on the other hand, properties are a regular language construct and are not so much system dependent.

Conclusions

- Some of the features currently implemented as requirements could be transformed in future into some kind of properties.
- In particular it concerns the properties of neutral elements on algebraic operations. Still there is a discussion on what should be the best syntax for such a neutrality property.
- At the moment, in the scope of interest there are also associativity and transitivity properties. However, they still remain in the to-do list due to some problems with efficient implementation.
- The implementation of similar algorithm as described for requirements ARITHM based on polynomial values can be implemented for other more general operation than operations on complex numbers.