A Guide to the Mizar Soft Type System*

Adam Naumowicz and Josef Urban

[1] University of Bialystok, Poland
[2] Czech Technical University in Prague, Czech Republic

Introduction Mizar [1, 3] is in a way both typed and untyped. In a foundational sense, Mizar is based on untyped set theory. The set-theoretical world initially consists of many objects of "just one type". However, the objects can have various properties (a number, ordinal number, complex number, Conway number, a relation, function, complex function, complex matrix), however none of them is considered to be of "foundational importance", and all these properties are treated as equal "adjectives" or "attributes", which are semantically just (dependent) predicates.

Typically, mathematicians are interested only in some particular properties of the objects, and fluently shift and focus between them, and take some of them as granted in various contexts. Thus, the set of natural numbers can once be treated as a measurable cardinal, another time as a subset of real (and thus "obviously" complex) numbers. A group can simultaneously be a subgroup of other groups. The Mizar "typing" mechanisms are concerned with providing such fluid and "obvious" treatment of the concepts, once the particular relations have been established *in the underlying logic*. This is especially important in large mathematical theories, where complicated and unpredictable hierarchies (ontologies) between the defined concepts arise, and it is often impossible to pre-design a "perfect type system" between the objects, that would specify upfront all the disjointness, inclusion, and other relations. An important part of the Mizar typing mechanisms is thus dynamic type change.

There have been numerous attempts to reconstruct elements of this type system in order to translate the mathematical data encoded in the Mizar language into other common mathematical data exchange formats like OMDoc [4], other proof assistants like HOL Light [6] or Isabelle [5]. The soft-typing mechanisms of Mizar have been actively used in MML already before 1990, i.e., before languages like Haskell started to be developed and provided motivation for the recent soft-typing ("type-class") mechanisms in systems like Coq and Isabelle. A particular advantage of the soft-typing approach is its straightforward translation to first-order ATP formats. This enabled the first encouraging ATP experiments over the whole Mizar library [8], leading to the expansion of hammer-style ITP methods in the last decade.

Types, modes, attributes and adjectives When any variable is introduced in Mizar, its **type** must be given. And for any term, the verifier computes its unique type. Types are then used in quantified and qualifying formulas, for parsing, semantic analysis, overloading resolution, and inferring object properties. Simple types in Mizar are constructed using **modes** and the constructors of **adjectives** are called **attributes**. Mizar supports two kinds of mode definitions: modes defined as a collection (called a cluster) of adjectives associated with an already defined radix type to which they may be applied, called expandable modes, and modes that define a type with an explicit definiens that must be fulfilled for an object to have that type. One of the features of the Mizar type system is that the types must be non-empty, i.e. there must exist at least one object of a given type. This restriction is introduced to guarantee that the formalized theory always has some denotation. Mode definitions thus require a proof of that fact stated in the form of the `existence` condition. The mother type in a mode definition is used to specify the direct predecessor of the defined mode in the tree of Mizar types (representing the type widening relation). The type constructed with the new mode widens to its mother type. The widening relation also takes into account the adjectives that come with types, i.e. the type with a shorter list of (comparable) adjectives is considered to be wider. Mizar types

*The authors are supported by COST Action CA15123. J. Urban was funded by ERC project 649043 – AI4REASON.

are **dependent**. They can have an empty list of arguments, but most commonly they have explicit and/or implicit arguments. Adjectives can also be expressed with their own visible arguments, e.g., `n-dimensional`, or `X-valued`.

Type Change Mechanisms Types of mathematical objects defined in the Mizar library form a sup-semilattice with widening (subtyping) relation as the order [2]. There are two hierarchies of types: the main one based on the type `set`, and the other based on the notion of structure. The most general type in Mizar (to which both sets and structures widen) is called `object`. Structures in Mizar can be used to model mathematical notions like groups, topological spaces, categories, etc. which are usually represented as tuples. Mizar supports multiple inheritance of structures that makes a whole hierarchy of interrelated structures available in the Mizar library, with the `1-sorted` structure being the common ancestor of almost all other structures. An important extension of the Mizar structure system is the possibility to define structures parameterized by arbitrary sets, or other structures.

The effective (semantic) type of a given Mizar term is determined by a number of factors - most importantly, by the available (imported from the library or introduced earlier in the same formalization) redefinitions and adjective registrations. **Redefinitions** are used to change the definiens or type for some constructor if such a change is provable with possibly more specific arguments. Depending on the kind of the redefined constructor and the redefined part, each redefinition induces a corresponding correctness condition that guarantees that the new definition is compatible with the old one. In the Mizar language, the common name **registration** refers to several kinds of Mizar features connected with automatic processing of the type information based on adjectives [7]. Grouping adjectives in so called clusters (hence the keyword cluster used in their syntax) enables automation of some type inference rules. Existential registrations are used to secure the nonemptiness of Mizar types. The dependencies of adjectives recorded as conditional registrations are used automatically by the Mizar verifier.

Sometimes, for syntactic (identification) purposes, e.g. to force the system use one of a number of matching redefinitions, the type of a term can be explicitely qualified to one which is less specific, e.g. `1 qua real number` whereas in standard environments the constant has the type `natural number` and then appropriate (more specific) definitions apply to it. The `reconsider` statement forces the system to treat any given term as if its type was the one stated (with extra justification provided), e.g. `reconsider R as Field` whereas the actual type of the variable `R` might be `Ring`. It is usually used if a particular type is required by some construct (e.g. definitional expansion) and the fact that a term has this type requires extra reasoning after the term is introduced in a proof.

References

[1] Bancerek, G. et al., Mizar: State-of-the-Art and Beyond. In M. Kerber et al. (Eds.), Intelligent Computer Mathematics, CICM 2015, LNAI 9150, 261-279, 2015.

[2] Bancerek. G., On the Structure of Mizar Types. ENTCS 85(7), 6985, 2003.

[3] Grabowski, A., Korniłowicz, A., Naumowicz, A., Four Decades of Mizar - Foreword. Journal of Automated Reasoning 55(3), pp. 191-198, 2015

[4] Iancu, M., Kohlhase, M., Rabe, F., Urban, J., The Mizar Mathematical Library in OMDoc: Translation and Applications, 191-202, Journal of Automated Reasoning (50:2), Springer 2013.

[5] Kaliszyk, C., Pąk, K., Urban, J., Towards a Mizar environment for Isabelle: foundations and language. CPP 2016: 58-65.

[6] Kunčar, O., Reconstruction of the Mizar Type System in the HOL Light System. Proc. of WDS'10.

[7] Naumowicz, A., Enhanced Processing of Adjectives in Mizar. In A. Grabowski and A. Naumowicz (Eds.), Computer Reconstruction of the Body of Mathematics, Studies in Logic, Grammar and Rhetoric 18(31), 89-101, 2009.

[8] J. Urban. MPTP - Motivation, Implementation, First Experiments. *Journal of Automated Reasoning*, 33(3-4):319–339, 2004.