Registrations vs Redefinitions in MIZAR

Artur Korniłowicz
Institute of Informatics, University of Białystok
K. Ciołkowskiego 1M, 15-245 Białystok, Poland
arturk@mizar.org

Abstract

In this paper we briefly discuss two constructions of the MIZAR language – redefinitions and registrations. We focus on practical aspects of using them in MIZAR texts to be effectively processed by the MIZAR VERIFIER. We describe situations when redefinitions can be and should be replaced by corresponding registrations.

1 Introduction

The MIZAR system [Ban15, Gra15] is a computer system invented for computer-assisted verification of mathematical papers. One of the main components of the system is the MIZAR language – a formal language designed for writing mathematical papers readable for humans and effectively processed by computers. The language consists of rules for writing first-order mathematical formulas, proofs, and also syntactic constructions to launch specialized algorithms increasing computational power of the VERIFIER (e.g. term identifications, term reductions [Kor13], flexary connectives [Kor15:b], property registrations [Nau04], etc.).

In this paper, in Sec. 2, we focus on two particular constructions – redefinitions and registrations, especially how they can be used in MIZAR texts to be effectively processed by the MIZAR VERIFIER. In Sec. 3, we present two examples taken from the Mizar Mathematical Library which illustrate our discussion.

2 Redefinitions and Registrations

In MIZAR, redefinitions can be used for three different purposes: a) to specialize (if provable) result types of defined functors and modes, b) to change definientia of notions (predicates, attributes, functors, and modes), and c) to declare properties of particular constructors [Nau04]. The first application influences the identification of operations and types, the second application can be used for the processing definitional expansions [Kor15:a], and the third one can be used for the justification of chosen statements automatically (without explicit references to appropriate theorems). An important feature of the processing redefinitions performed by the VERIFIER is that the order of redefinitions accessible in a given text (redefinitions imported from the database MML or declared in the text) is important. The last redefinition of a notion applicable for given arguments is applied.

On the other hand, registrations can be used for three other purposes[1]: a) to register the existence of objects satisfying required properties written as adjectives (existential registrations), b) to declare that some objects possess chosen properties (functorial registrations), and c) to declare that all elements of some type which satisfy a set of properties satisfy also another set of properties (conditional registrations). A significant feature of the processing registrations is that the order of registrations accessible in a given text plays no role (in opposite to the processing redefinitions), all registrations which can be applied for given arguments are applied.

From the point of view of this paper, the fact that only the last redefinition of a given notion is considered in the process of computing types of objects, but all registrations are, is crucial. It is the fundamental reason for

[1]The MIZAR word `registration` is also used in other contexts, to introduce items like: `identify`, `sethood`, `reduce`.

which registrations should be used instead of redefinitions, whenever possible. The question now is what are the situations when redefinitions can be replaced by corresponding registrations. Before we answer this question, let us mention that the MIZAR system provides two ways to introduce new types: not expandable (really new constructors) and expandable (shortcuts for collections of adjectives assigned to radix types)[2]. Now, we can formulate a rule which defines situations when redefinitions can be replaced by registrations: *If the result type of a functor and the mother type of a mode to which the original type of the functor and the mode is redefined is an expandable type, then such redefinitions are replaceable by a functorial registration in the case of functors and by a conditional registration in the case of modes.* Practical examples which depict the rule are presented in the next section.

3 Examples from the Mizar Mathematical Library

3.1 Functors

Let us consider the functor `Balls(x)` [Sko98] which defines a family of balls in a topological space generated by a metric space:

```
definition
  let M be non empty MetrStruct, x be Point of TopSpaceMetr(M);
  func Balls(x) -> Subset-Family of TopSpaceMetr(M)
  ...
end;
```

To declare that `Balls(x)` constitutes a base in the space, the authors used the redefinition:

```
definition
  let M be non empty MetrSpace, x be Point of TopSpaceMetr(M);
  redefine func Balls(x) -> Basis of x;
end;
```

But, because the mode `Basis of x` [Try97] is defined as an expandable mode:

```
definition
  let T be non empty TopStruct, x be Point of T;
  mode Basis of x is open x-quasi_basis Subset-Family of T;
end;
```

the redefinition can be replaced by the functorial registration:

```
registration
  let M be non empty MetrSpace, x be Point of TopSpaceMetr(M);
  cluster Balls(x) -> open x-quasi_basis;
end;
```

3.2 Modes

To exemplify how redefinitions of modes can be replaced by conditional registrations, let us consider the mode `Element of D` [Ban92], where `D` is a set containing only trees:

```
definition
  let D be constituted-Trees non empty set;
  redefine mode Element of D -> Tree;
end;
```

Because the mode `Tree` [Ban90] is defined as an expandable mode:

```
definition
  mode Tree is non empty Tree-like set;
end;
```

[2]In fact, there is another way to define new types – structures, but it is not relevant to the topic.

the redefinition can be replaced by the conditional registration:

```
registration
  let D be constituted-Trees non empty set;
  cluster -> non empty Tree-like for Element of D;
end;
```

3.3 Experiments

To detect all cases in the Mizar Mathematical Library when redefinitions of functors and modes could be replaced by corresponding registration, a special tool has been implemented by the author of the paper. In the Mizar Version 8.1.05 working with the MML Version 5.37.1267, 89 possible replacements of redefinitions were found.[3] A revision [Ala11, Gra07, Pak14] of the MML was proposed to the Library Committee.

References

[Ala11] Alama, J., Kohlhase, M., Mamane, L., Naumowicz, A., Rudnicki, P., Urban, J.: Licensing the Mizar Mathematical Library. In: Davenport, J.H. et al. (eds.) Proceedings of the 18th Calculemus and 10th International Conference on Intelligent Computer Mathematics. Lecture Notes in Computer Science, vol. 6824, 149–163. Springer-Verlag, Berlin, Heidelberg (2011), `http://dx.doi.org/10.1007/978-3-642-22673-1_11`

[Ban90] Bancerek, G.: Introduction to trees. Formalized Mathematics 1(2), 421–427 (1990), `http://fm.mizar.org/1990-1/pdf1-2/trees_1.pdf`

[Ban92] Bancerek, G.: Sets and functions of trees and joining operations of trees. Formalized Mathematics 3(2), 195–204 (1992), `http://fm.mizar.org/1992-3/pdf3-2/trees_3.pdf`

[Ban15] Bancerek, G., Byliński, C., Grabowski, A., Korniłowicz, A., Matuszewski, R., Naumowicz, A., Pąk, K., Urban, J.: Mizar: State-of-the-art and beyond. In: Kerber, M. et al. (eds.): Intelligent Computer Mathematics – International Conference, CICM 2015, Washington, DC, USA, Proceedings, Lecture Notes in Computer Science, vol. 9150, 261–279, Springer (2015), `http://dx.doi.org/10.1007/978-3-319-20615-8_17`

[Gra15] Grabowski, A., Korniłowicz, A., Naumowicz, A.: Four decades of Mizar. Journal of Automated Reasoning 55(3), 191–198 (2015), `http://dx.doi.org/10.1007/s10817-015-9345-1`

[Gra07] Grabowski, A., Schwarzweller, C.: Revisions as an essential tool to maintain mathematical repositories. In: Proceedings of the 14th Symposium on Towards Mechanized Mathematical Assistants: 6th International Conference. 235–249. Calculemus '07 / MKM '07, Springer-Verlag, Berlin, Heidelberg (2007), `http://dx.doi.org/10.1007/978-3-540-73086-6_20`

[Kor13] Korniłowicz, A.: On rewriting rules in Mizar. Journal of Automated Reasoning 50(2), 203–210 (2013), `http://dx.doi.org/10.1007/s10817-012-9261-6`

[Kor15:a] Korniłowicz, A.: Definitional expansions in Mizar. Journal of Automated Reasoning 55(3), 257–268 (2015), `http://dx.doi.org/10.1007/s10817-015-9331-7`

[Kor15:b] Korniłowicz, A.: Flexary connectives in Mizar. Computer Languages, Systems & Structures 44, 238–250 (2015), `http://dx.doi.org/10.1016/j.cl.2015.07.002`

[Nau04] Naumowicz, A., Byliński, C.: Improving Mizar texts with properties and requirements. In: Asperti, A. et al. (eds.) Mathematical Knowledge Management, Third International Conference, Proceedings. Lecture Notes in Computer Science, vol. 3119, 290–301 (2004), `http://dx.doi.org/10.1007/978-3-540-27818-4_21`

[Pak14] Pąk, K.: Improving legibility of natural deduction proofs is not trivial. Logical Methods in Computer Science 10(3), 1–30 (2014), `http://dx.doi.org/10.2168/LMCS-10(3:23)2014`

[Sko98] Skorulski, B.: First-countable, sequential, and Frechet spaces. Formalized Mathematics 7(1), 81–86 (1998), `http://fm.mizar.org/1998-7/pdf7-1/frechet.pdf`

[3] Computations were carried out at the Computer Center of University of Białystok `http://uco.uwb.edu.pl`

[Try97] Trybulec, A.: Baire spaces, Sober spaces. Formalized Mathematics 6(2), 289–294 (1997), http://fm.mizar.org/1997-6/pdf6-2/yellow_8.pdf