Testing Mizar User Interactivity in a University-level Introductory Course on Foundations of Mathematics

Adam Naumowicz

Institute of Informatics
University of Bialystok, Poland
adamn@math.uwb.edu.pl

Abstract

In this paper we describe selected interactivity aspects of using the Mizar proof assistant by inexperienced users. The presented analysis is based on the data collected during a one-semester university-level introductory mathematical course for computer science undergraduate students.

1 Introduction

The Mizar system [BBG+15] has been created to support developing students' mathematical reasoning skills by interacting with intelligent computer software capable of creating and proof checking formal mathematics [TKNK13]. Mizar, however, is not a typical interactive theorem prover in the sense characteristic to several other popular proof systems [HUW14]. Instead, its design utilizes the mode of interaction usually found in various source code compilers processing a complete input file in a series of lexical and semantic passes.

This mode of interaction may seem more adequate to batch processing multiple or massive data rather than real-time human-computer interplay. But, especially when tailored with a dedicated user interface, in particular J. Urban's Mizar Mode for Emacs [Urb06], the system enables a fairly interactive user experience of flexible gap-filling work on a plain text human-readable proof sketch with the system pointing out proof steps that require justification. The user may then focus on any reported gap in the proof with no imposed order as soon as the whole text has been parsed [GKN10].

For many years, various versions of Mizar have been applied in many educational settings, most typically at the university level to support introductory as well as advanced courses. In this work we present results based on the data collected during a one-semester university-level introductory mathematical course for computer science undergraduate students. The students hadn't had any previous contact with Mizar (or any other formal system, for that matter), so their workings can approximate the problems typically encountered by all inexperienced users.

2 Course Setting

The presented work is based on a course on foundations of mathematics for computer science undergraduates at the University of Bialystok in Poland, using the methodology described e.g. in [BZ07] or [RZ05]. The most essential difference between such a course and Mizar-based research aimed at formalizing more advanced

In: C. Kaliszyk, E. Brady, J. Davenport, W.M. Farmer, A. Kohlhase, M. Kohlhase, D. Müller, K. Pąk, and C. Sacerdoti Coen (eds.): Joint Proceedings of the FMM and LML Workshops, Doctoral Program and Work in Progress at the Conference on Intelligent Computer Mathematics 2019 co-located with the 12th Conference on Intelligent Computer Mathematics (CICM 2019), Prague, Czech Republic, July 8–12, 2019, published at http://ceur-ws.org

mathematical theories (c.f. [Nau06]) is the selection of available background knowledge the students have to become acquainted with before they can work with the system. A standard Mizar user is expected to work directly on top of the Mizar Mathematical Library (MML) providing a uniform centralized repository of formal definitions and theorems comprising all main mathematical theories [BBG+18]. This requires that the users first master interacting with the environment of the large database[Nau17]. To avoid such complications, students may be provided with a restricted environment containing definitions of the needed notions only. In this particular case of a one semester introductory course, the subject notions included elementary set operations, binary relations and natural numbers with induction.

There were 70 students who enrolled in this course for their first winter semester (15 weeks with one 90-minute class each week). The course had a rather low overall attendance rate (52.7%), with a typical peak at the beginning, and later in the middle of the semester (classes 5-10), which can be attributed to mid-semester activities (tests), see fig. 1. Despite this low attendance, interacting with the system using Emacs's Mizar mode did not cause any special problems.

Figure 1: Class attendance statistics during the semester

The students worked in groups of max. 15 persons. Except for three test sessions with randomized individual tasks, they were allowed to collaborate on solving common task sets usually consisting of five formal proofs to be developed during each class. Everyone could interact with the system at their individual pace and acquire their own way of understanding the system's feedback.

2.1 Recording the Interactions

Students worked on local computers with the necessary database already installed. The task sets were provided together in a ready-made file with a complete environment, so the students' job was just to complete the missing proofs. Whenever a student typed some input and called the Mizar verifier from within the Emacs editor, a customized Mizar mode recorded relevant data in a special log file. In particular, the time when the verifier was called and the complete input files were recorded including any error messages reported by the proof checker. At the end of each session students uploaded their log files to their individual accounts on the teachers' server.

During the experiment, the solutions of 553 task sets were recorded. The data revealed that overall the group made 24326 calls to the verifier, which accounts for c.a. 43.99 calls per user per class session. With such simple tasks, the time needed for the verifier to check the input file is negligible (usually less than 1s.), so the sheer average value suggests a typical call being made more or less every two minutes after some thinking and typing. However, the exact data shows that individual interaction strategies varied among the students.

Fig. 2 shows that a considerable number of users called the verifier less than 10 times during a session, clearly in preference of typing longer chunks of text rather than checking the text frequently. On the other hand, there were also task set recordings showing as many as 264 calls to the verifier during one session. That way of interaction indicates more of a guessing approach into finding the proof and using the proving capabilities of Mizar to construct the proof semi-automatically.

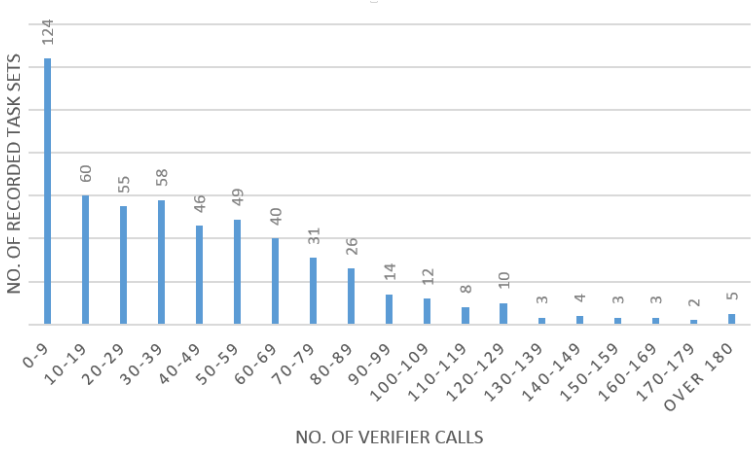

Figure 2: The number of verifier runs invoked by the students

2.2 Kinds of Errors Reported

The source codes of developed tasks contained 107 different errors reported (out of total 496 documented error messages issued by the Mizar verifier). Table 1 shows the top 20 (according to the frequency of occurrence) error codes together with the corresponding error messages presented to the users.

Table 1: Top 20 Most Frequent Error Codes and Messages

Occurrences	**Code**	**Message**
17017	4	This inference is not accepted
14320	70	Something remains to be proved
4519	395	Justification expected
3719	396	Formula expected
3137	51	Invalid conclusion
2578	330	Unexpected end of an item (perhaps ";" missing)
2342	321	Predicate symbol or "is" expected
1969	391	Incorrect beginning of a text item
1919	215	No pairing "end" for this word
1905	214	"end" missing
1712	52	Invalid assumption
1662	143	No implicit qualification
1582	55	Invalid generalization
1436	144	Unknown label
1064	131	No reserved type for a variable, free in the default type
942	164	Nothing to link
733	60	Something remains to be proved in this case
633	165	Unknown functor format
620	216	Unexpected "end"
603	153	Unknown predicate format

Not surprisingly, the list is opened with the most common error `*4` meaning that a given step needs to be justified with more information to be accepted by the checker as an obvious consequence of available references. However, the other items are not all that obvious. There are some errors reported by the scanner, parser and analyzer modules. Their frequent occurrences in the students' tasks indicate e.g. that the description might need less cryptic forms. For the teacher, this sort of feedback is essential in showing which Mizar constructs are less intuitive from the perspective of a new user and therefore require more explanation when they are being introduced to the students. Naturally, a typical teaching session presents ways of 'how to do things the right way'

rather than 'what not to do'. But anticipating potential problems based on their frequency may help minimizing the number of common errors encountered by students if they have been forewarned about them.

On the other hand, some of the issues can only be resolved by extending/changing a bit of the language's grammar (c.f. [Nau16]). In this particular case, a typical error results from the restriction of straightforward linking to a statements in a previous line using the keyword `then`. In consequence, the error `*164` (position 16 in Table 1) is reported whenever making such a link has been tried in the context of compound conditions. An experimental version of the Mizar software reflecting the relaxed syntax exists now and can be used for evaluating the change e.g. in some student classes[1]. It should be noted, however, that until the new syntax is generally accepted, articles written in this new "dialect" could not be accepted for inclusion to the MML.

3 Conclusions

The analysis of data collected during students' courses provided the developers of the Mizar system with valuable information which can be used to improve the user experience of future versions of the system. The statistics of frequently occurring errors encountered by the students are a good approximation of input typical to any novice-user texts in general. A further analysis can also identify typical user scenarios and provide ways to handle them in a more user-friendly manner.

3.0.1 Acknowledgements

The processing and analysis of the Mizar library has been performed using the infrastructure of the University of Bialystok High Performance Computing Center (`http://uco.uwb.edu.pl`).

References

[BBG+15] Grzegorz Bancerek, Czeslaw Bylinski, Adam Grabowski, Artur Kornilowicz, Roman Matuszewski, Adam Naumowicz, Karol Pak, and Josef Urban. Mizar: State-of-the-art and beyond. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM 2015, Washington, DC, USA, July 13-17, 2015, Proceedings*, volume 9150 of *Lecture Notes in Computer Science*, pages 261–279. Springer, 2015.

[BBG+18] Grzegorz Bancerek, Czeslaw Bylinski, Adam Grabowski, Artur Kornilowicz, Roman Matuszewski, Adam Naumowicz, and Karol Pak. The role of the Mizar Mathematical Library for interactive proof development in Mizar. *J. Autom. Reasoning*, 61(1-4):9–32, 2018.

[BZ07] Ewa Borak and Anna Zalewska. Mizar course in logic and set theory. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *Towards Mechanized Mathematical Assistants, 14th Symposium, Calculemus 2007, 6th International Conference, MKM 2007, Hagenberg, Austria, June 27-30, 2007, Proceedings*, volume 4573 of *Lecture Notes in Computer Science*, pages 191–204. Springer, 2007.

[GKN10] Adam Grabowski, Artur Kornilowicz, and Adam Naumowicz. Mizar in a nutshell. *J. Formalized Reasoning*, 3(2):153–245, 2010.

[HUW14] John Harrison, Josef Urban, and Freek Wiedijk. History of interactive theorem proving. In Jörg H. Siekmann, editor, *Computational Logic*, volume 9 of *Handbook of the History of Logic*, pages 135–214. Elsevier, 2014.

[Nau06] Adam Naumowicz. An example of formalizing recent mathematical results in Mizar. *J. Applied Logic*, 4(4):396–413, 2006.

[Nau16] Adam Naumowicz. Linking to compound conditions in Mizar. In Andrea Kohlhase, Paul Libbrecht, Bruce R. Miller, Adam Naumowicz, Walther Neuper, Pedro Quaresma, Frank Wm. Tompa, and Martin Suda, editors, *Joint Proceedings of the FM4M, MathUI, and ThEdu Workshops, Doctoral Program, and Work in Progress at the Conference on Intelligent Computer Mathematics 2016 co-located with the 9th Conference on Intelligent Computer Mathematics (CICM 2016), Bialystok, Poland, July 25-29, 2016.*, volume 1785 of *CEUR Workshop Proceedings*, pages 21–24. CEUR-WS.org, 2016.

[1] `http://mizar.uwb.edu.pl/~softadm/linking/`

[Nau17] Adam Naumowicz. Towards standardized Mizar environments. In Leszek Borzemski, Jerzy Swiatek, and Zofia Wilimowska, editors, *Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology - ISAT 2017 - Part II, Szklarska Poręba, Poland, September 17-19, 2017*, volume 656 of *Advances in Intelligent Systems and Computing*, pages 166–175. Springer, 2017.

[RZ05] Krzysztof Retel and Anna Zalewska. Mizar as a tool for teaching mathematics. *Mechanized Mathematics and Its Applications, Special Issue on 30 Years of Mizar*, 4(1):35–42, March 2005.

[TKNK13] Andrzej Trybulec, Artur Kornilowicz, Adam Naumowicz, and Krystyna Trybulec Kuperberg. Formal mathematics for mathematicians - foreward to the special issue. *J. Autom. Reasoning*, 50(2):119–121, 2013.

[Urb06] Josef Urban. Mizarmode - an integrated proof assistance tool for the Mizar way of formalizing mathematics. *J. Applied Logic*, 4(4):414–427, 2006.