First Order Proof Problems Extracted from an Article in the MIZAR Mathematical Library

Ingo Dahn*
Humboldt University, Berlin

Christoph Wernhard†
Humboldt University, Berlin

1 Introduction

Over the years, interactive theorem provers have built a large body of verified computer mathematics. The ILF MATHEMATICAL LIBRARY aims to make this knowledge available to other systems.

There are several reasons for such a project. One of them is economy. Verification of software and hardware frequently requires the proof of purely mathematical theorems. It is obviously inefficient, to use the time of experts in the design of software or hardware systems to prove such theorems. This should be handed over to mathematicians.

Currently, each interactive theorem prover has its own library of mathematical theorems. And in each of these systems, all theorems are proved from scratch. Again, this is obviously inefficient. Another reason for presenting a collection of mathematical theorems in a unified framework is safety. It should facilitate the verification of theorems in the library of one system by other systems.

A third reason is dynamics of research. New interactive theorem provers should obtain the possibility to show their usability for real-world problems without having to reprove elementary mathematical facts.

Last but not least it is hoped that reproving theorems in a uniform mathematical library will be considered as a challenge to the development of automated theorem provers.

The combination of theorems from different sources will pose some foundational problems. We shall not address these problems here.

The ILF MATHEMATICAL LIBRARY is clearly related with the QED project [1]. However — in contrast with this project — no proof checking is intended. Instead, emphasis is given to proof presentation, proof documentation and to the retrieval of formulas.

2 Representation of the Mizar Mathematical Library

The first library of theorems to be connected with the ILF system is the MIZAR MATHEMATICAL LIBRARY. The Mizar Project [8] is a long-term effort lead by Andrzej Trybulec at the Bialystok Branch of Warsaw University with the goal of providing software to support a working mathematician in proving theorems. Its kernel is a proof verifier for a high-level

*Humboldt University Berlin, Mathematical Institute, Ziegelstr. 13a, D-10099 Berlin, Germany dahn@mathematik.hu-berlin.de

†Humboldt University Berlin, Mathematical Institute, Ziegelstr. 13a, D-10099 Berlin, Germany wernhard@mathematik.hu-berlin.de

logic language. It is used to verify the Mizar library, a collection of mathematical papers, *articles*, written in the Mizar Language. This library has evolved over more than 10 years and consists of more than 20.000 theorems. It can be accessed through WWW servers in Poland and Canada. The ILF MATHEMATICAL LIBRARY augments these servers with extended search functions and — above all — with the possibility to deliver selected parts of the MIZAR MATHEMATICAL LIBRARY in a format which is easy to parse. Its address is

`http://www-irm.mathematik.hu-berlin.de/~ilf/mathlib.html`.

Parsing an original MIZAR article requires parsing the whole part of the library on which it is based. Reservation of variables, redefinition of concepts and hidden parameters are powerful tools that facilitate writing and reading of MIZAR articles. The ILF MATHEMATICAL LIBRARY preserves these tools for the selection of theorems from the library but resolves them in the internal presentation. Hence, parsers for other systems can analyze MIZAR theorems obtained from ILF in a formula-by-formula way.

This is possible because formulas in the ILF MATHEMATICAL LIBRARY have been obtained from a format which is used internally by the MIZAR system when an article is checked. This internal format has been made accessible by Czeslaw Bylinski from the MIZAR group. It distinguishes between different uses of the same symbol and adds hidden parameters. Moreover it contains intermediate proof steps which have been generated by the MIZAR system. However, it does not preserve the formulas as written by the author. ILF tries to recover these original formulas by reversing the transformations performed by MIZAR and by matching the results against the formulas in the MIZAR article.

3 Extracting Proof Problems

In order to select theorems from the MIZAR MATHEMATICAL LIBRARY, predicates and functors must be classified as either "well known", "interesting" or "out of scope". Only definitions and theorems containing at least one interesting concept and no concept which is out of scope will be presented initially. Also the level of detail of proof presentation can be determined.

Selected parts of the library can be restricted at will. It is also possible to add all lemmas and definitions needed to prove a certain theorem or all theorems which make use of a given theorem in their proof. E.g. a certain theorem can be selected together with all those formulas which are referred to in its proof. Then — in principle — other theorem provers can reprove the selected theorems from the quoted assumptions.

The contents of the web page with the selected definitions, theorems and proofs can be delivered as a LaTeX source file. In the same way, the data structure from which that page was produced can be obtained. So far it will be send in a format that should be understood by most Prolog and Lisp parsers. This data structure describes a block structured proof in the ILF standard proof format [5]. This format is also used by the ILF MAIL SERVER. It is described in the documentation of this server[1]. Other formats may be supported in the future.

[1] `http://www-irm.mathematik.hu-berlin.de/~ilf-serv`

4 Set-theoretic Proof Problems

Since current automated theorem provers do not permit the declaration of types, we extracted proof problems only from the MIZAR article *Boolean Properties of Sets* [9]. In this article, *set* is the only type which is used, hence the resulting proof problems can be considered as first order problems by ignoring the type information.

This article contains 97 theorems. The Boolean operators on sets are defined and their basic properties are proved. The proofs use axioms of Grothendieck-Tarski set theory and other theorems proved in this article. For each of these 97 theorems in the article a theory was built consisting of the definitions and theorems which have been referenced in its proof.

When a functor or predicate is defined in a MIZAR article, it is checked whether some standard properties (like commutativity or associativity) are immediate consequences of the definitions. Henceforth, these properties can be used in MIZAR articles without explicit reference — even without reference to the definition itself. Also the definition of the empty set is not explicit in the MIZAR MATHEMATICAL LIBRARY. Therefore, it was necessary to augment the set of used formulas by these properties of the occurring functors and predicates which may have been used tacitly. Consequently, there can be also axioms in the proof problems which are not necessary for the proof.

5 Reproving Mizar Theorems with Automated Theorem Provers

The ILF System ("Integrating Logical Functions") [4] is a Prolog based environment that integrates an interactive proving system with the control of automated provers like SETHEO [6], SPASS [10] OTTER [7] and PROTEIN [2] and a natural language presentation of proofs as LaTeX or HTML documents.

ILF translated the proof problems into input files for the provers CM[2], GANDALF[3], OTTER, SETHEO and SPASS. If necessary, first order formulas were transformed into clauses and equality axioms were added.

Then each of the provers was launched with a CPU limit of 15 seconds on a SPARC 20 workstation[4]. Without special tuning, they solved between 47 and 76 of the proof problems. The total number of proof problems solved by any of the prover is 81. Most of these have been solved within the first few seconds of the prover run.

Two of the 97 proof problems were known unsolvable because their axiomatization according to the theorems and definitions referenced in the MIZAR proof was not complete. As it later turned out, under certain circumstances MIZAR allows the tacit use of definition expansion in proofs.

Since conventional clausal normal form[5] was used in the experiments, proving theorems with a complex logical structure — e.g. logical equivalences — was most difficult. In some cases, the inherent logical complexity was hidden in term structures. E. g. simple equalities involving the symmetric difference operation can require complex resolution proofs when they have to be proved using the axiom of extensionality and the definition of symmetric difference.

[2] A PTTP/SETHEO-like prover implemented and integrated in Prolog
[3] A winner in the CADE-14 system competition
[4] SETHEO ran distributed on a number of machines of which the SPARC 20 was the fastest
[5] As opposed to definitional or nested normal forms

Most of the problems considered become easy for the automated theorem provers, when involved concepts are first expanded using their definitions. Some "difficult" problems are then even proved by the usual simplifications performed at normal form translation. A similar experience, with the suggestion of a strategy — *controlled definition instantiation* — is discussed in [3]. This emphasizes that the use of automated theorem provers should be preceded by a domain specific preprocessing of the input theory.

Another experiment distributed the proof problems over the local network. It used several provers in a competitive way. However, in most cases were a proof was found the run time of the prover was so short compared with the time needed for preprocessing and scheduling that the launch sequence of the provers determined the winner of the competition.

6 Proving More Theorems

MIZAR has built-in arithmetic of integers. Hence, in order to reprove theorems on numbers, sequences etc. it will be necessary to extend automated theorem provers with the ability to evaluate arithmetic expressions.

However, the main obstacle to reprove more theorems from the MIZAR MATHEMATICAL LIBRARY will be the type system. This type system is order-sorted and polymorphic.

There is a largest type called *set*. Given a type t, a variable x of this type and a predicate $p(x)$, the types of all x satisfying p can be formed as a subtype of t. Thus, each type could be described as a finite collection of predicates which select objects of this type from the most general type *set*. It should be noted, that p may contain other object variables beside x. There are no type variables. Types must be proved to be non-empty in order to make use of their defining properties.

In terms of set theoretical concepts, types denote classes. Some of these classes denote sets. In fact, by the collection principle, if all objects of type t are members of some object set a, then there is some set b such that

$$\forall\,(x : set)\,(x \in b \leftrightarrow \exists\,(y : t)\,(x = y))\,.$$

We call such a t a *small type* with *extension* b. For most real-world applications small types are sufficient. Especially, when t_1, t_2 are small types with extensions b_1, b_2 respectively, then there is a small type $function\,(b_1, b_2)$ consisting of all functions from b_1 into b_2. On the other hand, there are important mathematical concepts which cannot be described by small types. Examples are the class of all functions or the class of all groups.

Equality is built-in as a predicate with arguments of type *set*. Since each type is a subtype of *set*, arguments of arbitrary different types can occur on both sides of the equality sign.

7 Typed Problems for First Order Provers

The naive transformation of MIZAR proof problems into first order problems, replacing quantification of typed variables by relativizations of quantifiers, leads to the use of proof search to check type correctness. But for the MIZAR type system, there is an algorithm to determine the least type of a term. Given this least type, it is straightforward to check whether it is legal to instantiate a typed variable with this term. The restriction to legal variable instantiations can reduce the search space for automated theorem provers considerably. We mention that the SPASS theorem prover treats unary predicates as types. Other first order theorem provers can be used successfully for problems formulated in a typed language by encoding type information into terms so that only well-typed unifications can be performed during the proof process. But the full benefit of search space reduction cannot be realized unless the terms used to encode type information, are treated in a specific way by the prover.

In MIZAR, the smallest type of a term can depend on the types of the arguments. E.g. when R_1, R_2 are relations, the intersection $R_1 \cap R_2$ has the type *relation*, though $\cap$ is declared as an operator with values of type *set* only. It is an open problem to modify first order theorem provers in order to deal with this kind of type checking.

8 Conclusion

The reported experiments show, that most theorems proved interactively in [9] can be proved automatically by state-of-the-art theorem provers in a very short time. This indicates, that automated theorem provers can be a useful support for interactive theorem proving. On the other hand, we described some limitations of current automated provers which have to be overcome in order to facilitate their application to more complex problems.

References

[1] Anonymous: The QED Manifesto. In Proc. CADE-12, pp. 238–251, Springer, 1994.

[2] P. Baumgarter, U. Furbach: Protein: A PROver with a Therory Extension INterface. In Proc. CADE-12, pp. 769–773, Springer, 1994.

[3] W. W. Bledsoe: Non-resolution Theorem Proving. Artificial Intelligence 9, 1977, pp. 1–35.

[4] B. I. Dahn, J. Gehne, Th. Honigmann and A. Wolf: Integration of Automated and Interactive Theorem Proving in ILF. Proc. CADE-14, Townsville, LNAI 1249, 1997, pp. 57–60.

[5] B. I. Dahn and A. Wolf: A Calculus Supporting Structured Proofs. Journal for Information Processing and Cybernetics (EIK), (5–6): pp. 261–276, 1994. In Proc. CADE-14, pp. 57–60, Springer, 1997.

[6] C. Goller, R. Letz, K. Mayr, and J. Schumann: SETHEO V3.2: Recent Developments (System Abstract). In Proc. CADE-12, pp. 778–782, Springer, 1994.

[7] W. McCune: Otter 2.0. In Proc. CADE-10, pp. 663–664, Springer, 1990.

[8] P. Rudnicki: An Overview of the Mizar Project. Proceedings of the 1992 Workshop on Types for Proofs and Programs, Chalmers University of Technology, Bastad 1992.

[9] Z. Trybulec and H. 'Swiceczkowska: Boolean Properties of Sets. Journal of Formalized Mathematics, Vol.1, 1990.

[10] C. Weidenbach, B. Gaede, and G. Rock: Spass & Flotter, Version 0.42. In Proc. CADE-13, pp. 141–145, Springer, 1996.