Escape to Mizar from ATPs

Jesse Alama*
Center for Artificial Intelligence
New University of Lisbon
Portugal
j.alama@fct.unl.pt, http://centria.di.fct.unl.pt/~alama/

Abstract

We announce a tool for mapping E derivations to Mizar proofs. Our mapping complements earlier work that generates problems for automated theorem provers from Mizar inference checking problems. We describe the tool, explain the mapping, and show how we solved some of the difficulties that arise in mapping proofs between different logical formalisms, even when they are based on the same notion of logical consequence, as Mizar and E are (namely, first-order classical logic with identity).

1 Introduction

The problem of generating a mapping between proofs in different formats is an important research problem. Proofs coming from a many sources can be found today. There are about as many implemented proof formats as there are different systems for interactive and automated theorem proving, not to mention the "pure" proof formats coming from mathematical logic. Even within the latter we find a plethora of possibilities. If we pick a Hilbert-style system, there is a choice about which axioms and rules of inference to pick. Even natural deduction comes in a number of shapes: Jáskowski, Gentzen, Fitch, Suppes... [15]. It seems likely that as the use of proof systems grows we will need to have better tools for mapping between different; this need has been recognized for decades [22, 1], and it still seems we have some way to go.

This paper discusses the problem of transforming derivations output by the E automated theorem prover into Mizar texts.

Mizar is a language for writing mathematical texts in a "natural" style. It features a kind of natural deduction proof language. The library of knowledge formalized in Mizar, the Mizar Mathematical Library (MML), is quite advanced, going from the axioms of set theory to graduate-level pure mathematics. For the purposes of this paper we are not interested in the MML. Instead, we view Mizar as a language and a suite of tools for carrying out arbitrary reasoning in first-order classical logic.

Our work is available at

https://github.com/jessealama/tptp4mizar

Related work is discussed in Section 2. Section 3 discusses an important preliminary exercise to mapping derivations, and which is perhaps already of interest: mapping an arbitrary TPTP problem (not necessarily derivations) into a corresponding Mizar article. The generated Mizar text has the same flat structure as initial TPTP problem from which it comes. Section 4 is the heart of the paper; it discusses in detail translation from E derivations to Mizar proofs. Because of the fine-grained level of detail offered by E and the simple multi-premise "obvious inference" rule of

*Supported by the ESF research project *Dialogical Foundations of Semantics* within the ESF Eurocores program *LogICCC* (funded by the Portuguese Science Foundation, FCT LogICCC/0001/2007). Research for this paper was partially done while a visiting fellow at the Isaac Newton Institute for the Mathematical Sciences in the programme 'Semantics & Syntax'. Josef Urban inspired this project and provided many helpful suggestions. Artur Korniłowicz clarified some important details of Mizar proofs.

Mizar, the mapping is more or less straightforward, save for *skolemization* and *resolution*, neither of which have direct analogues in "human friendly" Mizar texts. Skolemization is discussed in Section 4.2 and our treatment of resolution is discussed in 4.3. The problem of making the generated Mizar texts more humanly comprehensible is discussed in Section 4.4. Section 5 concludes and proposes applications and further opportunities for development. Appendix A is a complete example of a text (a solution to the Dreadbury Mansion puzzle found by E, translated to Mizar) produced by our translation.

2 Related work

In recent years there is an interest in adding automation to interactive theorem proving systems. An important challenge is to make sense, at the level of the interactive theorem prover, of the solution produced by external automated reasoning tools. Such *proof reconstruction* has been done for Isabelle/HOL [13]. There, the problem of finding an Isabelle/HOL text suitable for solving an inference problem P is done as follows:

1. Translate P to a first-order theorem proving problem P^*.

2. Solve P^* using an automated theorem prover, yielding solution S^*.

3. Translate S^* into a Isabelle/HOL text, yielding a solution S of the original problem.

The work described in this paper could be used to provide a similar service for Mizar. It is interesting to note that in the case of Mizar the semantics of the source logic and the logic of the external theorem prover are the same: first-order classical logic with identity. In the Isabelle/HOL case, at step (1) there is a potential loss of information because of a mismatch of Isabelle/HOL's logic and the logic of the ATPs used to solve problems (which may not in any case matter at step (3)). In the Mizar context, two-thirds (steps (1) and (2)) of the problem has been solved [17]; our work was motivated by that paper. Steps toward (3) have been taken in the form of Urban's ott2miz[1]. In fact, more than 2/3 of the problem is solved. Our work here builds on ott2miz by accounting for the clause normal form transformation, rather than starting with the clause normal form of a problem. Our translated proofs thus start with (the Mizar form of) the relevant initial formulas, which arguably improves the readability of the proofs. Moreover, our tool works with arbitrary TPTP problems and TSTP derivations (produced by E), rather than with Otter proof objects. The restriction to E is not essential; there is no inherent obstacle to extending our work to handle TSTP derivations produced by other automated theorem provers, provided that these derivations are sufficiently detailed, like E's. One must acknowledge, of course, that providing high-quality, fine-grained proof objects is a challenging practical problem for automated theorem provers.

To account for the clausal normal form transformation, one needs to deal with skolemization. This is a well-known issue in discussions surrounding proof objects for automated theorem provers [3]. Interestingly, our method for handling skolemization is quite analogous to the handling of quantifiers in the problem opposite ours, namely, converting Mizar proofs to TSTP derivations [21] in the setting of MPTP (Mizar Problems for Theorem Provers) [20]. There, Henkin-type implications are a natural solution to the problem of justifying a substitution instance of a formula given that its generalization is justified. Our justification of skolemization steps is virtually the same as this; see Section 4.2 for details.

[1] See its homepage https://github.com/JUrban/ott2miz and its announcement http://mizar.uwb.edu.pl/forum/archive/0306/msg00000.html on the Mizar users mailing list.

An export and cross-verification of Mizar proofs by ATPs has been carried out [21]. Such work is an inverse of ours because it goes from Mizar proofs to ATP problems.

We do not intend to enter into a discussion about the proof identity problem. For a discussion, see Došen [5]. Certainly the intension behind the mapping is to preserve whatever abstract proof expressed by the E derivation. That the E derivation and the Mizar text generated from it are isomorphic will be clarified by the discussion below of the translation algorithm. Mapping such as the one discussed here can help contribute to a concrete investigation of the proof identity problem, which in fact motivates the project reported here. The reader need not share the author's interest in the proof identity problem to understand what follows.

It is well-known that derivations carried out in clause-based calculi (such as resolution and kindred methods) tend to be difficult to understand, if not downright inscrutable. An important problem for the automated reasoning community for many years is to find methods whereby we can understand machine-discovered proofs, such as resolution refutations. One approach to this problem is to map resolution derivations into natural deduction proofs. Much work has been done in this direction [11, 12, 7, 6, 9, 10]. The transformations we employ are rather simple. Because of the coarseness of Mizar's proof apparatus (there is essentially only one rule of inference that subsumes most of the traditional introduction and elimination rules of natural deduction), we need not be concerned with a translation that preserves the fine structure of an E derivation. To "clean up" the generated text, we take advantage of the various proof "enhancers" bundled with the standard Mizar distribution [8, §4.6]. These enhancers suggest compressions of a Mizar text that make it more parsimonious while preserving its semantics. In the end, though, it would seem that the judgment of whether an "enhanced" Mizar text is the best representative of a resolution proof is something that has to be left to the reader.

3 Translating TPTP problems into Mizar texts

In this section we describe a method for generating a Mizar text from an arbitrary (first-order) TPTP problem [18]. TPTP problems are not themselves derivations, so this mapping is not the heart of our work. However, it was an important first step to mapping derivations to Mizar proofs because it revealed some difficulties that had to be solved in the translation of formulas part of the mapping of derivations to Mizar proofs. The next section is devoted to the proof mapping problem.

TPTP is a language for specifying automated reasoning problems. One states some axioms and definitions, and perhaps a conjecture. Although TPTP has in recent years been extended to support various extensions of the language of first-order logic, we are interested in this paper only in the first-order part of TPTP.

To construct a Mizar text from a TPTP problem, one first identifies the function and predicate symbols of the TPTP problem and creates a *environment* for the text. This step is necessary because Mizar is a richer language than TPTP. Given a well-formed TPTP file, one can simply determine, for each symbol appearing in it, whether it is a function or a predicate, and what its arity is. Since (at the time of writing) first-order TPTP focuses only on the case of *one-sorted* first-order logic, there is no issue about the sorts of the arguments and values. The language of Mizar, on the other hand, permits overloading of various kinds and has (dependent) types. There is no issue of inferring from a purported Mizar text what the predicate and function symbols are. To implement this complexity, when working with Mizar on specifies in advance its so-called environment. The environment provides the necessary information to make sense of the text.

Constructing an environment for a Mizar text amounts to creating a handful of XML files. Normally, one does not develop Mizar texts from scratch but rather builds on some preexisting formalizations. Since we not interested in using the Mizar library, we cannot use the usual toolchain. Instead, we create a fresh environment with respect to which the generated Mizar text is sensible. This environment gives a meaning to the TPTP problem even if the TPTP "problem" is actually a derivation. Constructing Mizar proofs from E derivations (expressed in the TSTP notation) is the subject of the next section.

4 Translating E derivations into Mizar texts

This section discusses the main part of our contribution: mapping E derivations to Mizar texts.

The input to our procedure is an E derivation in TSTP format [19] (the standard E distribution comes with a tool, *epclextract*, which can translate derivations expressed in E's custom proof language into proofs in the desired format).

The Mizar proof is isomorphic to the E derivation in the sense that the premises P_{E} of the E derivation map to a set P_{Mizar} of the same cardinality and the same logical form, and the conclusion c_{E} of the E derivation maps directly to the sole theorem c_{Mizar} of the Mizar text. The logical content of the two proofs are the same because E and Mizar are both based on first-order classical logic. Because E's calculus is based essentially on clauses while Mizar works with formulas, some hurdles need to be overcome when mapping (i) the part of an E derivation dealing with converting the input problem to clause normal form, and (ii) applications of the rule of resolution. We describe the mapping and our solution to these difficulties.

As one might expect, the mapping between an E derivation, which operates essentially on clauses, is not a simple one-to-one mapping of formulas (more precisely, clauses) to formulas. E's calculus can to a large extent be recognized by Mizar in the sense that most steps in an E derivation do map directly to (single) steps in the generated Mizar text. Two classes of inferences, though, raises some problems: skolemization and resolution, which are the heart of a resolution calculus such as the one behind E.

It seems to be a hard AI problem to transform arbitrary resolution proofs into human-comprehensible natural deductions. There often seems to be a artificial "flavor" of such proofs that no spice can overcome. Still, some simple organizational principles can help to make the proof more manageable. (Later in Section 4.4 we will see some stronger syntactic and semantic methods, going beyond the simple structural guidelines we are about to discuss, for "enhancing" the generated proofs even further.) Section 4.1 discusses the overall organization of the generated proof. In Section 4.2 we discuss the skolemization problem. In Section 4.3 we discuss the problem of resolution.

4.1 Global and local organization of the proof

The first batch of transformation do not compress the derivations in any way: every step in the TSTP derivation appears in the Mizar output. However, the refutation is "groomed" in the following ways:

1. Linearly order the formulas.

   Unlike TPTP/TSTP problems, where order of formulas is immaterial, the order of formulas in Mizar has to be coherent. We topologically sort the input ordered in the obvious way (if conclusion A uses formula B as a premise, then B should appear earlier than A) and work with a linear order.

2. Because one can "reserve" variables globally in Mizar, one can strip away the initial universal prefix of clauses-as-formulas.

This transformation not only makes the formulas appearing in the proof shorter and hence more readable, it helps to keep Mizar's by rule of inference aligned with the various clause-oriented rules of inference in E's calculus (clauses don't have quantifiers).

3. Separate reasoning done among the axioms (establishing lemmas) from the application of lemmas toward the derivation of $\perp$.

   In other words, we distinguish conclusions that depend on the conjecture from conclusions that are independent of it.

4. Separate those lemmas that are used in the refutation proper from those that not used. (I.e., distinguish lemmas that are used in the refutation proper from the lemmas that are used only to prove other lemmas.)

Step (1) is strongly necessary because if a conclusion is drawn in a Mizar text from a premise that has not yet been introduced, this is a fatal error. Step (2) is needed for a deeper reason: if we were to deal always with explicit universal closures of formulas, we would quickly start to outstrip the notion of obvious inference on which Mizar is based. Steps (3)–(5) are not necessary; there is nothing wrong with disregarding those organizational principles. However, there is a cost: abandoning them results in an undifferentiated, disorganized melange of inferences, a mere "print out" in Mizar form of the E derivation.

A refutation starts with some axioms, a conjecture, and proceeds by negating the conjecture formula and deriving $\perp$ by reasoning with the axioms and the negation of the conjecture. Mizar texts in the Mizar Mathematical Library, on the other hand, if read at their toplevel, are intended to be consistent: given some axioms and lemmas, one states theorems. The *proofs* of these lemmas and theorems may use proof by contradiction, but that is done inside a proof block, outside of which any contradictory assumptions and conclusions derived therein are no longer "accessible". However, a TSTP representation of a refutation is a flat sequence of formulas ending with a contradiction: the axioms, the conjecture, the negation of the conjecture, and conclusions drawn among the axioms and the negation of the conjecture all at the same level.

To capture the spirit of proof by contradiction while ensuring that the toplevel content of the generated Mizar article is coherent (or at least not manifestly incoherent), we refactor E refutations into so-called diffuse reasoning blocks. We write:

```
theorem φ
proof
  now
    assume ¬φ;
    S1: ⟨conclusion 1⟩ by ...;
    S2: ⟨conclusion 2⟩ by ...;
    ...
    Sn: ⟨conclusion n⟩ by ...;
    thus contradiction by S_a1, S_a2, ..., S_am
  end;
  hence thesis;
end;
```

This concludes the discussion of the organization of the generated Mizar proof.

4.2 Skolemization

E's finely detailed proof output contains not simply the derivation of $\perp$ starting from the clause form of the input formulas. E can also record the transformation of the input formulas into clause form. It is important to preserve these inferences because they give information about what was actually given to E; throwing away this information strikes us as unwelcome because one would have to work harder to make sense of the overall proof.

If we insist on preserving skolemization steps in the Mizar output, then we have a difficulty in accounting for them. Carrying out this task is a well-known issue in generating proof objects [3, 4]. The difficulty is that skolem functions are curious creatures in an interactive setting like Mizar's. Introducing a function in into a Mizar text requires that the use can prove existence and uniqueness of its definiens. But what is the definiens of a skolem function?

We solve the problem by introducing, as part of the environment of an article (and not in the generated text), a "definition" for skolem functions in the following manner. To take a simple example, suppose we have proved $\forall x \exists y \varphi$ and we have that $\forall x \varphi[y := f(x)]$ is "derived" from this, in the sense that it is is the conclusion of a skolemization step. We covertly introduce at this point a new definition:

$$(\forall x \exists y \varphi) \rightarrow \forall x \varphi[y := f(x)]$$

This formula does not have the usual shape of an explicit definition of a function. One wonders how one would prove existence and uniqueness for this definiens. We do not address these problems; in effect, the above implication is treated as a new axiom.

Our approach seems defensible to us. After all, E does not give a proof that introducing the skolem function is acceptable, so there is no step in the E derivation that would contain the needed information. Giving a proof in Mizar that would justify skolemization steps is in fact possible. One introduces a new type τf inhabited by definition by those objects that satisfy the sentence $\forall x \exists y \varphi$, prove that the type is inhabited by exploiting the fact that the domain of interpretation of any first-order structure is non-empty, and finally defining f outright using Mizar's built-in Hilbert choice operator. Initial experiments with this approach to skolemization lead us to turn off this feature by default because it introduces "noise" into the Mizar proof. We know that skolemization is a valid transformation, so it seems excessive to us to put an explicit justification of every skolemization step.

There is one limitation with the current approach to skolemization at the moment. We require that all skolemization steps introduce exactly one skolem function.

4.3 Resolution

Targeting Mizar is sensible because of the presence of a single rule of inference, called `by`, which takes a variable number of premises. The intended meaning of an application

$$\frac{\varphi_1, \ldots, \varphi_n}{\varphi} \texttt{by}$$

of `by` is that φ is an "obvious" inference from premises $\varphi_1, \ldots, \varphi_n$. See Davis [2] and Rudnicki [16] for more information about the the tradition of "obvious inference" in which Mizar works. The implementation in Mizar diverges somewhat from these proposals, but roughly speaking a conclusion is obtained by an "obvious inference" from some premises if there is a Herbrand proof of the conclusion in which we have chosen at most one substitution instance of each premise.

One important difficulty for mapping arbitrary resolution proofs to Mizar texts is that Mizar's notion of "obvious inference" overlaps with various forms of resolution, but is neither weaker nor stronger than resolution. The consequence of this is that it is generally not the case that an application of resolution can be mapped to a single acceptable application of Mizar's `by` rule. Consider the following example:

Example 1 (Non-obvious resolution inference). Consider the inference

$$\frac{\neg l(x) \vee d(x) \qquad \neg l(x) \vee \neg d(x) \vee \neg d(y)}{\neg l(x) \vee \neg d(y)}\ \text{Resolution}$$

Here l and d are unary predicate symbols and x and y are variables; all formulas should be read as implicitly universally quantified. This application of resolution simply eliminates $d(x)$ from the premises.

If we map the two premises and the conclusion of the application of resolution to three Mizar theorems and attempt justify the mapped conclusion simply by appealing by name to the two mapped premises, then we are asking to check an application of `by`, as follows:

$$\frac{\forall x\,[\neg l(x) \vee d(x)] \qquad \forall x, y\,[\neg l(x) \vee \neg d(x) \vee \neg d(y)]}{\forall x, y\,[\neg l(x) \vee \neg d(y)]}\ \texttt{by}$$

The problem here is that we cannot choose a single substitution instance of the premises such that we can find a Herbrand derivation, and hence the inference is non-obvious even though it is essentially (i.e., at the clause level) a single application of propositional resolution.

The reason for the difficulty is that we are making things difficulty for ourselves by working at the level of formulas rather than clauses. A solution is available: map the application of resolution not to a single application of Mizar's `by` rule, but to a proof:

```
((not l x) or (not d y))
proof
  A: (not l x) or (not d x)              by Premise1;
  B: (not l x) or (not d x) or (not d y) by Premise2;
  thus thesis by A,B;
end;
```

There is an application of Mizar's `by` rule at the end, whose conclusion is `thesis`, i.e., the formula to be proved at that point in the proof. We solve the problem by reasoning with substitution instances of the premises, obtained by taking instances of the premises (these are `A` and `B`, respectively) rather than with whole universal formulas. Note that the substitution instances are not built from constants and function symbols, but from (fixed) variables.

4.4 Compressing Mizar proofs

The "epicycles" of resolution notwithstanding, Mizar is able to compress many of E's proof steps: many steps can be combined into a single acceptable application of Mizar's `by` rule of inference. For example, if φ is inferred from φ' from variable renaming, and φ' is inferred by an application of conjunction elimination to φ'', typically in the Mizar setting φ can be inferred from φ'' alone by a single application of `by`. This is typical for most of the fine-grained rules of E's calculus: their applications are acceptable according to Mizar's `by`, and often they can be composed (sometimes multiple times) while still being acceptable to `by`. Other rules in E's proof calculus that can often be eliminated are variable rewritings, putting formulas into negation normal form, reordering of literals in clauses (but recall that Mizar proofs are written at the level of full first-order logic, not in a clause language). More interesting compressions exploit the gap between "obvious inference" and E's more articulated calculus.

Compressing proofs helps us to get a sense of what the proof is about. The Mizar notion of obvious inference has been tested through daily work with substantial mathematical proofs for decades, and thus enjoys a time-tested robustness (though it is not always uncontroversial). It seems to be an open problem to specify what we mean by the "true" or "best" view of a proof. When Mizar texts come from E proofs, Mizar finds that the steps are usually excessively detailed (i.e., most steps are obvious) and can be compressed. On the other hand, often the whole proof cannot be compressed into a single application of `by`. We employ the algorithm discussed in [17]: a simple fixed-point algorithm is used to maximally compress a Mizar text. Thus, by repeatedly attempting to compress the proof until we reach the limits of `by`, we obtain a more parsimonious presentation of the proof.

Proof compression is not without its pitfalls; if one compresses Mizar proofs too much, the Mizar text can become as "inhuman" as the resolution proof from which it comes. This is a well-known phenomenon in the Mizar community. Applying the proof compression tools seems to require a human's *bon sens*. Experience with texts generated by our translation shows that often considerable compression is possible, but at the cost of introducing a new artificial "scent" into the Mizar text.

5 Conclusion and future work

One naturally wants to extend the work here to work with output of other theorem provers, such as Vampire. There is no inherent difficulty in that, though it appears that the TSTP derivations output by Vampire contain different information compared to E proofs; the generic transformations described in Section 4.1 would carry over, but the mapping of skolemization and resolution steps of Sections 4.2 and 4.3 will likely need to be customized for Vampire.

The TPTP language recognizes definitions, but whether an automated theorem prover treats them differently from an axiom is unspecified. In Mizar, definitions play a vital role. After all, Mizar is designed to be a language for developing mathematical theories; only secondarily is it a language for representing solutions to arbitrary reasoning problems, as we are using it in this paper. One could try to detect definitions either by scanning the problem looking for formulas that have the form of definitions, or, if the original TPTP problem is available, one can extract the formulas whose TPTP status is `definition`. Such definition detection and synthesis has no semantic effect, but could make the generated Mizar texts more manageable and perhaps even facilitate new compressions.

At the moment the tool simply translates E derivations to Mizar proofs. A web-based frontend to the translator could help to spur increased usage (and testing) of our system. One can even imagine our tool as part of the SystemOnTPTP suite [18].

An important incompleteness of the current solution is the treatment of equality. Some atomic equational reasoning steps (specifically, inferences involving non-ground equality literals) in E derivations can be non-Mizar-obvious. One possible solution is to use Prover9's Ivy proof objects. Ivy derivations provide some information (namely, which instances of which variables in non-ground literals) that (at present) is missing from E's proof object output.

For the sake of clarity in the mapping of skolemization steps in E derivation to Mizar steps, we restricted attention to those E derivations in which each skolemization step introduces exactly one new skolem function. The restriction does not reflect a weakness of Mizar; it is a merely technical limitation and we intend to remove it.

We have thus completed the cycle started in [17] and returned from ATPs to Mizar. We leave it to the reader to decide whether he wishes to escape again.

References

[1] P.B. Andrews. More on the problem of finding a mapping between clause representation and natural-deduction representation. *Journal of Automated Reasoning*, 7(2):285–286, 1991.

[2] M. Davis. Obvious logical inferences. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 530–531, 1981.

[3] Hans de Nivelle. Extraction of proofs from the clausal normal form transformation. In *Computer Science Logic*, volume 2471 of *Lecture Notes in Computer Science*, pages 921–951, 2002.

[4] Hans de Nivelle. Translation of resolution proofs into short first-order proofs without choice axioms. *Information and Computation*, 199(1-2):24 – 54, 2005.

[5] Kosta Došen. Identity of proofs based on normalization and generality. *Bulletin of Symbolic Logic*, 9:477–503, 2003.

[6] U. Egly and K. Genther. Structuring of computer-generated proofs by cut introduction. *Computational Logic and Proof Theory*, pages 140–152, 1997.

[7] A. Felty and D. Miller. Proof explanation and revision. In *AAAI-86 Proceedings*, 1987.

[8] A. Grabowski, A. Kornilowicz, and A. Naumowicz. Mizar in a nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.

[9] C. Lingenfelder. Structuring computer generated proofs. In *International Joint Conference on Artificial Intelligence*. Citeseer, 1989.

[10] Andreas Meier. System description: Tramp: Transformation of machine-found proofs into natural deduction proofs at the assertion level. In David McAllester, editor, *Automated Deduction - CADE-17*, volume 1831 of *Lecture Notes in Computer Science*, pages 460–464. Springer Berlin / Heidelberg, 2000.

[11] Dale Miller. Expansion tree proofs and their conversion to natural deduction proofs. In R. Shostak, editor, *7th International Conference on Automated Deduction*, volume 170 of *Lecture Notes in Computer Science*, pages 375–393. Springer Berlin / Heidelberg, 1984.

[12] Dale A. Miller. A compact representation of proofs. *Studia Logica*, 46:347–370, 1987. 10.1007/BF00370646.

[13] L. Paulson and Kong Woei Susanto. Source-level proof reconstruction for interactive theorem proving. In Klaus Schneider and Jens Bran, editors, *Theorem Proving in Higher-Order Logics*, volume 4732 of *Lecture Notes in Computer Science*, pages 232–245. Springer, 2007.

[14] F.J. Pelletier. Seventy-five Problems for Testing Automatic Theorem Provers. *Journal of Automated Reasoning*, 2(2):191–216, 1986.

[15] Francis Jeffry Pelletier. A brief history of natural deduction. *History and Philosophy of Logic*, 20(1):1–31, 1999.

[16] P. Rudnicki. Obvious inferences. *Journal of Automated reasoning*, 3(4):383–393, 1987.

[17] Piotr Rudnicki and Josef Urban. Escape to ATP for Mizar. In Pascal Fontaine and Aaron Stump, editors, *PxTP 2011: First International Workshop on Proof eXchange for Theorem Proving*, pages 46–59, 2011.

[18] G. Sutcliffe. The TPTP Problem Library and associated infrastructure: The FOF and CNF parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.

[19] Geoff Sutcliffe, Stephan Schulz, Koen Claessen, and Allen Van Gelder. Using the TPTP language for writing derivations and finite interpretations. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, pages 67–81. Springer Berlin / Heidelberg, 2006.

[20] Josef Urban. MPTP 0.2: Design, implementation, and initial experiments. *Journal of Automated Reasoning*, 37(1-2):21–43, 2006.

[21] Josef Urban and Geoff Sutcliffe. ATP-based cross-verification of Mizar proofs: Method, systems, and first experiments. *Mathematics in Computer Science*, 2(2):231–251, 2008.

[22] Larry Wos. The problem of finding a mapping between clause representation and natural-deduction representation. *Journal of Automated Reasoning*, 6(2):211–212, 1990.

A Pelletier's Dreadbury Mansion Puzzle: From E to Mizar

```
Ax1: ex X1 st (lives X1 & killed X1,agatha) by AXIOMS:1;

Ax2: lives X1 implies (X1 = agatha or X1 = butler or X1 = charles) by AXIOMS:2;

Ax3: killed X1,X2 implies hates X1,X2 by AXIOMS:3;

Ax4: killed X1,X2 implies (not richer X1,X2) by AXIOMS:4;

Ax5: hates agatha,X1 implies (not hates charles,X1) by AXIOMS:5;

Ax6: (not X1 = butler) implies hates agatha,X1 by AXIOMS:6;

Ax7: (not richer X1,agatha) implies hates butler,X1 by AXIOMS:7;

Ax8: hates agatha,X1 implies hates butler,X1 by AXIOMS:8;

Ax9: ex X2 st (not hates X1,X2) by AXIOMS:9;

Ax10: not agatha = butler by AXIOMS:10;

S1: killed skolem1,agatha by Ax1,SKOLEM:def 1;

S2: agatha = skolem1 or butler = skolem1 or charles = skolem1 by Ax2,Ax1,SKOLEM:def 1;

S3: not hates agatha,(skolem2 butler) by Ax9,SKOLEM:def 2,Ax8;

S4: hates charles,agatha or skolem1 = butler or skolem1 = agatha by Ax3,Ax1,SKOLEM:def 1,S2;

S5: butler = (skolem2 butler) by S3,Ax6;

S6: not hates butler,butler by Ax9,SKOLEM:def 2,S5;

S7: hates butler,butler or skolem1 = agatha by Ax4,Ax7,Ax1,SKOLEM:def 1,Ax5,S4,Ax6,Ax10;

S8: skolem1 = agatha by S7,S6;

theorem
killed agatha,agatha
proof
  now
    assume S9: not killed agatha,agatha;
    thus contradiction by S1,S8,S9;
  end;
  hence thesis;
end;
```

Pelletier's Dreadbury Mansion [14] goes as follows:

> Someone who lives in Dreadbury Mansion killed Aunt Agatha. Agatha, the butler, and Charles live in Dreadbury Mansion, and are the only people who live therein. A killer always hates his victim, and is never richer than his victim. Charles hates no one that Aunt Agatha hates. Agatha hates everyone except the butler. The butler hates everyone not richer than Aunt Agatha. The butler hates everyone Aunt Agatha hates. No one hates everyone. Agatha is not the butler.

The problem is: Who killed Aunt Agatha? (Answer: she killed herself.) The problem belongs to the TPTP Problem Library (it is known there as PUZ001+1) and can easily by solved by many automated theorem provers. Above is the result of mapping E's solution to a standalone Mizar text and then compressing it as described in Section 4.4. Two skolem functions `skolem1` (arity 0) and `skolem2` (arity 2) are introduced. There are 10 axioms and 8 steps that do not depend depend on the negation of the conjecture (`killed agatha,agatha`) This problem is solved essentially by forward reasoning from the axioms; proof by contradiction is unnecessary, but that is the nature of E's solution.