Declarative Proof Translation

Cezary Kaliszyk
University of Innsbruck, Austria
University of Warsaw, Poland
cezary.kaliszyk@uibk.ac.at

Karol Pąk
University of Białystok, Poland
pakkarol@uwb.edu.pl

Abstract

Declarative proof styles of different proof assistants include a number of incompatible features. In this paper we discuss and classify the differences between them and propose efficient algorithms for declarative proof outline translation. We demonstrate the practicality of our algorithms by automatically translating the proof outlines in 200 articles from the Mizar Mathematical Library to the Isabelle/Isar proof style. This generates the corresponding theories with 15301 proof outlines accepted by the Isabelle proof checker. The goal of our translation is to produce a declarative proof in the target system that is both accepted and short and therefore readable. For this three kinds of adaptations are required. First, the proof structure often needs to be rebuilt to capture the extensions of the natural deduction rules supported by the systems. Second, the references to previous items and their labels need to be matched and aligned. Finally, adaptations in the annotations of individual proof step may be necessary.

1 Introduction

Declarative proof languages have been included in many proof assistants, since they provide more readable and more maintainable proofs. Examples include Isabelle/Isar [9], the Mizar proof language [5], Lean [4], the Coq declarative proof mode Czar [3], and various declarative proof modes for HOL [10, 11, 6]. They all imitate natural deduction, because it has been developed as a minimal language capable of describing natural logical reasonings. However, all extend or modify natural deduction, usually depending on how they were developed or because of the motivations of the language creators. Some were designed to fit an existing infrastructure (for example an LCF prover), while some focus on imitating the mathematical practice. The largest example of the latter is the Mizar Mathematical Library (MML) [5, 2], which includes many constructs non-standard to natural deduction.

In this paper we discuss the incompatibilities between the declarative styles and propose translations between the features of such languages and showcase this on a large part of the Mizar Mathematical Library. The particular contributions are:
A comparison of the features present in the declarative proof styles (Section 2) and efficient scalable translations that eliminate the features nor present in the other styles (Section 3);

An automated translation of the declarative proof outlines of 200 articles from the Mizar Mathematical Library to Isabelle/Isar (Section 4). The application of the translation gives 15301 declarative toplevel proof outlines accepted by Isabelle in the Isabelle/Mizar object logic [8]. The proof skeleton transformation steps are all automatically correctly justified, but the justifications of the individual Mizar by steps are mostly not covered by the Isabelle/Mizar automation and are assumed.

Related work. We have [7] previously translated the toplevel statements of a smaller part of the MML to Isabelle without any proofs. Many translations between procedural proofs have been proposed in the past. Adams [1] gives an overview of such translations. Additionally he considers the efficiency of such translations, which has been a major issue for proof auditing, for example in the Flyspeck project. Proof translations between declarative proofs and procedural proofs in a single system has been considered before [11].

2 Declarative Proof Styles

We first discuss the features present in the declarative proof modes of different proof assistants and later present a table that compares the presence of these features in the systems (Table 1).

The two earliest declarative proof languages, the Mizar language [5] and Isabelle/Isar [9], differ most as they were developed quite differently. The former started as an extension of the Jaśkowski natural deduction. The latter tried to add declarative natural deduction elements to an LCF style theorem prover, which meant combining declarative proofs with procedural ones. These two styles have influenced declarative proof modes developed since.

A common feature of all such systems is a set of basic natural deduction steps (also referred to as skeleton steps). Matching these steps with the reasoning can be done explicitly, using a so-called reasoning path. The reasoning path is a list of rules used in procedural systems, which describes the process in which the goal needs to be transformed or simplified. We will first discuss the use of reasoning path in the various systems and their advantages, and later discuss other differences that arise.

Isabelle/Isar allows the goal to be transformed and rebuilt in a most flexible manner, however all transformation rules must be provided before the start of an individual reasoning. A drawback of such a solution is, for example, the treatment of the existential quantifier. In order to instantiate it, the suitable term needs to be available before the proof and cannot be constructed in the proof block. A simplification of the reasoning path that removes this restriction has been considered in Lean [4] where the exists.intro rule can be formulated after a witness is obtained.

A further restriction of the reasoning path makes the thesis completely implicit. This has been considered in Mizar, Czar [3], and the two declarative modes for HOL Light (miz3 [10, 11] and Harrison’s Mizar Mode [6], which we will denote shortly MM). In such systems the implicit thesis can be referred to as thesis. A limited procedure for transforming it in every skeleton step is necessary. Additionally, the order of the skeleton steps is mostly specified by the shape of the proved formula. A partial conclusion allows specifying the proved conjunct and proceed to subsequent ones. Czar is most flexible in this respect, since the implicit thesis can be transformed by the reconsider thesis as construction.
### Table 1
Comparison of features present in the declarative proof styles of different proof assistants.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Mizar</th>
<th>Lean</th>
<th>Isabelle/Isar</th>
<th>Czar</th>
<th>miz3</th>
<th>MM_H</th>
</tr>
</thead>
<tbody>
<tr>
<td>reason-path</td>
<td>–</td>
<td>+</td>
<td>+</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>inline ∃_intro</td>
<td>take</td>
<td>ex.intro</td>
<td>–</td>
<td>take</td>
<td>take</td>
<td>take</td>
</tr>
<tr>
<td>unfold</td>
<td>partial</td>
<td>partial</td>
<td>full</td>
<td>full</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>cases</td>
<td>after</td>
<td>before, EM</td>
<td>before, EM</td>
<td>after</td>
<td>after</td>
<td>after</td>
</tr>
<tr>
<td>thesis</td>
<td>thus/hence</td>
<td>show</td>
<td>show/thus</td>
<td>thus</td>
<td>thus</td>
<td>thus</td>
</tr>
<tr>
<td>∃_elim</td>
<td>consider</td>
<td>obtain</td>
<td>obtain</td>
<td>consider</td>
<td>consider</td>
<td>consider</td>
</tr>
<tr>
<td>diffuse</td>
<td>now...end</td>
<td>–</td>
<td>{...}</td>
<td>–</td>
<td>now...end</td>
<td>–</td>
</tr>
</tbody>
</table>

Mizar is the only system that implicitly unfolds user-selected definitions to match the thesis to the provided skeleton steps. Unfolding definitions in all other systems is manual, and often all the occurrences of a given definition must be unfolded together. Isabelle/Isar and Lean include attributes that transform facts before their use (e.g. `[simplified]`).

The proof modes also include two possible ways how reasoning by cases is realized. In the first approach, the user specifies all the cases before the reasoning and then proceeds with each individual case. The second approach allows the user to directly prove the necessary cases. At the end of the reasoning the system will build the alternative based on the explicitly given cases and possibly ask the user to justify that all the cases have been covered. The latter approach has been considered in Mizar, miz3, MM_H, and Czar. In Isabelle and Lean it is necessary to specify the cases (or give a formula $\phi$ for which excluded middle, EM will be used) before the reasoning.

Certain declarative modes support the extraction of information from nested proof blocks without explicitly giving the proof goal. This is referred to as a diffuse statement and supported by Mizar, miz3, and Isabelle/Isar. There are minor differences in the flexibility of such constructions, so we mark them by the corresponding syntax (`now...end` and `{...}`) in Table 1. Similarly, the existential elimination construction may or may not allow linking to the statement about the witness. We again mark this using the corresponding syntax (`obtain / consider`) in the table.

### 3 Translations

In this section we assume the the foundations are compatible and that we know how to translate the syntax of individual statements. A statement syntax translation will be necessary for each pair of systems and we will use one in the next section. A translation between two systems comprises of: rebuilding the proof structure to skeleton steps provided by the systems; adapting the references to previous items and labels; possibly adding the annotations of individual proof steps by the reasoning path. We will attempt to reconstruct the proof structure by introducing a small number of skeleton steps supported by the target system. The skeleton steps will be annotated only with the justification elements necessary in the target system, such as ∀_intro, ⇒_intro, or explicit references to the conclusion (such as show). We discuss below eliminating particular features, if they are not supported by the target system. After the application of these transformation, the resulting proof text needs to be optimized to make use of the special features of the target system and the labels, references, and justifications updated.
∃ introduction. If not supported by the target system, they can be eliminated by introducing a cut with the existential formula available as a lemma and used in the reasoning path or explicitly given by a command, depending on the target system.

diffuse statement. In a similar way, diffuse statements (defined in the previous section 2) can be eliminated from the proof skeleton if they are not supported by the target proof system. For this, the thesis of the proof block needs to be reconstructed and explicitly provided.

cases. Proofs by cases are replaced by a case covering lemma and series of lemmas case → thesis justified by the reasonings given in the source system.

thesis reference. If the target system does not support a reference to the thesis, it is replaced by the formulation extracted from the source system. The only case where the target thesis is used, would be when the original thesis is not modified. For example in Isabelle, the use of proof- allows avoiding a repetition of the whole goal statement.

reasoning path. If the target system does require a reasoning path, the proof needs to be transformed to a shape where we can provide a correct reasoning path. In particular we assume that before any universal quantifier introduction (fix/let depending on the system) the thesis is universally quantified, for implication introduction (assume) it is an implication, and the show/thus is the formula or its first conjunct. This generates quite unnatural parentheses, which can be removed in a post-processing phase. Also note that in some systems (mostly logical frameworks) separating assume steps changes the reasoning path. The transformation follows the diagram:

\[
\begin{align*}
\text{skeleton step} & \quad \text{new thesis} & \quad \text{additional rules} \\
\text{fix/let } x & \quad \forall x. \text{ thesis} & \quad \text{ball} \\
\text{assume } \alpha_1 : \alpha_1 \text{ and } \alpha_2 : \alpha_2 \quad & \quad \alpha_1 \land (\alpha_2 \land (\ldots(\alpha_{n-1} \land \alpha_n)\ldots)) \\n\text{and...and } \alpha_{n-1} : \alpha_{n-1} \text{ and } \alpha_n : \alpha_n & \quad \text{impl}_{1 \to n-1} \text{ times} \\
\text{show/thus } \alpha & \quad \alpha \land \text{ thesis} \\
\text{take term} & \quad \exists x. \text{ thesis}(x:=\text{term}) \\
\end{align*}
\]

where implMI connects uncurry and impl; conjMI is a modification of conjI; ball, bexI are the bounded quantifier introduction rules used with object-level types.

identifier scopes and namespaces. Newly introduced identifiers (x:=term) are also not treated uniformly across systems (for example in Mizar, the second kind of take construction may introduce a variable with the same name). In order to avoid problems, in cases where ambiguities can arise (it will be only 17 cases in all the proofs in the next section), identifiers will be renamed.

final thesis adjustment. The transformations discussed above derive for every block a thesis that is equivalent to the original one, but not always syntactically identical. If it is not identical, we introduce a cut in the target system. Finally the proof is adapted for readability in the target system, removing e.g. references to previous steps if they can be implicit or use then etc. Further refinements of the resulting text are left as future work.

4 Case Study

We have implemented these transformations and applied them to the 200 articles of the Mizar library obtaining natural deduction proof outlines that can be expressed in Isabelle/Isar. Isabelle accepts all the proof outlines, however the current Isabelle/Mizar automation is not able to handle most of the individual proof steps justifications yet, and these are assumed so far. In this section we showcase two original and translated lemmas. For details on the Isabelle/Mizar object logic and its notations we refer to [8].
We proposed translation techniques for the various features present in declarative proof languages and we automatically translated the proof outlines from 200 articles of the MML to Isabelle/Isar. Isabelle accepts all the translated proof outlines and the increase in the proof size imposed by our translation is relatively small (factor 1.7). Future work includes extending the translation to Mizar structures and proof schemes which would allow applying theorem DrinkerParadox:
\[ \exists x. P(x) \rightarrow (\forall y. P(y)) \]

proof:
\[ \begin{align*}
\text{have cases: } (\exists x. \neg P(x)) \lor (\forall x. P(x)) & \text{ by auto} \\
\text{have case1: } (\exists x. \neg P(x)) \rightarrow (\exists t. P(t) \rightarrow (\forall y. P(y))) & \text{ by auto} \\
\text{proof (rule impI)} & \\
\text{assume } \exists x. \neg P(x) & \\
\text{then obtain } x \text{ where } [ty]: x \text{ be set and } \\
A1: \neg P(x) & \text{ by auto} \\
\text{show } \exists t. P(t) \rightarrow (\forall y. P(y)) & \text{ by auto} \\
\text{proof (rule impI)} & \\
\text{assume } \exists x. \neg P(x) & \\
\text{thus } \forall y. P(y) \text{ using A1 by simp} & \\
\text{qed auto} & \\
\text{have case2: } (\forall x. P(x)) \rightarrow (\exists x. P(x) \rightarrow (\forall y. P(y))) & \\
\text{proof (rule impI)} & \\
\text{assume } A2: \forall x. P(x) & \\
\text{obtain } x \text{ where } [ty]: x \text{ be set and } \\
xDef: x = \text{ the set by auto} & \\
\text{show } \exists x. P(x) \rightarrow (\forall y. P(y)) & \\
\text{proof (rule impI)} & \\
\text{assume } \exists x. \neg P(x) & \\
\text{thus } \forall y. P(y) \text{ using A2 by simp} & \\
\text{qed auto} & \\
\text{show } ?\text{thesis using cases case1 case2 by auto} & \\
\text{qed} &
\end{align*} \]

Figure 1 Drinker’s paradox in Mizar and its automated translation to Isabelle. Variables are implicitly typed as set. The example is a schematic extension of Wenzel and Wiedijk’s example comparing Mizar with Isar [9].

In Figure 1 we present a simple proof that showcases the transformations the four different kinds of skeleton step reconstruction, variable rename in take, and uses existential introduction. In the proof automatically translated according to the introduced transformations Isabelle/Mizar’s mauto works as a justification of every step. Every take step requires an additional obtain and type calculation. The proof by cases uses excluded middle, which is supported by Isabelle. Among the 3236 proofs by cases, 1354 required a justification that the considered cases are complete, and the most complex proof involves 16 cases.

Figure 2 showcases a more advanced MML proof, where automated thesis adjustments are also necessary. Also the Isabelle/Mizar automation does not support Mizar’s term generation for properties, so the individual proof step justification required additional facts. These were symmetry \( a+b = b+a \), reductions \( a+b-a = b \), and the reflexivity of \( \leq \). Last was for example necessary to derive B1: \( t<\_0M \) from A1. All other steps were successfully proved by mauto.

Among the 20233 subproofs in MML200, we need the additional cut to transform the thesis in 14827 cases (large majority are the same modulo parentheses). When it comes to definition unfolding, the unfolded definition needs to be explicitly provided. This occurs in 5144 subproofs. Inline existential introduction steps introduce 13027 additional proof blocks.

5 Conclusion

We proposed translation techniques for the various features present in declarative proof languages and we automatically translated the proof outlines from 200 articles of the MML to Isabelle/Isar. Isabelle accepts all the translated proof outlines and the increase in the proof size imposed by our translation is relatively small (factor 1.7). Future work includes extending the translation to Mizar structures and proof schemes which would allow applying...
Declarative Proof Translation

Theorem (Declarative Version)

\[ f(x + t) = f(x) \cdot t + f'(x) \cdot t + o(t) \]

\(0 < t \)

Proof

Assume \( f \) is differentiable on \([a, b]\).

Let \( t \) be an arbitrary real number such that \( 0 < t \).

Consider \( x_0 \) be a real number that satisfies

\[ f(x_0 + t) = f(x_0) + t \cdot f'(x_0) + o(t) \]

Hence, the theorem follows by taking \( x_0 \) as the desired value.

\[ \square \]

Figure 2 The Lagrange theorem in Mizar and its automated translation to Isabelle. The individual proof step justifications have been omitted, and are available in the accompanying formalization.

the techniques to a large subsequent part of the Mizar library. Finally, developing a more powerful Mizar-like automation would be necessary to verify all the individual proof steps.

References


