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 out 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.

2012 ACM Subject Classification Theory of computation → Interactive proof systems

Keywords and phrases Declarative Proof, Translation, Isabelle/Isar, Mizar

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Funding *Cezary Kaliszyk*: ERC starting grant no. 714034 *SMART*
Karol Pąk: the Polish National Science Center granted by decision n°DEC-2015/19/D/ST6/01473

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 C_zar [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 inthe 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 is 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, C_zar [3], and the two declarative modes for HOL Light (miz3 [10, 11] and Harrison's Mizar Mode [6], which we will denote shortly MM_H). 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. C_zar is most flexible in this respect, since the implicit thesis can be transformed by the reconsider thesis as construction.

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 ways reasoning by cases are introduced. 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

	Mizar	Lean	Isabelle/Isar	C_zar	miz3	MM_H
reason-path	–	+	+	–	–	–
inline $\exists_{\mathrm{intro}}$	take	ex.intro	–	take	take	take
unfold	partial	partial	full	full	–	–
cases	after	before, EM	before, EM	after	after	after
thesis	thus/hence	show	show/thus	thus	thus	thus
$\exists_{\mathrm{elim}}$	consider	obtain	obtain	consider	consider	consider
diffuse	now...end	–	{...}	–	now...end	–

Table 1 Comparison of features present in the declarative proof styles of different proof assistants. `miz3` refer's to Wiedijk's Mizar mode for HOL and MM_H refers to Harrison's Mizar mode for HOL. For features present, but where their semantics slightly differ, we mark this with the syntax.

approach has been considered in Mizar, miz3, MM_H, and C_zar. In Isabelle and Lean it is necessary to specify the cases (or give a formula ϕ 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 a compatibility between the foundations and the statement syntax. 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 $\forall_{\mathrm{intro}}$, $\Rightarrow_{\mathrm{intro}}$, 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.

$\exists$ 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 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* $\rightarrow$ *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

```
scheme DrinkerParadox{P[set]}:
  ex x st P[x] implies for y holds P[y]
proof
  per cases;


    suppose ex x st not P[x];
      then consider x such that
A1:     not P[x];
      take x;

      assume P[x];
      hence for y holds P[y] by A1;

    end;


    suppose
A2:     for x holds P[x];

      take x=the set;

      assume P[x];
      thus for y holds P[y] by A2;
    end;


end;
```

```
theorem Drinker_paradox:
  ∃x. P(x) ⟶ (∀y. P(y))
proof-
  have cases: (∃x.¬P(x)) ∨ (∀x. P(x)) by auto
  have case1: (∃x.¬P(x)) ⟶ (∃t. P(t) ⟶ (∀y. P(y)))
  proof(rule impI)
    assume ∃x. ¬P(x)
    then obtain x where [ty]: x be set and
    A1: ¬P(x) by auto
    show ∃t. P(t) ⟶ (∀y. P(y))
    proof(rule bexI[of _ x],rule impI)
      assume P(x)
      thus ∀y. P(y) using A1 by simp
    qed auto
  qed
  have case2: (∀x. P(x)) ⟶ (∃x. P(x) ⟶ (∀y. P(y)))
  proof(rule impI)
    assume A2: ∀x. P(x)
    obtain x where [ty]: x be set and
    xDef: x = the set by auto
    show ∃x. P(x) ⟶ (∀y. P(y))
    proof(rule bexI[of _ x],rule impI)
      assume P(x)
      show ∀y. P(y) using A2 by simp
    qed auto
  qed
  show ?thesis using cases case1 case2 by auto
qed
```

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].

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 fix/let the thesis is universally quantified, for 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:

skeleton step	new thesis	additional rules
fix/let x	$\forall x.$ thesis	ballI
assume A_1:α_1 and A_2:α_2 and...and A_{n-1}:α_{n-1} and A_n:α_n	$\alpha_1 \wedge (\alpha_2 \wedge (\ldots(\alpha_{n-1} \wedge \alpha_n)\ldots))$ $\longrightarrow$ thesis	$\underbrace{\text{impMI}, \ldots, \text{impMI}}_{n-1 \text{ times}}$, impI
show/thus α	$\alpha\wedge$ thesis	conjMI
take term	$\exists x.$ thesis(term:=x)	bexI[of "term"]

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

identifier scopes and namespaces. Newly introduced identifiers (term:=x) are also not treated uniformly across systems (for example in Mizar, the second kinds of take construction may introduce a same variable). 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

```
theorem :: ROLLE:4 Lagrange Theorem
  for x,t be Real st 0<t
    for f be PartFunc of REAL,REAL st
      [.x,x+t.] c= dom f &
      f|[.x,x+t.] is continuous &
      f is_differentiable_on ].x,x+t.[
  ex s be Real st 0<s & s<1 &
    f.(x+t) = f.x + t*diff(f,x+s*t)
proof
  let x,t be Real such that
A1: 0<t;
  let f be PartFunc of REAL,REAL;
  assume [.x,x+t.] c= dom f &
    f|[.x,x+t.] is continuous &
    f is_differentiable_on ].x,x+t.[;
  then consider x0 be Real such that
A2: x0 in ].x,x+t.[ and
A3: diff(f,x0)=(f.(x+t)-f.x)/(x+t-x)
    by...

  take s = (x0-x)/t;


  x0 in {r where r is Real:x<r & r<x+t} by...
  then
A4: ex g be Real st g=x0 & x<g & g<x+t by...
  then 0<x0-x by...
  then 0/t < (x0-x)/t by...
  hence 0<s by ...
  x0-x<t by...
  then (x0-x)/t<t/t by...
  hence s<1 by...
A5: s*t+x = (x0-x)+x by...
  f.x+t*diff(f,x0)=f.x+(f.(x+t)-f.x) by...
  hence thesis by ...

end;
```

```
mtheorem Lagrange:
  ∀x:Real. ∀t:Real. 0_M < t ⟶
   ∀x:PartFunc of ℝ,ℝ.
     (⟦x,x+t⟧ ⊆ dom f ∧
      f|⟦x,x+t⟧ be continuous) ∧
     f is differentiable on ⦇x,x+t⦈ ⟶
  ∃s:Real. 0_M < s ∧ (s < 1_M ∧
   f.(x+t) = f.x + t ∗ diff(f,x + s∗t))
proof(rule ballI,rule ballI,rule impI,rule ballI,rule impI)
  fix x assume [ty]: x be Real fix t assume [ty]: t be Real
  assume A1: 0_M < t   hence B1: t <> 0_M ...
  fix f assume [ty]: f be PartFunc of ℝ,ℝ
  have [ty]: f be Relation ...
  assume (⟦x, x + t⟧ ⊆ dom f ∧ f|⟦x,x+t⟧ be continuous) ∧
    f is differentiable on ⦇x,x + t⦈
  then obtain x0 where [ty]: x0 be Real and
  A2: x0 ∈ ⦇x,x + t⦈ and
  A3: diff(f,x0) = (f.(x+t) - f .x)/(x+t-x) ...
  obtain s where [ty]: s be set and sDef: s = (x0-x)/t ...
  have [ty]: s is Real ...
  show ∃s:Real. 0_M < s ∧ (s < 1_M∧
    f.(x+t) = f.x + t ∗ diff(f,x+s∗t))
  proof(rule bexI[of _ s],rule conjMI,rule conjMI)
    have x0 ∈ {r where r be Real : x < r ∧ r < x + t} ...
    hence
    A4: ∃g:Real. (g = x0 ∧ x < g) ∧ g < x + t ...
    hence 0_M < x0 - x ...
    hence 0_M / t < (x0 - x)/t ...
    thus 0_M < s ...
    have x0 - x < t ...
    hence (x0 - x)/ t < t / t ...
    thus s < 1_M ...
    have A5: s ∗ t + x = x0 - x + x ...
    have f.x + t ∗ diff(f,x0) = f.x + (f.(x+t) - f.x) ...
    thus f.(x + t) = f.x + t ∗ diff(f,x+s∗t) ...
  qed
qed
```

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.

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].

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: $\mathrm{t} <> \mathbf{0}_{\mathrm{M}}$ 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. Future work includes extending the translation to Mizar structures and proof schemes which would allow applying 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. The translated formalization is available at:

`http://cl-informatik.uibk.ac.at/cek/itp19mml200/`

References

1. Mark Adams. Proof auditing formalised mathematics. *J. Formalized Reasoning*, 9(1):3–32, 2016. URL: `https://jfr.unibo.it/article/view/4576`.
2. Grzegorz Bancerek, Czesław Byliński, Adam Grabowski, Artur Korniłowicz, Roman Matuszewski, Adam Naumowicz, and Karol Pąk. The role of the Mizar Mathematical Library for interactive proof development in Mizar. *Journal of Automated Reasoning*, 2017.
3. Pierre Corbineau. A declarative language for the Coq proof assistant. In Marino Miculan, Ivan Scagnetto, and Furio Honsell, editors, *Types for Proofs and Programs, TYPES 2007*, volume 4941 of *LNCS*, pages 69–84. Springer, 2007.
4. Leonardo Mendonça de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. The Lean theorem prover (system description). In Amy P. Felty and Aart Middeldorp, editors, *Conference on Automated Deduction, CADE 2015*, volume 9195 of *LNCS*, pages 378–388. Springer, 2015. `doi:10.1007/978-3-319-21401-6_26`.
5. Adam Grabowski, Artur Korniłowicz, and Adam Naumowicz. Four decades of Mizar. *Journal of Automated Reasoning*, 55(3):191–198, 2015. `doi:10.1007/s10817-015-9345-1`.
6. John Harrison. A Mizar mode for HOL. In Joakim von Wright, Jim Grundy, and John Harrison, editors, *Theorem Proving in Higher Order Logics: TPHOLs 1996*, volume 1125 of *LNCS*, pages 203–220. Springer, 1996. `doi:10.1007/BFb0105406`.
7. Cezary Kaliszyk and Karol Pąk. Isabelle import infrastructure for the Mizar mathematical library. In Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef, editors, *Intelligent Computer Mathematics (CICM 2018)*, volume 11006 of *LNCS*, pages 131–146. Springer, 2018. `doi:10.1007/978-3-319-96812-4_13`.
8. Cezary Kaliszyk and Karol Pąk. Semantics of Mizar as an Isabelle object logic. *Journal of Automated Reasoning*, 2018. `doi:10.1007/s10817-018-9479-z`.
9. Markus Wenzel and Freek Wiedijk. A comparison of Mizar and Isar. *J. Autom. Reasoning*, 29(3-4):389–411, 2002. `doi:10.1023/A:1021935419355`.
10. Freek Wiedijk. Mizar Light for HOL Light. In Richard J. Boulton and Paul B. Jackson, editors, *Theorem Proving in Higher Order Logics, TPHOLs 2001*, volume 2152 of *LNCS*, pages 378–394. Springer, 2001. `doi:10.1007/3-540-44755-5_26`.
11. Freek Wiedijk. A synthesis of the procedural and declarative styles of interactive theorem proving. *Logical Methods in Computer Science*, 8(1), 2012.