Computer-Assisted Mathematics at Work

The Hahn-Banach Theorem in Isabelle/Isar

Gertrud Bauer and Markus Wenzel

Technische Universität München
Institut für Informatik, Arcisstraße 21, 80290 München, Germany
http://www.in.tum.de/~bauerg/
http://www.in.tum.de/~wenzelm/

Abstract. We present a complete formalization of the Hahn-Banach theorem in the simply-typed set-theory of Isabelle/HOL, such that both the modeling of the underlying mathematical notions and the full proofs are intelligible to human readers. This is achieved by means of the Isar environment, which provides a framework for high-level reasoning based on natural deduction. The final result is presented as a readable formal proof document, following usual presentations in mathematical textbooks quite closely. Our case study demonstrates that Isabelle/Isar is capable to support this kind of application of formal logic very well, while being open for an even larger scope.

1 Introduction

The general idea of formalizing mathematics has already a long tradition. The desire to capture the way of human reasoning can be traced back far into the past, just consider Leibniz's *calculemus* manifest as a classic example. Purely syntactic formulation of mathematics with mechanical checking of proofs has finally matured during the 20th century. Roughly speaking, in its first half it has been demonstrated that mathematics could *in principle* be completely reduced to very basic logical principles. In the second half of the century the advent of computers enabled logicians to build systems for *actually doing* non-trivial applications in a fully formal setting. Over the last decades, many successful mechanized *proof checkers* and *proof assistants* have emerged, just consider de Bruijn's pioneering AUTOMATH project [21], or major contemporary theorem proving environments like Coq [12], Isabelle [25], and HOL [15].

This line of development represents tools for *actual verification*, in the sense that a very high level of confidence in correctness of the results is achieved. There is a wider picture of formal tools, though, including the important markets of *symbolic computation* (Computer Algebra) and *falsification aids*. The latter provide systematic ways to exhibit errors and counterexamples, rather than prove correctness. This is mainly the area of Model Checking, but general purpose theorem provers such as PVS [23] are usually positioned here as well.

Getting back to actual verification, we observe that current tactical provers (e.g. Isabelle [25] or Coq [12]) are usually quite inaccessible to non-specialist

users. This issue has been addressed in several ways, e.g. by providing graphical user interfaces to help users putting together tactic scripts (e.g. [1, 2]). Another major approach is to relate representations of formal proof objects directly with natural language, e.g. narrating λ-terms in plain English (or even French) [13]. There is also a more general *grammatical framework* based on type theory to support multi-lingual formal documents [17]. These efforts would ultimately result in a complete mathematical vernacular based on natural language (e.g. [9]).

Mizar [26, 29, 19, 35] has pioneered a rather different approach, by providing a higher-level proof language as its input format in the first place — avoiding the kind of machine-oriented transformations of tactical proving, which have so little in common with expressing mathematical ideas. While Mizar proved very successful for doing mainstream mathematics [18], it also has some fundamental limitations. The Mizar environment — the theory and proof language, together with its notion of "obvious inferences" — has been particularly tailored for applications within a first-order formulation of typed set-theory (Tarski-Grothendieck). It is unclear how to to change the logical basis, or even just basic proof tools. Learning how to use Mizar is difficult, because of its batch-mode nature and several complications due to first-order logic. Also note that Mizar does not claim the same level of formal correctness, as established by major proof checkers, such as Coq or Isabelle. It could be still possible to give fully formal foundations for Mizar in principle.

DECLARE [27, 28] is another more recent development of combining Mizar concepts and tactical proving into a "declarative" theorem proving system, suited for non-trivial meta-theoretical studies such as operational semantics.

Our present work employs the Isabelle/Isar system [32] as an environment for computer-assisted formal mathematics. Isar (which stands for *Intelligible semi-automated reasoning*) offers a generic approach to high-level natural deduction [31]. From the user's point of view, *formal proof documents* are the most fundamental concept of Isar. Following the basic structure of mathematical textbooks, iterating definition — theorem — proof, the actual text is written in a formal language with semantics firmly based on logic.

Isar provides a fresh start of the general idea of Mizar, while avoiding its shortcomings. The Isar framework is based on a few basic logical principles only, with the actual object-logic being left open as a parameter. Thus we gain logical flexibility, while also supporting the case of fully formal machine-checked proof with high confidence in the results as actual theorems. The basic mechanism of Isar proof checking does not depend on automated reasoning, nevertheless existing proof tools may be plugged in easily.

Interactive proof development, with incremental interpretation of Isar proof text, is considered an important issue. The Isabelle/Isar implementation [32] supports a simple model of *live document editing* that requires very basic user interface support only. Together with the existing Proof General interface [1], we already obtain a reasonable working environment for actual applications.

We have chosen the Hahn-Banach Theorem [16, 20] as a realistic case study of computer-assisted mathematics performed in Isabelle/Isar. The theorem has

been completely formalized (in two versions), together with any required notions of functional analysis, using Isabelle/HOL set-theory as logical basis [6, 5]. This particular example shall serve as a basis for a general assessment of the requirements of large-scale formalized mathematics.

Why does intelligible reasoning matter anyway? It is certainly fun to see computer-assisted mathematics actually work in non-trivial applications, and show the results to other people. Further, being able to communicate machine-checkable formal concepts adequately has an important cultural value [3], influencing the way that formal logic is perceived as an issue of practical relevance in general. The particular case of formal proof in education has been addressed many times before (e.g. [8]). We even raise the general philosophical principle that any important (or even critical) piece of formal code (proofs or programs) should be in itself open for human understanding. Informal explanations (e.g. comments) and mechanic analysis (e.g. independent proof checking) play an important role, but also have their limitations (e.g. comments could be misleading or even inconsistent with the formal code). Having an adequate language of formal discourse available, we are enabled to communicate our reasoning directly in a format that may be machine-checked later. Thus we achieve a "second source" for correctness: inspecting the formal source we (hopefully) get convinced of its plausibility, while knowing that it has passed a trusted proof checker as well.

The rest of this paper is structured as follows. Section 2 explains basic issues of formal proof in Isabelle/Isar by giving some examples. Section 3 briefly reviews central aspects of Isar as a working environment for formalized mathematics. Section 4 discusses a fully formal treatment of the Hahn-Banach Theorem as a realistic example of mainstream mathematics in Isabelle/Isar.

2 Basic Examples

In order to get a first idea how computer-assisted mathematics may look like in Isabelle/Isar, we consider basic group theory as a small "practical" example. We introduce the abstract structure of general groups over some carrier type α, together with product $\_ \cdot \_$ and inverse $\_^{-1}$ operations, unit element 1, and axioms stating associativity, and the left inverse and unit properties. As usual, the right inverse and unit laws may be derived as theorems of group theory.

Below, we start a new theory context *Group* derived from the plain *HOL* basis. Then we introduce constant declarations with Isabelle-style mixfix annotations for concrete syntax. The structure of general groups over some carrier type is defined by employing Isabelle's *Axiomatic Type Classes* [30, 34], which provide a useful mechanism for abstract algebraic concepts. Finally we establish the two basic consequences of the group axioms as formally proven theorems.

```
theory Group = HOL:

consts
  prod :: "'a → 'a → 'a"        (infixl "·" 70)
```

```
  inv  :: "'a ⇒ 'a"              ("(_⁻¹)" [1000] 999)
  unit :: "'a"                   ("1")

axclass group < "term"
  assoc:        "(x · y) · z = x · (y · z)"
  left_inv:     "x⁻¹ · x = 1"
  left_unit:    "1 · x = x"

theorem right_inv: "x · x⁻¹ = (1::'a::group)" ⟨proof⟩

theorem right_unit: "x · 1 = (x::'a::group)"
proof -
  have "x · 1 = x · (x⁻¹ · x)" by (simp only: left_inv)
  also have "... = (x · x⁻¹) · x" by (simp only: assoc)
  also have "... = 1 · x" by (simp only: right_inv)
  also have "... = x" by (simp only: left_unit)
  finally show ?thesis .
qed

end
```

This text directly represents the input format of Isabelle/Isar, apart from some simple pretty printing applied in the presentation. Using the Proof General interface [1] one may even achieve a similar display on screen. Our definition of abstract groups uses **axclass** (see [34] for more details). Both of the proofs above are conducted by *calculational reasoning*, the first one has been suppressed in the presentation, though.

As is typical for forward-reasoning, the initial proof step does not apply any reduction yet, which is indicated by "**proof** -". The proof body establishes the main thesis by a sequence of intermediate results (**have** proven via a single step of **by** each[1]) that are composed by transitivity. The "..." notation refers to the most recent right-hand side expression. The **also** element causes the current calculation to be combined with the latest fact. So does **finally**, but it also concludes the calculation by offering the final result to the next statement.

Isar calculations are more general than shown here. Calculational elements may be even combined with plain natural deduction (e.g. [33, §6]), without having to subscribe to a fully calculational view of logic in general [14].

In the next example we review slightly more involved logical reasoning: Smullyan's *Drinkers' principle* (e.g. [3]) is a puzzle of pure classical logic. It states that there is some individual such that whenever he is getting drunk, everybody else does so as well ("drunk" may be replaced by any predicate).

Theorem (Drinkers' Principle). $\exists x.\ Q\ x \to (\forall y.\ Q\ y)$

Proof. We show $Q\ x \to (\forall y.\ Q\ y)$ for some x by case analysis. Assume $\forall y.\ Q\ y$, then any individual makes the implication true. Assume $\neg\ (\forall y.\ Q\ y)$, then there is an y such that $\neg\ Q\ y$ holds, which makes the implication true as well.

[1] Isabelle's simplifier is used here to normalize with a single equation only.

This narration of usual informal mathematics style is turned into a formal Isabelle/Isar text as follows, while retaining the overall structure of reasoning.

```
theorem Drinkers'_Principle: "∃x. Q x ⟶ (∀y. Q y)"
proof cases
  assume "∀y. Q y"
  fix any have "Q any ⟶ (∀y. Q y)" ..
  thus ?thesis ..
next
  assume "¬(∀y. Q y)"
  then obtain y where "¬ Q y" ⟨proof⟩
  hence "Q y ⟶ (∀y. Q y)" ..
  thus ?thesis ..
qed
```

Isar certainly does require some understanding of the language semantics [31, 32] in order to appreciate the formal reasoning in detail. Subsequently, we shall point out the most important aspects of this proof.

The outermost "**proof** *cases*" step refers to the propositional case-split rule $(A \Rightarrow C) \Rightarrow (\neg\ A \Rightarrow C) \Rightarrow C$, thus the body gets divided into two branches, which are separated by **next**. The rule admits to introduce an additional local hypotheses using **assume** in each case. In order to establish the main thesis, an existential statement, we prove the goal for some suitable witness. In the first case, **fix** augments the context by a new local variable (without any additional assumptions), and **have** states the desired implication. The double-dot proof "**..**" means that the result is established from the current context by a single standard structural rule (here $\longrightarrow$-intro). With this result, the thesis is just another single step away (apparently via $\exists$-intro).

Note that idiomatic phrases such as "**thus** ?*thesis* **..**" are quite typical for Isar. We have seen a similar one in the group calculation: "**finally show** ?*thesis* **.**". Isar avoids specialized language elements as much as possible, reducing anything to few principles only. The flexible way that the basic entities may be composed into well-formed proof texts results in a very rich language.

The second case of our proof is similar to the first one, but the witness element is produced differently: the assumption $\neg\ (\forall y.\ Q\ \ y)$ classically yields $\exists y.\ \neg\ Q\ y$, so we may pick any such y when showing the main goal (by virtue of the $\exists$-elim rule). The derived Isar language element **obtain** arranges this kind of formal reasoning in a way that is close to usual mathematical practice. In particular, the existential statement and the actual elimination step are put out of the main focus, highlighting the resulting context modification instead. Above we have even suppressed an actual proof, leaving a place holder. Completing this in terms of basic logical reasoning would be just an exercise on de Morgan's Law, turning $\neg\ (\forall y.\ Q\ \ y)$ into $\exists y.\ \neg\ Q\ \ y$.

Alternatively, the proof for **obtain** may be finished with some automated proof tool, say "**by** *blast*", which refers to Isabelle's tableau prover. How to proceed in such situations is mainly a question of methodology. It is up to the

author to determine which parts of the proof are considered relevant for the intended audience, while the proof language has to offer the structural flexibility.

3 An Environment for Intelligible Formal Mathematics

We discuss some central aspects of the Isar proof language, which is at the heart of our environment for intelligible formal mathematics. Three stages are considered: a minimal logical framework for *primitive natural deduction*, the Isar *primary proof language* seen as a logically redundant enrichment, and *derived proof schemes* for advanced applications. The resulting architecture fully preserves machine-checkable correctness as provided by the primitive level.

3.1 Logical Foundations

We closely follow Isabelle's meta-logic [24], which is an intuitionistic $\forall/\Rightarrow/\equiv$-fragment of higher-order logic. Logical syntax is that of simply-typed λ-calculus. Proof rules are the standard ones for minimal logic, with definitional equality $\equiv$. Proof objects may be represented within a typed λ-calculus with separate abstraction and application for simply-typed terms $x : \tau$ and propositions $a : A$.[2] The set H of propositions in *Hereditary Harrop Form* (HHF) is defined inductively as $H = \forall \overline{x}.\, \overline{H} \Rightarrow A$, where x refers to the set of variables, A to atomic propositions, and $\overline{x}$, $\overline{H}$ to lists. HHF formulae play a central role in representing both natural deduction rules and internal proof states [24, 25]. Note that according to HHF, contexts have the canonical form $\Gamma = \overline{x}, \overline{H}$.

Common object-logics based on natural deduction (e.g. classical HOL, ZF set-theory, even type theory) can be expressed within this meta-logic in a convenient way [25]. Any such formalization may be directly re-used within the Isar framework, including theory libraries, definitional packages and proof tools [32].

3.2 Basic Proof Language

The Isar core proof language provides 12 primitive elements, which are interpreted on top of the basic logical framework by referring to its primitive inferences (mostly derived rules for back-chaining and proof-by-assumption), together with some additional book-keeping [31]. The Isar primitives are as follows [32, Appendix A]: "**fix** $x : \tau$" and "**assume** $a : A$" augment the context, **then** indicates forward chaining (e.g. to do elimination in the subsequent reduction step), "**have** $a : A$" and "**show** $a : A$" claim local statements (the latter includes solving of some pending goal afterwards), "**proof** m" performs an initial proof step by applying some method, "**qed** m" concludes a (sub-)proof, **{ }** and **next** manage block structure, "**note** $a = \overline{b}$" reconsiders facts, and "**let** $p = t$" abbreviates terms via higher-order matching against some pattern.

[2] Note that τ is usually suppressed due to type-inference, while a is omitted internally in implementations following the "LCF-approach".

In addition, there are 7 basic defined elements: "**by** m_1 m_2" for proofs with an empty body, "**..**" for single-rule proofs, "**.**" for immediate proofs, **hence**/**thus** for claims with forward chaining indicated, and "**from** $\overline{a}$"/"**with** $\overline{a}$" for explicit forward chaining from (additional) facts.

The most essential ingredient to achieve a well-balanced basis for intelligible proof is free choice over forward vs. backward reasoning. To see how this works out in Isar, we relate chunks of proof text to an enriched version of primitive proof objects (cf. §3.1). We define backward ($\triangleright$) and forward ($\triangleleft$) application operators for lists of λ-terms such that $\overline{s} \triangleright t \triangleleft \overline{u} \equiv t\ \overline{s}\ \overline{u}$. Now let $\mathcal{R}$ be a natural deduction rule $\overline{a : A} \Rightarrow \overline{b : B} \Rightarrow C$, where the premises are separated into two lists $\overline{a : A}$ and $\overline{b : B}$ (for clarity we suppress any contexts of the premises).

The body of Isar (sub)-proofs consists of a sequence of context elements or proven statements. These may be put into the following standard form:

$\langle \mathit{context} \rangle$ **from** $\overline{a}$ **show** C **proof** (rule $\mathcal{R}$) $\langle \mathit{body} \rangle$ **qed**

where $\langle \mathit{body} \rangle$ is recursively of the same structure. This is a correct piece of reasoning, if $\langle \mathit{body} \rangle$ proves "**show** B_i" for each B_i in $\overline{B}$, such that $\overline{a} \triangleright \mathcal{R} \triangleleft \pi(\overline{B})$ establishes C, for some permutation π.

The impact on the overall structure of Isar proofs is as follows. The sub-problems stemming from rule $\mathcal{R}$ are split into parts $\overline{a}$ and $\pi(\overline{B})$, where the $\overline{a : A}$ have been established beforehand and $\overline{b : B}$ are deferred to sub-proofs at a deeper level. It is important to note that the two sections are *not* handled symmetrically: $\overline{a}$ refers to facts from the context by *symbolic names* and in a *fixed order*, while in the body sub-problems are stated as *explicit propositions* in an *arbitrary order* $\pi(\overline{B})$. This enables readable proofs, since the $\overline{A}$ statements can be easily spotted verbatim in the preceding context, while the members of $\overline{B}$ appear in the body below, in an appropriate order to handle the more interesting ones first. Note that the fixed order of the $\overline{a}$ specification still admits the corresponding statements to appear anywhere in the context. On the other hand, this policy improves clarity and robustness of proof checking, since it makes it easy to determine rule $\mathcal{R}$ automatically from the structure of $\overline{A}$ and C (for elimination or introduction, respectively) without any serious search. Consequently, Isar proofs seldom name $\mathcal{R}$ explicitly, but usually decompose according to implicit standard rules.

3.3 Derived Proof Schemes

Large case studies such as the Hahn-Banach Theorem show that realistic mathematical applications demand additional proof support, apart from the pure natural deduction provided so far (cf. §3.2). On the other hand, the basic Isar proof language turns out to be sufficiently expressive to admit advanced schemes as further derived elements. Subsequently, we discuss a flexible form of calculational proof, and generalized reasoning with eliminated existence.

Calculational Proof can be understood as iterated reasoning with transitivity rules, such that the final result emerges from folding a sequence of facts together. This may involve any suitable "binary" rule, like $s = t \Rightarrow t = u \Rightarrow s = u$, the same for $<$ and $\leq$, including any combination of these. Substitution $s = t \Rightarrow P\ s \Rightarrow P\ t$ works as well, then composition means to replace equal sub-terms.

Isar calculations work incrementally, maintaining a secondary result called *calculation* by composition with the primary one *this*, which always refers to the latest fact. We now just define two new language elements, **also** and **finally**.

$$\begin{aligned} \textbf{also}_0 &\equiv \textbf{note}\ \mathit{calculation} = \mathit{this} \\ \textbf{also}_{n+1} &\equiv \textbf{note}\ \mathit{calculation} = \mathit{trans}\ [\mathit{OF}\ \ \mathit{calculation}\ \ \mathit{this}] \\ \textbf{finally} &\equiv \textbf{also from}\ \mathit{calculation} \end{aligned}$$

Here $\textbf{also}_0$ refers to the first, and $\textbf{also}_{n+1}$ to further occurrences of **also** within a calculational sequence, at the *same* level of blocks. The *OF* operation combines logical rules using higher-order resolution (back-chaining). For atomic propositions, *OF* indeed coincides with application in λ-calculus. The *trans* rule above is determined by higher-order unification from a set of transitivities declared in the theory library. These rules usually include plain transitivity of $=/</\leq$, and substitution of $=$, or even $</\leq$ with monotonicity conditions extracted in the expected way. Determining rules implicitly by higher-order unification works very well in practice, without any serious search required.

Another version of calculational elements are **moreover** and **ultimately**, which are even more simple since they only collect facts without applying any rules yet. This is quite useful to accumulate a number of intermediate results that contribute to some ultimate result. Thus the proof text is often easier to read as we avoid explicit naming of intermediate facts.

$$\begin{aligned} \textbf{moreover} &\equiv \textbf{note}\ \mathit{calculation} = \mathit{calculation}\ \ \mathit{this} \\ \textbf{ultimately} &\equiv \textbf{moreover from}\ \mathit{calculation} \end{aligned}$$

One may also use **also** and **moreover** together within the same calculation, say if using rules that require more than two facts to yield the intended result.

Eliminated Existence Reasoning means that additional variables with certain hypotheses are introduced, as justified by a corresponding soundness proof. Consider the special case of eliminating $\exists x.\ H[x]$, where an additional x with assumption $H[x]$ may be obtained. The derived Isar element **obtain** is defined as follows (optional $\langle \mathit{facts} \rangle$ may be have been indicated for forward chaining).

$\langle \mathit{facts} \rangle$ **obtain** $\overline{x}$ **where** $\overline{H[\overline{x}]}$ $\langle \mathit{proof} \rangle \equiv$
{
  fix C
  assume $\forall \overline{x}.\ \overline{H[\overline{x}]} \Rightarrow C$
  from $\langle \mathit{facts} \rangle$ **have** C $\langle \mathit{proof} \rangle$
}
fix $\overline{x}$ **assume*** $\overline{H[\overline{x}]}$

After having finished the soundness proof, the assumptions $\overline{H[\overline{x}]}$ are introduced with an internal hint of being *obtained*. This tells the Isar interpreter how to discharge this context element later, whenever a result is exported from its scope. According to the nature of existential reasoning, parameters $\overline{x}$ may never occur in a final conclusion, only in intermediate results within the same context.

The **obtain** scheme has many virtues in reducing the complexity of formal proof texts. For example, duplicate occurrences of $\overline{x}$ and $\overline{H}$ in the text are avoided. Furthermore, the soundness proof of **obtain** is usually straightforward by using existing facts together with basic automated tools (e.g. rewriting). Speaking in terms of first-order logic, the proof would basically correspond to iterated introduction of $\exists$ and $\wedge$, but **obtain** in Isar does not even mention any particular $\exists$ or $\wedge$ connective of the object-logic. This way we gain both flexibility and avoid cumbersome automated reasoning with existential quantifiers.

Between the two extremes of basic **assume** and **obtain** there are further derived context elements in Isar: "**def** $x \equiv t$" is like "**fix** x **assume** $x \equiv t$" where the equation is discharged by generalization and reflexivity later, while **presume** is just like **assume**, but leaves the assumption as a new subgoal.

3.4 Addressing Correctness

In order to see how Isar fares in the quest of correctness [3], recall its basic arrangement of formal concepts: there are two main levels, the *primitive* logical core and the *primary* Isar proof language; these are related by an interpretation function, providing an operational semantics of Isar proof texts in terms of primitive inferences [31]. First of all, suppose we believe in the basic logical framework (see §3.1), and know how to implement it at the highest conceivable level of correctness (cf. the discussion in [3, 4]). Furthermore, we may formulate correctness (or even completeness) results of Isar proofs related to primitive ones by virtue of the operational semantics. While this would tell us that the Isar machine operates adequately, without producing nonsense or failing unexpectedly, it is *not* the primary means to achieve ultimate machine-checked correctness: both the Isar interpreter program and its correctness proof are sufficiently complex to lower the resulting level of confidence at least by an order of magnitude.

Fortunately we can do better, even with informal proof sketches of the Isar machine correctness result and an unverified implementation only. The key property of the Isar interpretation process is that actual "theorems" can be treated as *non-observable objects* that are manipulated abstractly, without ever depending on the actual structure of internal proofs or propositions. Thus the original notion of correctness of the primitive level is passed undisturbed to the primary one of Isar proof text processing.

4 The Hahn-Banach Theorem

The Hahn-Banach Theorem is probably the most fundamental result in functional analysis (e.g. [20]). We will consider an informal proof in a standard

mathematical textbook [16, §36] where two different versions of the theorem are presented, one for general linear spaces, and one for normed vector spaces.

We show how the underlying mathematical notions can be expressed in a very natural way, employing the simply-typed set theory of HOL [11, 15]. We also present a proof in Isabelle/Isar, which closely follows the original one [16].

4.1 Structure of the Proof

Theorem (Hahn-Banach). Let F be a subspace of a real vector space E, let p be a semi-norm on E, and f be a linear form defined on F such that f is bounded by p, i.e. $\forall x \in F.\ f\ x \leq p\ x$. Then f can be extended to a linear form h on E such that h is norm-preserving, i.e. h is also bounded by p on E.

Proof Sketch.

1. Define M as the set of norm-preserving extensions of f to subspaces of E. The linear forms in M are ordered by domain extension.
2. We show that every non-empty chain in M has an upper bound in M.
3. With Zorn's Lemma we conclude that there is a maximal function g in M.
4. The domain H of g is the whole space E, as shown by classical contradiction:
   - Assuming g is not defined on whole E, it can still be extended in a norm-preserving way to a super-space H' of H.
   - Thus g can not be maximal. Contradiction!

From this we also get a version of the Hahn-Banach theorem for normed spaces [16, §36]. The complete formal proof of this corollary is given in [6, 5].

4.2 Formalization in HOL Set Theory

We formalize basic notions of functional analysis in HOL set-theory: vector spaces, subspaces, and an order of functions by domain extension. Further notions such as normed vector spaces, continuous linear forms and norms of functions are required for the version for normed vector spaces only, see [6, 5].

Note that our development does not require any topological notions. The interpretation of bounded linear forms as being "continuous" is left informal. In fact, this treatment follows the usual practice in functional analysis [16].

Vector Spaces. There are several ways of defining abstract mathematical structures such as vector spaces in HOL. One is to define axiomatic type classes (cf. the group example in §2). Another general principle is to define structures as predicates over a carrier set together with operations. We apply a particular instance of this principle where we use polymorphic operations $+$, $-$ and 0 on a generic type α. Further, we introduce an operation $\cdot :: \mathbb{R} \to \alpha \to \alpha$ and define

$$\begin{aligned} &\mathit{is\text{-}vectorspace} :: \alpha\ \mathit{set} \to \mathit{bool} \\ &\mathit{is\text{-}vectorspace}\ V \equiv V \neq \{\}\ \wedge\ (\forall x \in V.\, \forall y \in V.\, \forall z \in V.\, \forall a\ b. \\ &\qquad x + y \in V\ \wedge\ a \cdot x \in V\ \wedge\ (x + y) + z = x + (y + z)\ \wedge\ \ldots) \end{aligned}$$

Alternatively, we could have defined a class of records with components for the carrier set and corresponding operations. While this closely reflects common treatment of mathematical structures *in theory*, it deviates from the usual practice, since we would have to refer to explicit record selector and update operations all the time. Our present approach has the advantage of mimicking informal mathematical usage, by identifying a structure with its carrier.

Subspaces. The way that vector spaces have been modeled above enables subspaces to be described succinctly: just use HOL's $\subseteq$ relation on the carriers and express closure wrt. vector space operations as usual [6]. Furthermore, the main proof will construct chains of vector spaces, with the supremum simply as $\bigcup$.

Observing these abstract virtues, we shall also validate our notion of subspaces in concrete instances. For example, using type $\mathbb{N} \to \mathbb{R}$ for α we would first define $+$, $-$, $\cdot$, 0 point-wise on the whole domain. Then any n-dimensional space $\mathbb{R}^n$ would correspond to $\{f.\ \forall i \geq n.\ f_i = 0\}$. Apparently, any such carrier set is closed under vector operations — it does not matter that these are defined on the whole type. Common infinitary vector spaces can be defined as well: l^∞ becomes $\{f.\ \exists c.\ \forall i.\ |f_i| < c\}$, and l^p becomes $\{f.\ \exists c.\ \forall k.\ \sum_{i<k} |f_i|^p < c\}$. Using $\mathbb{R} \to \mathbb{R}$ for α we could even define the very rich class of L^p spaces, provided we also have a sufficient base of real analysis and measure theory in HOL.

Partial functions ordered by domain extension. Expressing partial functions in an inherently total setting like HOL requires some care. A standard technique that always works is to consider the graph of a function. This turns out to be perfectly adequate for our application. We define graphs as follows:

$$\begin{aligned}
&\alpha\ \mathit{graph} = (\alpha \times \mathbb{R})\ \mathit{set} \\
&\mathit{graph} :: \alpha\ \mathit{set} \to (\alpha \to \mathbb{R}) \to \alpha\ \mathit{graph} \\
&\mathit{graph}\ F\ f \equiv \{(x, f\ x).\ \ x \in F\}
\end{aligned}$$

When speaking informally we never distinguish a function from its graph. With the above definition we can now introduce the order on functions by extension very easily: h is an extension of f iff $\mathit{graph}\ F\ f \subseteq \mathit{graph}\ H\ h$.

For the proof of the Hahn-Banach theorem we need the set of all norm-preserving extensions of a linear form f defined on a vector space F. This can be expressed in HOL in a very natural way. It is the set of all graphs of linear extensions of f, to super-spaces H of F, that are bounded by the semi-norm p:

$$\begin{aligned}
&\mathit{norm\text{-}pres\text{-}extensions} :: \alpha\ \mathit{set} \to (\alpha \to \mathbb{R}) \to \alpha\ \mathit{set} \to (\alpha \to \mathbb{R}) \to \alpha\ \mathit{graph}\ \mathit{set} \\
&\mathit{norm\text{-}pres\text{-}extensions}\ E\ p\ F\ f \equiv \{\mathit{graph}\ H\ h.\ \ \mathit{is\text{-}linearform}\ H\ h \\
&\qquad \wedge\ \mathit{is\text{-}subspace}\ H\ E\ \wedge\ \mathit{is\text{-}subspace}\ F\ H \\
&\qquad \wedge\ \mathit{graph}\ F\ f \subseteq \mathit{graph}\ H\ h\ \wedge\ (\forall x \in H.\ h\ x \leq p\ x)\}
\end{aligned}$$

The canonical order by inclusion on this set of graphs corresponds to the order of functions by domain extension.

Zorn's Lemma. We follow the informal proof of the Hahn-Banach Theorem [16] in using Zorn's Lemma, which can be actually proved in Isabelle/HOL using Hilbert's ε choice operator. The following formulation will be used: "Let M be a non-empty ordered set; if any non-empty chain c in M has an upper bound in M, then M has a maximal element, i.e. $\exists g \in M.\ \forall x \in M.\ g \leq x \Rightarrow g = x$."

4.3 The Main Proof in Isabelle/Isar

We present an abstracted version of the actual formal proof in Isabelle/Isar [6]. The structure of the text follows that of the sketch given in §4.1. Readers familiar with the Isar semantics should be able to follow the reasoning mostly from the formal text only. We have augmented the text by comments giving the corresponding informal reading of each significant step as well.

theory *HahnBanach = HahnBanachLemmas:*

theorem *HahnBanach:*
"is_vectorspace E ==> is_subspace F E ==> is_seminorm E p ==>
is_linearform F f ==> ∀x∈F. f x <= p x ==>
∃h. is_linearform E h ∧ (∀x∈F. h x = f x)
∧ (∀x∈E. h x <= p x)"
— Let E be a vector space, F a subspace of E, p a seminorm on E,
— and f a linear form on F such that f is bounded by p,
— then f can be extended to a linear form h on E in a norm-preserving way.

proof -
assume *"is_vectorspace E" "is_subspace F E" "is_seminorm E p"*
and *"is_linearform F f" "∀x∈F. f x <= p x"*
— Assume the context of the theorem.

def *M == "norm_pres_extensions E p F f"*
— Define M as the set of all norm-preserving extensions of F.

{
fix *c* **assume** *"c∈chain M" "∃x. x∈c"*
have *"⋃c∈M"* ⟨*proof*⟩
— Show that every non-empty chain c of M has an upper bound in M:
— $\bigcup c$ is greater than any element of the chain c, so it suffices to show $\bigcup c \in M$.
}
hence *"∃g∈M. ∀x∈M. g ⊆ x --> g = x"* ⟨*proof*⟩
— With Zorn's Lemma we can conclude that there is a maximal element in M.

thus *?thesis*
proof
fix *g* **assume** *"g∈M" "∀x∈M. g ⊆ x --> g = x"*
— We consider such a maximal element $g \in M$.

obtain *H h* **where** *"graph H h = g" "is_linearform H h"*
"is_subspace H E" "is_subspace F H" "graph F f ⊆ graph H h"
"∀x∈H. h x <= p x" ⟨*proof*⟩

— g is a norm-preserving extension of f, in other words:
— g is the graph of some linear form h defined on a subspace H of E,
— and h is an extension of f that is again bounded by p.

have *"H = E"*
— We show that h is defined on whole E by classical contradiction.

proof *(rule classical)*
assume *"H ≠ E"*
— Assume h is not defined on whole E. Then show that h can be extended
— in a norm-preserving way to a function h' with the graph g'.

have *"∃g'∈M. g ⊆ g' ∧ g ≠ g'"*
proof -
obtain *x'* **where** *"x'∈E" "x' ∉ H"* ⟨*proof*⟩
— Pick $x' \in E \setminus H$.

def *H' == "H + lin x'"*
— Define H' as the direct sum of H and the linear closure of x'.

obtain *xi* **where** *"∀y∈H. - p (y + x') - h y <= xi*
∧ xi <= p (y + x') - h y" ⟨*proof*⟩
— Pick a real number ξ that fulfills certain inequations; this will
— be used to establish that h' is a norm-preserving extension of h.

def *h' == "λx. let (y,a) = SOME (y,a). x = y + a · x' ∧ y∈H*
*in h y + a * xi"*
— Define the extension h' of h to H' using ξ.

show *?thesis*
proof
show *"g ⊆ graph H' h' ∧ g ≠ graph H' h'"* ⟨*proof*⟩
— Show that h' is an extension of h ...

show *"graph H' h'∈M"* ⟨*proof*⟩
— and h' is norm-preserving.

qed
qed
hence *"¬ (∀x∈M. g ⊆ x --> g = x)"* **by** *simp*
— So the graph g of h cannot be maximal. Contradiction!

thus *"H = E"* **by** *contradiction*
qed
thus *"∃h. is_linearform E h ∧ (∀x∈F. h x = f x)*
∧ (∀x∈E. h x <= p x)" ⟨*proof*⟩
qed
qed

end

For the above presentation we have pruned the actual Isabelle/Isar proof [6] at the outermost level, in order to focus on the main course of reasoning. Never-

theless, the resulting text still qualifies as well-formed Isar proof, provided that any omitted ⟨*proof*⟩ does. Note that Isar enjoys *compositional proof checking*.

The overall structure of the complete Isar proof of this representative example is as follows. The topmost proof outline (of a few pages) refers to a number of special lemmas via some "glue code", which is expressed using automated proof tools (e.g. the simplifier). The proofs of the lemmas typically require considerably more effort, mainly due to gory details usually skipped in informal presentations, such as [16]. Furthermore, any particular application usually requires some additional background theory of basic notions. The subsequent numbers give a rough comparison of three Hahn-Banach proofs, where Heuser's is with pen-and-paper.

	basic notions	special lemmas	main proof
Heuser [16]	?	–	3 pages
Mizar [22]	?	25 pages	8 pages
Isar [6]	35 pages	16 pages	5 pages

While the Mizar proof differs from ours in many details, both have a similar level of abstraction. Also note that the Mizar version refers to a large library of formalized mathematics that is hard to pin down exactly.

The complete Hahn-Banach development [6] as distributed with Isabelle99-1 takes 63 pages in total. It includes some additional explanations and an alternative version of the main theorem. The performance of Isabelle/Isar in processing this document is quite reasonable: proof checking plus LaTeX generation takes less than 3 minutes on a 300 MHz machine; real memory requirements are about 40 MB. These figures are typical for Isabelle/HOL in general, the overhead for Isar proof text processing compared to primitive tactic applications is very small.

5 Conclusion

We have evaluated the Isabelle/Isar environment by the case study of the Hahn-Banach Theorem, as a large example of formalized mathematics. Using simply-typed classical HOL set-theory, we have been able to model the underlying notions of functional analysis similar to the informal presentation in the textbook [16]. Furthermore, the high-level Isar proof language has enabled us to provide a machine-checked proof, with the reasoning arranged at differently conceptual levels, such that the topmost outline closely resembles an informal proof sketch.

The Hahn-Banach theorem already appears in informal mathematics in a multitude of formulations, and quite different approaches to its proof (cf. [20]). There are some machine-checked formalizations as well, notably a Mizar version [22] (which is based on Tarski-Grothendieck set-theory), and a formulation in Martin-Löf Type Theory [10] that has been checked with the Agda system. In contrast to the Mizar and Isar versions, which basically share the same presentation of Hahn-Banach in a classical setting of functional analysis (using Zorn's Lemma), the Martin-Löf one is presented quite differently within the setting of point-free formal topology [10].

Our case study on Hahn-Banach has shown that Isabelle/Isar is ready for complex mathematical applications. For future development, we aim at extending our scope towards further areas in computer science, such as meta-theoretical studies of programming languages (e.g. type systems and operational semantics). This will probably demand further derived elements on top of the basic Isar proof language, such as more compact representations of local contexts stemming from abstract algebraic structures or large case analysis rules. Furthermore, users would certainly appreciate further assistance in constructing Isar proof documents, such as systematic support for common proof patterns.

References

[1] D. Aspinall. Proof General: A generic tool for proof development. In *European Joint Conferences on Theory and Practice of Software (ETAPS)*, 2000.

[2] D. Aspinall. Protocols for interactive e-Proof. In *Theorem Proving in Higher Order Logics (TPHOLs)*, 2000. Unpublished work-in-progress paper, http://zermelo.dcs.ed.ac.uk/~da/drafts/eproof.ps.gz.

[3] H. P. Barendregt. The quest for correctness. In *Images of SMC Research*, pages 39–58. Stichting Mathematisch Centrum, Amsterdam, 1996.

[4] H. P. Barendregt, G. Barthe, and M. Ruys. A two level approach towards lean proof-checking. In S. Berardi and M. Coppo, editors, *Types for Proofs and Programs (TYPES'95)*, LNCS 1158, 1995.

[5] G. Bauer. Lesbare Formale Beweise in Isabelle/Isar — Der Satz von Hahn-Banach. Master's thesis, Technische Universität München, 1999.

[6] G. Bauer. *The Hahn-Banach Theorem for real vector spaces.* Technische Universität München, 2000. Isabelle/Isar proof document, http://isabelle.in.tum.de/library/HOL/HOL-Real/HahnBanach/document.pdf.

[7] Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Thery, editors. *Theorem Proving in Higher Order Logics: TPHOLs '99*, LNCS 1690, 1999.

[8] R. Burstall. Teaching people to write proofs: a tool. In *CafeOBJ Symposium, Numazu, Japan*, April 1998.

[9] P. Callaghan and Z. Luo. Mathematical vernacular in type theory-based proof assistants. Workshop on User Interfaces in Theorem Proving, Eindhoven, 1998.

[10] J. Cederquist, T. Coquand, and S. Negri. The Hahn-Banach theorem in type theory. In G. Sambin and J. Smith, editors, *Twenty-Five years of Constructive Type Theory.* Oxford University Press, 1998.

[11] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, pages 56–68, 1940.

[12] C. Cornes, J. Courant, J.-C. Filliâtre, G. Huet, P. Manoury, and C. Muñoz. *The Coq Proof Assistant User's Guide, version 6.1.* INRIA-Rocquencourt et CNRS-ENS Lyon, 1996.

[13] Y. Coscoy, G. Kahn, and L. Théry. Extracting text from proofs. In *Typed Lambda Calculus and Applications*, volume 902 of *LNCS*. Springer, 1995.

[14] E. W. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics.* Texts and monographs in computer science. Springer, 1990.

[15] M. J. C. Gordon and T. F. Melham (editors). *Introduction to HOL: A theorem proving environment for higher order logic.* Cambridge University Press, 1993.

[16] H. Heuser. *Funktionalanalysis: Theorie und Anwendung.* Teubner, 1986.

[17] P. Mäenpää and A. Ranta. The type theory and type checker of GF. In *Colloquium on Principles, Logics, and Implementations of High-Level Programming Languages*. Workshop on Logical Frameworks and Meta-languages, Paris, 1999.
[18] Mizar mathematical library. http://www.mizar.org/library/.
[19] M. Muzalewski. *An Outline of PC Mizar*. Fondation of Logic, Mathematics and Informatics — Mizar Users Group, 1993. http://www.cs.kun.nl/~freek/mizar/mizarmanual.ps.gz.
[20] L. Narici and E. Beckenstein. The Hahn-Banach Theorem: The life and times. In *Topology Atlas*. York University, Toronto, Ontario, Canada, 1996. http://at.yorku.ca/topology/preprint.htm and http://at.yorku.ca/p/a/a/a/16.htm.
[21] R. P. Nederpelt, J. H. Geuvers, and R. C. de Vrijer, editors. *Selected Papers on Automath*, Studies in Logic 133. North Holland, 1994.
[22] B. Nowak and A. Trybulec. Hahn-Banach Theorem. *Journal of Formalized Mathematics*, 5, 1993. http://mizar.uwb.edu.pl/JFM/Vol5/hahnban.html.
[23] S. Owre, S. Rajan, J. M. Rushby, N. Shankar, and M. Srivas. PVS: combining specification, proof checking, and model checking. In R. Alur and T. A. Henzinger, editors, *Computer Aided Verification*, LNCS 1102, 1996.
[24] L. C. Paulson. Isabelle: the next 700 theorem provers. In P. Odifreddi, editor, *Logic and Computer Science*. Academic Press, 1990.
[25] L. C. Paulson. *Isabelle: A Generic Theorem Prover*. LNCS 828. Springer, 1994.
[26] P. Rudnicki. An overview of the MIZAR project. In *1992 Workshop on Types for Proofs and Programs*. Chalmers University of Technology, Bastad, 1992.
[27] D. Syme. *Declarative Theorem Proving for Operational Semantics*. PhD thesis, University of Cambridge, 1998.
[28] D. Syme. Three tactic theorem proving. In Bertot et al. [7].
[29] A. Trybulec. Some features of the Mizar language. Presented at a workshop in Turin, Italy, 1993.
[30] M. Wenzel. Type classes and overloading in higher-order logic. In *10th International Conference on Theorem Proving in Higher Order Logics, TPHOLs'97*, LNCS 1275, 1997.
[31] M. Wenzel. Isar — a generic interpretative approach to readable formal proof documents. In Bertot et al. [7].
[32] M. Wenzel. *The Isabelle/Isar Reference manual*. Technische Universität München, 2000. Part of the Isabelle documentation, http://isabelle.in.tum.de/doc/isar-ref.pdf.
[33] M. Wenzel. *Miscellaneous Isabelle/Isar examples for Higher-Order Logic*. Technische Universität München, 2000. Isabelle/Isar proof document, http://isabelle.in.tum.de/library/HOL/Isar_examples/document.pdf.
[34] M. Wenzel. *Using Axiomatic Type Classes in Isabelle*, 2000. Part of the Isabelle documentation, http://isabelle.in.tum.de/doc/axclass.pdf.
[35] F. Wiedijk. Mizar: An impression. Unpublished paper, 1999. http://www.cs.kun.nl/~freek/mizar/mizarintro.ps.gz.