Logic-free reasoning in Isabelle/Isar

Stefan Berghofer and Makarius Wenzel*

Technische Universität München
Institut für Informatik, Boltzmannstraße 3, 85748 Garching, Germany
http://www.in.tum.de/~berghofe/
http://www.in.tum.de/~wenzelm/

Abstract. Traditionally a rigorous mathematical document consists of a sequence of definition – statement – proof. Taking this basic outline as starting point we investigate how these three categories of text can be represented adequately in the formal language of Isabelle/Isar.
Proofs represented in human-readable form have been the initial motivation of Isar language design 10 years ago. The principles developed here allow to turn deductions of the Isabelle logical framework into a format that transcends the raw logical calculus, with more direct description of reasoning using pseudo-natural language elements.
Statements describe the main result of a theorem in an open format as a reasoning scheme, saying that in the context of certain parameters and assumptions certain conclusions can be derived. This idea of turning Isar context elements into rule statements has been recently refined to support the dual form of elimination rules as well.
Definitions in their primitive form merely name existing elements of the logical environment, by stating a suitable equation or logical equivalence. Inductive definitions provide a convenient derived principle to describe a new predicate as the closure of given natural deduction rules. Again there is a direct connection to Isar principles, rules stemming from an inductive characterization are immediately available in structured reasoning.
All three categories benefit from replacing raw logical encodings by native Isar language elements. The overall formality in the presented mathematical text is reduced. Instead of manipulating auxiliary logical connectives and quantifiers, the mathematical concepts are emphasized.

1 Introduction

Isabelle/Isar [13, 14, 15] enables to produce formal mathematical documents with full proof checking. Similar in spirit to the Mizar system [12, 11], the user writes text in a formal language that is checked by the machine. As a side-effect of this, Isabelle/Isar produces high-quality documents using existing LaTeX technology: the present paper is an example of such a formally processed document.

Rigorous mathematics is centered around *proofs*, and this view is taken to the extreme in Isabelle/Isar. The demands for human-readable proofs, which is the hardest part in formalized mathematics, are taken as guidelines for the design

* Supported by BMBF project "Verisoft" (grant 01 IS C38)

of the much simpler elements of *statements* and *definitions*. While the initial conception of the Isar proof language dates back almost 10 years, some more recent additions help to express structured statements and inductive definitions even more succinctly, in a "logic-free" style. This enables higher Isar idioms to focus on the mathematics of the application at hand, instead of demanding recurrent exercises in formal logic from the user. So *mathematical reasoning* is emphasized, and auxiliary logical constructions are left behind eventually.

Our basic approach works essentially in *bottom-up* manner, starting from primitive logical principles towards mathematical reasoning that is eventually free from the logic (which better serves in the background for foundational purposes only). As the art of human-readable formal reasoning evolves further, we hope to move towards a stage that meets with other approaches that are coming the *top-down* way from informal mathematics.

Overview. §2 reviews the original idea of "natural deduction" due to Gentzen, and its implementation in the Isabelle/Pure framework. §3 gives an overview of the Isar proof language as a linearized expression of structured proofs in the underlying logical framework. §4 introduces structured Isar statements, which enable to state and prove reasoning schemes conveniently, without going through the logical framework again. §5 covers a recent refinement of the well-known concept of inductive definitions, which enables to obtain natural deduction rules directly from basic definitions, without intermediate statements or proofs. §6 illustrates the benefits of the native "logic-free" style of Isar definitions, statements, and proofs by an example about well-founded multiset ordering.

2 Natural Deduction Revisited

About 75 years ago Gentzen introduced a logical calculus for "natural deduction" [3] that was intended to formalize the way mathematical reasoning actually works, unlike earlier calculi due to Hilbert and Russel. Since we share the motivation to approximate mathematical reasoning, we briefly review some aspects of traditional natural deduction as relevant for Isabelle/Isar.

Gentzen uses a two-dimensional diagrammatic representation of reasoning patterns, which may be composed to proof trees according to certain principles. Each logical connective is characterized by giving *introduction rules* and *elimination rules*. This is illustrated for $\longrightarrow$ and $\forall$ as follows (in our notation):

$$\frac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \longrightarrow B}\ (\longrightarrow I) \qquad \frac{A \longrightarrow B \quad A}{B}\ (\longrightarrow E)$$

$$\frac{\begin{array}{c}[a]\\ \vdots\\ B(a)\end{array}}{\forall x.\ B(x)}\ (\forall\, I) \qquad \frac{\forall x.\ B(x)}{B(a)}\ (\forall\, E)$$

Inferences work by moving from assumptions (upper part) to conclusions (lower part). Nested inferences, as indicated by three dots and brackets, allow to refer to *local* assumptions or parameters, which are *discharged* when forming the final conclusion. Note that in $(\forall I)$ we have treated the locally "fresh" parameter a analogous to an assumption, which reflects the formal treatment in the Isabelle framework. Traditional logic texts often treat this important detail merely as a footnote ("eigenvariable condition").

The Isabelle/Pure framework [8, 9] implements a *generic* version of higher-order natural deduction, without presupposing any particular object-logic. Natural deduction rules are represented in Isabelle as propositions of the "meta-logic", which provides the framework connectives of implication $A \Longrightarrow B$ and quantification $\bigwedge x.\ B\ x$. This first-class representations of primitive and derived natural deduction rules is supported by two main operations: *resolution* for mixed forward-backward chaining of partial proof trees, and *assumption* for closing branches. Both may involve higher-order unification, which results in a very flexible rule-calculus that resembles higher-order logic programming [15, §2.2].

According to the initial "logical framework" idea of Isabelle [8, 9], the user may specify a new object-logic by declaring connectives as (higher-order) term constants, and rules as axioms. For example, the minimal logic of $\longrightarrow$ and $\forall$ could be declared as follows (using type i for individuals and o for propositions):

$$\begin{array}{ll}
\mathit{imp} & :: o \Rightarrow o \Rightarrow o \qquad (\textbf{infix } \longrightarrow) \\
\mathit{impI} & : \bigwedge A\ B.\ (A \Longrightarrow B) \Longrightarrow A \longrightarrow B \\
\mathit{impE} & : \bigwedge A\ B.\ (A \longrightarrow B) \Longrightarrow A \Longrightarrow B \\
\mathit{all} & :: (i \Rightarrow o) \Rightarrow o \qquad (\textbf{binder } \forall) \\
\mathit{allI} & : \bigwedge B.\ (\bigwedge a.\ B\ a) \Longrightarrow \forall x.\ B\ x \\
\mathit{allE} & : \bigwedge a\ B.\ (\forall x.\ B\ x) \Longrightarrow B\ a
\end{array}$$

Note that outermost $\bigwedge$ is usually left implicit. The above rules merely reflect the minimal logic of $\Longrightarrow$ and $\bigwedge$ of the framework. The idea of generic natural deduction becomes more apparent when the object-logic is enriched by further connectives, for example:

$$\begin{array}{ll}
\mathit{conj} & :: o \Rightarrow o \Rightarrow o \qquad (\textbf{infix } \wedge) \\
\mathit{conjI} & : A \Longrightarrow B \Longrightarrow A \wedge B \\
\mathit{conjE} & : A \wedge B \Longrightarrow (A \Longrightarrow B \Longrightarrow C) \Longrightarrow C \\
\mathit{disj} & :: o \Rightarrow o \Rightarrow o \qquad (\textbf{infix } \vee) \\
\mathit{disjI}_1 & : A \Longrightarrow A \vee B \\
\mathit{disjI}_2 & : B \Longrightarrow A \vee B \\
\mathit{disjE} & : A \vee B \Longrightarrow (A \Longrightarrow C) \Longrightarrow (B \Longrightarrow C) \Longrightarrow C \\
\mathit{ex} & :: (i \Rightarrow o) \Rightarrow o \qquad (\textbf{binder } \exists) \\
\mathit{exI} & : B\ a \Longrightarrow \exists x.\ B\ x \\
\mathit{exE} & : (\exists x.\ B\ x) \Longrightarrow (\bigwedge a.\ B\ a \Longrightarrow C) \Longrightarrow C
\end{array}$$

These rules for predicate logic follow Gentzen [3], except for conjunction elimination. Instead of two projections $A \wedge B \Longrightarrow A$ and $A \wedge B \Longrightarrow B$, our *conjE* rule enables to assume local facts A and B, independently from the main goal. Other

typical situations of elimination are represented by *disjE*, which splits the main goal into two cases with different local assumptions, and *exE*, which augments the main goal by a local parameter a such that $B\ a$ may be assumed.

This uniform presentation of eliminations is typical for Isabelle/Pure [8, 9], but often appears peculiar to users without a strong background in formal logic. Even in Gentzen's original article, the *disjE* and *exE* rules are explained with special care, while "the other rules should be easy to understand" [3]. In the Isar proof language (§3), we shall provide a refined view on elimination, that expresses directly the idea of being able to assume local assumptions over local parameters, potentially with a case-split involved.

The examples for natural deduction presented so far have referred to traditional connectives of predicate logic: $\longrightarrow$, $\forall$, $\wedge$, $\vee$, $\exists$ etc. There is nothing special about these in the generic framework of Isabelle/Pure. We may just as well reason directly with concepts of set theory, lattice theory etc. without going through predicate logic again. Here are natural deduction rules for $x \in A \cap B$:

$$\begin{array}{l} \mathit{interI} : x \in A \Longrightarrow x \in B \Longrightarrow x \in A \cap B \\ \mathit{interE} : x \in A \cap B \Longrightarrow (x \in A \Longrightarrow x \in B \Longrightarrow C) \Longrightarrow C \end{array}$$

In practice, such domain-specific rules are not axiomatized, but derived from the definitions of the underlying concepts. In fact, the majority of rules will be of the latter kind — after the initial object-logic axiomatization, regular users proceed in this strictly definitional manner. Thus the role of the logical framework as foundation for new logics is changed into that of a tool for plain mathematical reasoning with derived concepts. Then the main purpose of the special connectives $\Longrightarrow$ and $\bigwedge$ is to outline *reasoning patterns* in a "declarative" fashion. Guided by the indicated structure of natural deduction rules, structured proofs are composed internally by means of the *resolution* and *assumption* principles.

The remaining question is how to obtain natural deduction rules conveniently. As we shall see later (§5), a refined version of the well-known concept of *inductive definitions* allows to produce elimination rules quite naturally from a "logic-free" specification of the introduction rules only. The system will derive a proper predicate definition internally, and derive the corresponding rules, which may then be turned immediately into Isar proof texts in the application.

3 Isar Proofs

The Isar proof language [13, 14, 15] enables to express formal natural deduction proofs in a linear form that approximates traditional mathematical reasoning. Gentzen [3] admits that his calculus looses information present in the "narrated" version of informal reasoning: it is unclear where to start reading two-dimensional proof trees. This linguistic structure is recovered in Isabelle/Isar: proof texts are written with pseudo-natural language elements, which are interpreted by the Isar proof processor in terms of the underlying logical framework of Isabelle/Pure, see [15, §3.3] for further details.

It is important to understand that Isar is not another calculus, but a *language* that is interpreted by imposing certain policies on the existing rule calculus of Isabelle/Pure. To this end, Isar introduces non-logical concepts to organize formal entities notably the *proof context*, the *goal state* (optional), and a register for the latest *result*. The overall proof configuration is arranged as a stack over these components, which enables block-structured reasoning within a flat logic.

An Isar proof body resembles a mathematical notepad: statements of various kinds may be written down, some refer to already established facts (**note**), some extend the context by new parameters and assumptions (**fix** and **assume**), some produce derived results (**have** and **show**, followed by a sub-proof). Moreover, there are several elements to indicate the information flow between facts and goals, notably **then**, **from**, **with**, **using**, **also**, **finally**, **moreover**, **ultimately**.

Previous facts may be referenced either by name, or by a literal proposition enclosed in special parentheses. For example, in the scope of **assume** a: A, both a and $\langle A \rangle$ refer to the same (hypothetical) result. In the subsequent examples, we mostly use the latter form for clarity. The labelled version is preferable in larger applications, when propositions are getting bigger. The special name *this* always refers to the result of the last statement.

From the perspective of the logical framework, the main purpose of Isar is to produce and compose natural deduction rules. The most elementary way to produce a rule works by concluding a result within the local scope of some extra hypotheses, which are discharged when leaving the scope. For example:

{
  fix x **and** y
  assume $A\ x$ **and** $B\ y$
  have $C\ x\ y$ ⟨*proof*⟩
}
note $\langle \bigwedge x\ y.\ A\ x \Longrightarrow B\ y \Longrightarrow C\ x\ y \rangle$

Within the body of a sub-proof, **fix**–**assume**–**show** yields a rule as above, but the result is used to refine a pending subgoal (matching both the assumptions and conclusion as indicated in the text). The structure of the goal tells which assumptions are admissible in the sub-proof, but there is some flexibility due to the way back-chaining works in the logical framework. For example:

have $\bigwedge x\ y.\ A\ x \Longrightarrow B\ y \Longrightarrow C\ x\ y$
proof −
  fix x **and** y
  assume $A\ x$ **and** $B\ y$
  show $C\ x\ y$ ⟨*proof*⟩
qed

have $\bigwedge x\ y.\ A\ x \Longrightarrow B\ y \Longrightarrow C\ x\ y$
proof −
  fix y **assume** $B\ y$
  fix x **assume** $A\ x$
  show $C\ x\ y$ ⟨*proof*⟩
qed

The **proof** and **qed** elements are not just delimiters, but admit initial and terminal refinements of pending goals. The default for **proof** is to apply a canonical elimination or introduction rule declared in the background context, using the "*rule*" method. The default for **qed** is to do nothing; in any case the final stage of concluding a sub-proof is to finish pending sub-goals trivially by assump-

tion. Further abbreviations for terminal proofs are "**by** *method*$_1$ *method*$_2$" for "**proof** *method*$_1$ **qed** *method*$_2$", and "**..**" for "**by** *rule*", and "**.**" for "**by** *this*".

With standard introduction and elimination rules declared in the library, we can now rephrase natural deduction schemes (§2) as linear Isar text:

have $A \longrightarrow B$
proof
  assume A
  show B ⟨*proof*⟩
qed

assume $A \longrightarrow B$ **and** A
then have B **..**

have $\forall x.\ B\ x$
proof
  fix a
  show $B\ a$ ⟨*proof*⟩
qed

assume $\forall x.\ B\ x$
then have $B\ a$ **..**

Here we have mimicked Gentzen's diagrammatic reasoning, composing proof texts according to the structure of the underlying rules. Isar is much more flexible in arranging natural deduction proof outlines, though. Some of the rule premises may be established beforehand and pushed into the goal statement; the proof body will only cover the remaining premises. This allows mixed forward-backward reasoning according to the following general pattern:

from *facts*$_1$ **have** *props* **using** *facts*$_2$ **proof** (*method*$_1$) *body* **qed** (*method*$_2$)

For example, premise $A \longrightarrow B$ could be provided either as "**from** ‹$A \longrightarrow B$›" before the goal, as "**using** ‹$A \longrightarrow B$›" after the goal, or as "**show** $A \longrightarrow B$" in the body. It is up to the author of the proof to arrange facts adequately, to gain readability by the most natural flow of information. Sub-structured premises are usually addressed within a sub-proof, using **fix**–**assume**–**show** in backwards mode, as seen in the above introduction proofs of $\longrightarrow$ and $\forall$.

The other rules from §2 can be directly turned into Isar proof texts as well, but eliminations of the form $\ldots \Longrightarrow (\bigwedge a.\ B\ a \Longrightarrow C) \Longrightarrow C$ demand special attention. A naive rendering in Isar would require the main goal C given beforehand, and a sub-proof that proves the same C in a context that may be enriched by additional parameters and assumptions. Isar's **obtain** element [15, §3.1] supports this style of reasoning directly, in a logic-free fashion. For example:

{
  obtain x **and** y **where** $A\ x$ **and** $B\ y$ ⟨*proof*⟩
  have C ⟨*proof*⟩
}
note ‹C›

The proof obligation of "**obtain** x **and** y **where** $A\ x$ **and** $B\ y$" corresponds to the rear-part of an eliminations rule: $(\bigwedge x\ y.\ A\ x \Longrightarrow B\ y \Longrightarrow \mathit{thesis}) \Longrightarrow \mathit{thesis}$ for a hypothetical *thesis* that is arbitrary, but fixed. Having finished that proof, the context is augmented by "**fix** x **and** y **assume** $A\ x$ **and** $B\ y$".

Results exported from that scope are unaffected by these additional assumptions, provided the auxiliary parameters are not mentioned in the conclusion!

We can now spell out the remaining natural deduction schemes of §2 adequately, only *disjE* requires explicit sub-proofs involving the main conclusion C, because **obtain** cannot split a proof text into several cases.

assume A **and** B
then have $A \wedge B$ **..**

assume $A \wedge B$
then obtain A **and** B **..**

assume A
then have $A \vee B$ **..**

assume B
then have $A \vee B$ **..**

assume $A \vee B$
then have C
proof
  assume A
  then show C ⟨*proof*⟩
next
  assume B
  then show C ⟨*proof*⟩
qed

assume $B\ a$
then have $\exists x.\ B\ x$ **..**

assume $\exists x.\ B\ x$
then obtain a **where** $B\ a$ **..**

4 Isar Statements

Isar proof composition is centered around natural deduction rules of the logical framework. Such rules may be established as regular theorems like this:

theorem r: $\bigwedge x\ y.\ A\ x \Longrightarrow B\ y \Longrightarrow C\ x\ y$
proof −
  fix x **and** y
  assume $A\ x$ **and** $B\ y$
  show $C\ x\ y$ ⟨*proof*⟩
qed

This is slightly unsatisfactory, because the structure of the result is specified redundantly in the main statement and the proof body, using framework connectives $\bigwedge/\Longrightarrow$ vs. Isar proof elements **fix**–**assume**–**show**, respectively. Moreover, exposing the Isabelle/Pure rendering of the intended reasoning scheme gives the head statement a rather technical appearance. This is even worse for elimination rules, due to extra rule nesting $\ldots \Longrightarrow (\bigwedge a.\ B\ a \Longrightarrow C) \Longrightarrow C$ etc.

Isar statements address these issues by introducing first-class notation for certain rule schemes. As seen in the initial example in §3, proof blocks allow to produce natural deduction rules on the spot, by discharging local parameters and assumptions, e.g. "**{ fix** x **assume** $A\ x$ **have** $B\ x$ ⟨*proof*⟩ **}**" for $\bigwedge x.\ A\ x \Longrightarrow B\ x$. Based on this idea we introduce three kinds of clausal Isar statements.

1. *Big clauses* have the form "**fixes** *vars* **assumes** *props* **shows** *props*" and specify the outermost structure of a natural deduction reasoning pattern.

The given **fixes** and **assumes** elements determine a local context, **shows** poses simultaneous local goals within that. The subsequent proof proceeds directly within the local scope; the final result emerges by discharging the context, producing corresponding $\bigwedge/\Longrightarrow$ rule structure behind the scenes.

2. *Dual clauses* have the form "**fixes** *vars* **assumes** *props* **obtains** *vars* **where** *props*" and abbreviate certain big clauses: "**obtains** a **where** $B\ a$" expands to "**fixes** *thesis* **assumes** $\bigwedge a.\ B\ a \Longrightarrow$ *thesis* **shows** *thesis*". Case-splits may be indicated by several clauses separated by "|", which corresponds to multiple branches of the form $\bigwedge a_i.\ B_i\ a_i \Longrightarrow$ *thesis*. According to the principles behind big clauses, the resulting rule will have exactly the elimination format described in §2. Within the proof body, each **obtains** case corresponds to a different hypothetical rule to conclude the main *thesis*; one of these possibilities has to be chosen eventually.
3. *Small clauses* are of the form "$B\ x$ **if** $A\ x$ **for** x" and indicate the second-level rule structure of framework propositions within big clauses. This corresponds directly to $\bigwedge x.\ A\ x \Longrightarrow B\ x$, but clausal notation may not be nested further.

The basic **fixes**–**assumes**–**shows** form of big clauses has been available in Isabelle/Isar for many years. The dual form is a recent addition, which has first appeared officially in Isabelle2007. Small clauses are not available in official Isabelle yet, but are an experimental addition for the present paper only.

Our initial proof of $\bigwedge x\ y.\ A\ x \Longrightarrow B\ y \Longrightarrow C\ x\ y$ is now rephrased as follows:

theorem r:
fixes x **and** y
assumes $A\ x$ **and** $B\ y$
shows $C\ x\ y$ ⟨*proof*⟩

See also §6 for proofs involving **obtains**. To continue our running example of predicate logic, we rephrase the previous natural deduction rules from §2:

theorem *impI*: **assumes** B **if** A **shows** $A \longrightarrow B$
theorem *impE*: **assumes** $A \longrightarrow B$ **and** A **obtains** B

theorem *allI*: **assumes** $B\ a$ **for** a **shows** $\forall x.\ B\ x$
theorem *allE*: **assumes** $\forall x.\ B\ x$ **obtains** $B\ a$

theorem *conjI*: **assumes** A **and** B **shows** $A \wedge B$
theorem *conjE*: **assumes** $A \wedge B$ **obtains** A **and** B

theorem $disjI_1$: **assumes** A **shows** $A \vee B$
theorem $disjI_2$: **assumes** B **shows** $A \vee B$
theorem *disjE*: **assumes** $A \vee B$ **obtains** A | B

theorem *exI*: **assumes** $B\ a$ **shows** $\exists x.\ B\ x$
theorem *exE*: **assumes** $\exists x.\ B\ x$ **obtains** a **where** $B\ a$

In other words, we have managed to express the inherent structure of reasoning schemes without demanding auxiliary logical connectives, not even those of the Isabelle/Pure framework. Only concepts of the application, which happens to be

predicate logic as an object-language here, and native Isar elements are involved. The same works for domain-specific rules, e.g. those for set theory seen before:

theorem *interI*: **assumes** $x \in A$ **and** $x \in B$ **shows** $x \in A \cap B$
theorem *interE*: **assumes** $x \in A \cap B$ **obtains** $x \in A$ **and** $x \in B$

5 Inductive Definitions

Inductive predicates provide a convenient way to define concepts by specifying a collection of characteristic *introduction rules*. Support for inductive definitions is available in many theorem provers. Melham [6] describes a version for the HOL system using an impredicative encoding, meaning that the definition involves universal quantification over predicate variables, whereas Harrison's inductive definition package for HOL [4] uses an encoding based on the Knaster-Tarski fixpoint theorem. The Coq system [2] is based on the *Calculus of Inductive Constructions* introduced by Paulin-Mohring, which contains inductive definitions as a primitive concept [7]. Inductive definitions in Isabelle were first introduced by Paulson [10], using fixpoints over the lattice of sets. Our refined version works on generic lattices, which subsume predicates in HOL.

Many well-known concepts of mathematics can be viewed as an inductive predicate. E.g. the transitive closure of a relation can be defined as follows:

inductive *trcl* **for** $R :: \alpha \Rightarrow \alpha \Rightarrow bool$
where
trcl R x x **for** *x*
| *trcl R x z* **if** *R x y* **and** *trcl R y z* **for** *x y z*

The rules of **inductive** may be specified using the format of "small clauses" introduced in §4. Internally, the system derives further natural deduction rules that may be turned into Isar proofs as discussed in §3. By virtue of its definition as the least predicate closed under these rules, any inductive predicate admits an *induction* and an *inversion* principle (case analysis). For example:

assume *trcl R a b*
then have *P a b*
proof (*rule trcl.induct*)
fix *x*
show *P x x* ⟨*proof*⟩ — induction base
next
fix *x y z*
assume *R x y* **and** *trcl R y z* **and** *P y z*
then show *P x z* ⟨*proof*⟩ — induction step
qed

This induction principle is a consequence of *trcl* being defined as the least fixpoint of a *predicate transformer* of type $(\alpha \Rightarrow \alpha \Rightarrow bool) \Rightarrow \alpha \Rightarrow \alpha \Rightarrow bool$:

trcl $\equiv$

$\lambda R.\ \mathit{lfp}\ (\lambda p\ x_1\ x_2.$
$\quad (\exists x.\ x_1 = x \wedge x_2 = x) \vee$
$\quad (\exists x\ y\ z.\ x_1 = x \wedge x_2 = z \wedge R\ x\ y \wedge p\ y\ z))$

The body of the function $(\lambda p\ x_1\ x_2.\ \ldots)$ is a disjunction, whose two parts correspond to the two introduction rules for *trcl*. Using the fact that the predicate transformer is monotonic, the induction principle follows from this definition using the Knaster-Tarski theorem for least fixpoints on complete lattices:

$$\frac{\mathit{mono}\ f \qquad f\ (\mathit{lfp}\ f \sqcap P) \sqsubseteq P}{\mathit{lfp}\ f \sqsubseteq P}$$

The ordering relation $\sqsubseteq$ and the infimum operator $\sqcap$ is defined on the complete lattice of n-ary predicates in a pointwise fashion:

$$P \sqsubseteq Q \equiv \forall x_1\ \ldots\ x_n.\ P\ x_1\ \ldots\ x_n \longrightarrow Q\ x_1\ \ldots\ x_n$$
$$P \sqcap Q \equiv \lambda x_1\ \ldots\ x_n.\ P\ x_1\ \ldots\ x_n \wedge Q\ x_1\ \ldots\ x_n$$

The premise $f\ (\mathit{lfp}\ f \sqcap P) \sqsubseteq P$ of the fixpoint theorem is established by the proofs of the induction base and the induction step in the above proof pattern.

Case analysis corresponds to the observation that if an inductive predicate holds, one of its introduction rules must have been used to derive it. This principle can be viewed as a degenerate form of induction, since there is no induction hypothesis. For the transitive closure, the case analysis scheme is:

assume *trcl R a b*
then have *Q*
proof (*rule trcl.cases*)
  fix *x*
  assume $a = x$ **and** $b = x$
  then show *Q* ⟨*proof*⟩
next
  fix *x y z*
  assume $a = x$ **and** $b = z$ **and** *R x y* **and** *trcl R y z*
  then show *Q* ⟨*proof*⟩
qed

Although the case analysis rule could be derived from the above least fixpoint theorem as well, it is proved from the fixpoint unfolding theorem $\mathit{mono}\ f \Longrightarrow \mathit{lfp}\ f = f\ (\mathit{lfp}\ f)$ which has the advantage that exactly the same proof technique can also be used in the case of *coinductive* predicates, using *gfp* in place of *lfp*.

Inductive predicates are very convenient to formalize mathematical concepts succinctly, even if there is no recursion involved. For example, the composition of two relations R and S can be defined as follows:

inductive *comp* **for** $R :: \alpha \Rightarrow \beta \Rightarrow \mathit{bool}$ **and** $S :: \beta \Rightarrow \gamma \Rightarrow \mathit{bool}$
  where *comp R S x z* **if** *R x y* **and** *S y z* **for** *x y z*

For *comp*, the underlying primitive definition is $\mathit{comp} \equiv \lambda R\ S.\ \mathit{lfp}\ (\lambda p\ x_1\ x_2.\ \exists x\ y\ z.\ x_1 = x \wedge x_2 = z \wedge R\ x\ y \wedge S\ y\ z)$. For fixpoints of constant functions like the above we have $\mathit{lfp}\ (\lambda x.\ t) = t$, which easily follows from the fixpoint

unfolding theorem. Using the same principles, we can even characterize basic operators of predicate logic as inductive predicates with zero arguments. E.g.

inductive *and* **for** A B :: *bool*
  where *and* A B **if** A **and** B

inductive *or* **for** A B :: *bool*
  where *or* A B **if** A | *or* A B **if** B

inductive *exists* **for** B :: $\alpha \Rightarrow$ *bool*
  where *exists* B **if** B a **for** a

Again, these operators are just examples. Real applications would introduce their genuine notions directly as inductive definitions.

6 Case-study: Well-founded Multiset Ordering

To illustrate the "logic-free" style of definitions, statements and proofs in Isar, we formalize some aspects of well-founded multiset ordering. A multiset is a finite "bag" of items, which can be modeled as a function from items to natural numbers that yields a non-zero value only on a finite domain. Multiset notation is reminiscent of plain sets: $\{\!|a, a, b, b, b, c|\!\}$ for enumeration, $a \in B$ for membership, $A \uplus B$ for union etc. The structure of multisets can also be characterized inductively, with base case $\{\!|\}\!\}$ and step case $B \uplus \{\!|a|\!\}$ for a multiset B.

Given an ordering on items, multisets can be ordered by the following intuitive process: one item of the bag is removed and replaced by the content of another bag of strictly smaller items; this is repeated transitively. The main theorem states that the resulting relation on multisets is well-founded, provided that the item ordering is well-founded. Below we merely cover the basic definitions and a technical lemma required for the well-foundedness proof.[1]

Our development refers to a locally fixed *less* relation, which is introduced by commencing the following locale context (see also [1]).

locale *less-relation* = **fixes** *less* :: $\alpha \Rightarrow \alpha \Rightarrow$ *bool* (**infix** $\prec$ 50)
begin

The locale already contributes to the "logic-free" approach, since it avoids explicit abstraction or quantification over that parameter.

A bag of items is compared to a single item in point-wise manner as follows:

definition *lesser* (**infix** $\prec$ 50) **where** $B \prec a \leftrightarrow (\forall b.\ b \in B \longrightarrow b \prec a)$

lemma *lesserI*: **assumes** $b \prec a$ **for** b **shows** $B \prec a$
  using *assms* **unfolding** *lesser-def* **by** *auto*

[1] See `http://isabelle.in.tum.de/dist/library/HOL/Library/Multiset.html` for a rather old version of the complete formalization that mixes quite different styles; the main well-foundedness theorem is called *wf-mult* there.

lemma *lesserE*: **assumes** $B \prec a$ **and** $b \in B$ **obtains** $b \prec a$
  using *assms* **unfolding** *lesser-def* **by** *auto*

Obviously, the primitive predicate definition of $B \prec a$ is *not* logic-free, since it uses $\forall$ and $\longrightarrow$ connectives. The two extra "boiler plate" lemmas amend this by providing an alternative characterization in natural deduction style. (In the bits of proof shown below, we never need to analyze the *lesser* relation, though).

Next we define the main idea of the multiset ordering process. The subsequent inductive predicate $N \prec\prec M$ expresses a single step of splitting off an element from $M = B \uplus \{\!|a|\!\}$ and replacing it by a point-wise smaller multiset. (The full ordering emerges as the transitive closure of that relation.)

inductive *less-multiset* (**infix** $\prec\prec$ 50)
  where $B \uplus C \prec\prec B \uplus \{\!|a|\!\}$ **if** $C \prec a$ **for** $a\ B\ C$

This rather succinct logic-free definition characterizes the relation by a single clause — there are no other cases and no recursion either. Even this degenerate form of inductive definition is very convenient in formal reasoning. Here the decomposition of the two multisets is specified directly via pattern matching, with side-conditions and parameters expressed as native clauses of Isabelle/Isar.

In contrast, the original formulation from the Isabelle/HOL library uses an encoding that involves intermediate layers of predicate logic and set theory, with separate equations to express the decomposition.

definition *less-mult* =
  $\{(N, M).\ \exists a\ B\ C.\ M = B \uplus \{\!|a|\!\} \wedge N = B \uplus C \wedge C \prec a\}$

While this might look familiar to anybody trained in logic, manipulating such auxiliary structure in formal proof requires extra steps that do not contribute to the application. Nevertheless, even rather bulky encodings do often happen to work out in practice by means of reasonably strong "proof automation". We illustrate this by proving formally that both definitions are equivalent.

lemma $N \prec\prec M \leftrightarrow (N, M) \in$ *less-mult*
  unfolding *less-mult-def*
proof
  assume $N \prec\prec M$
  then obtain $a\ B\ C$ **where** $M = B \uplus \{\!|a|\!\}$ **and** $N = B \uplus C$ **and** $C \prec a$
    by (*rule less-multiset.cases*)
  then show $(N, M) \in \{(N, M).\ \exists a\ B\ C.\ M = B \uplus \{\!|a|\!\} \wedge N = B \uplus C \wedge C \prec a\}$
    by *auto*
next
  assume $(N, M) \in \{(N, M).\ \exists a\ B\ C.\ M = B \uplus \{\!|a|\!\} \wedge N = B \uplus C \wedge C \prec a\}$
  then obtain $a\ B\ C$ **where** $M = B \uplus \{\!|a|\!\}$ **and** $N = B \uplus C$ **and** $C \prec a$
    by *auto*
  from $\langle C \prec a \rangle$ **have** $B \uplus C \prec\prec B \uplus \{\!|a|\!\}$ **by** (*rule less-multiset.intros*)
  with $\langle M = B \uplus \{\!|a|\!\} \rangle$ **and** $\langle N = B \uplus C \rangle$ **show** $N \prec\prec M$ **by** *simp*
qed

This rather lengthy proof merely shuffles logical connectives back and forth, without being very informative. The *auto* method involved here is a fully-featured

combination of classical proof search with equational normalization; it successfully bridges the gap between the intermediate statements given in the text. On the other hand, this extra overhead can be avoided by the logic-free characterization of the inductive definition from the very beginning. So we continue in that style now, working on the mathematics of multiset orderings instead of doing exercises in formal logic and automated reasoning.

The proof of the main theorem combines well-founded induction over the relation $\prec$ of items with structural induction over multisets. At some point in the induction step, the multiset ordering $N \prec\prec B \uplus \{\!|a|\!\}$ needs to be analyzed:

lemma *less-add-cases*:
  assumes $N \prec\prec B \uplus \{\!|a|\!\}$
  obtains
    (1) M **where** $M \prec\prec B$ **and** $N = M \uplus \{\!|a|\!\}$
  | (2) C **where** $C \prec a$ **and** $N = B \uplus C$

Ultimately, the rule resulting from this goal statement will split an arbitrary fact $N \prec\prec B \uplus \{\!|a|\!\}$ into two cases as specified above. In the present proof context, we are still in the course of establishing this claim. Here $N \prec\prec B \uplus \{\!|a|\!\}$ is available as a local fact, and there are two possibilities to finish the hypothetical main *thesis*, namely rule 1: *thesis* **if** $M \prec\prec B$ **and** $N = M \uplus \{\!|a|\!\}$ **for** M and rule 2: *thesis* **if** $C \prec a$ **and** $N = B \uplus C$ **for** C.

This means the subsequent proof already starts out in a nicely decomposed version of the idea of splitting cases and obtaining local parameters and assumptions, without having to work through auxiliary $\vee$, $\wedge$, $\exists$ connectives again:

proof −
  from ‹$N \prec\prec B \uplus \{\!|a|\!\}$›
  obtain $a'\ B'\ C$ **where**
      $B \uplus \{\!|a|\!\} = B' \uplus \{\!|a'|\!\}$ **and**
      $N = B' \uplus C$ **and**
      $C \prec a'$
    by (*rule less-multiset.cases*) *simp-all*
  from ‹$B \uplus \{\!|a|\!\} = B' \uplus \{\!|a'|\!\}$› **show** *thesis*
  proof (*rule add-eq-cases*)
    assume $B = B'$ **and** $a = a'$
    with ‹$C \prec a'$› **and** ‹$N = B' \uplus C$›
    have $C \prec a$ **and** $N = B \uplus C$ **by** *simp-all*
    then show *thesis* **by** (*rule* 2)
  next
    fix C' **assume** $B' = C' \uplus \{\!|a|\!\}$ **and** $B = C' \uplus \{\!|a'|\!\}$
    show *thesis*
    proof (*rule* 1)
      from ‹$C \prec a'$› **have** $C' \uplus C \prec\prec C' \uplus \{\!|a'|\!\}$ **by** (*rule less-multiset.intros*)
      with ‹$B = C' \uplus \{\!|a'|\!\}$› **show** $C' \uplus C \prec\prec B$ **by** *simp*
      from ‹$B' = C' \uplus \{\!|a|\!\}$› **and** ‹$N = B' \uplus C$›
      show $N = C' \uplus C \uplus \{\!|a|\!\}$ **by** (*simp add*: *union-ac*)
    qed
  qed
qed

Above the initial **obtain** statement augments the local context by means of standard elimination of the $N \prec\!\!\!\prec M$ relation, using the corresponding *cases* rule. The sub-proof via *add-eq-cases* involves another **obtains** rule proven in the background library; its statement is structurally similar to *less-add-cases*.

So our proof manages to maintain the logic-free style, no auxiliary connectives are involved, only some algebraic operators from the application domain. The old proof in the Isabelle/HOL library requires about two times more text, even though it uses many abbreviations for sub-terms and local facts. Moreover, it needs more automation to work through extraneous logical structure.

end

7 Conclusion and Related Work

Isabelle/Isar shares the mission of formal reasoning that approximates traditional mathematical style with the pioneering Mizar system [12, 11]. There are many similarities and dissimilarities, see also [17, 16] for some comparison.

Concerning the logical foundations, Isar uses the Isabelle/Pure framework [8, 9] which implements a generic higher-order version of Gentzen's natural deduction calculus [3]. In contrast, Mizar works specifically with classical first-order logic, and the style of reasoning is modeled after the "supposition calculus" due to Jaskowski [5]. The basic paradigm of structured proof composition in Mizar is quite different from Isar. Where Isar revolves around natural deduction rules that emerge from local proof bodies and refine pending goals eventually, Mizar allows to operate more directly on the logical structure of a claim in consecutive refinement steps: `let` to move past universal quantification, `assume` to move past an implication etc. In contrast, **fix** and **assume** in Isar do not operate on a goal structure, but construct a context that will impose a certain rule structure on the final **show** result. This can make a difference in practice: in proving an implication a Mizar proof needs to say `assume A` invariably, while in Isar the corresponding "**assume** A" is only required if that fact is actually used later.

Essentially, there are Mizar proof elements for each of the logical connectives of $\wedge$, $\vee$, $\longrightarrow$, $\forall$, $\exists$, but English words are used here both for the connectives and the corresponding proof elements. For example, the proposition `for x holds A[x]` can be established by `let x` and a proof of `A[x]` in that scope. Thus Mizar enables to produce a proof text according to principles from classical first-order logic, while Isar is more puristic in referring to generic natural deduction, where predicate logic is just one example. This different attitude is best illustrated by existential elimination, which works in Mizar via `consider a such that B[a]` and is closely tied to actual existential quantification `ex x st B[x]`. In Isar "**obtain** a **where** $B\ a$" merely espresses the more elementary idea of being able to augment the local scope by a hypothetical entity a with property $B\ a$. This might follow from a fact $\exists\, x.\ B\ x$, but the elimination is better performed by a domain-specific rule $\ldots \Longrightarrow (\bigwedge a.\ B\ a \Longrightarrow C) \Longrightarrow C$, or "**obtains** a **where** $B\ a$" as explained in the present paper. Our inductive definitions are particularly well suited to produce such rules.

This means certain aspects of Mizar are about predicate logic, rather than mathematics. In contrast, our "logic-free" style in Isar enables more direct expression of definitions, statements, and proofs — reducing the overall formality of the text.

References

[1] Ballarin, C.: Interpretation of locales in Isabelle: Theories and proof contexts. In: J.M. Borwein, W.M. Farmer (eds.) Mathematical Knowledge Management (MKM 2006), *LNAI*, vol. 4108. Springer-Verlag (2006)

[2] Barras, B., et al.: The Coq Proof Assistant Reference Manual. INRIA (2006)

[3] Gentzen, G.: Untersuchungen über das logische Schließen (I + II). Mathematische Zeitschrift **39** (1935)

[4] Harrison, J.: Inductive definitions: automation and application. In: P.J. Windley, T. Schubert, J. Alves-Foss (eds.) Higher Order Logic Theorem Proving and its Applications, *LNCS*, vol. 971. Springer-Verlag (1995)

[5] Jaskowski, S.: On the rules of suppositions. Studia Logica **1** (1934)

[6] Melham, T.F.: A package for inductive relation definitions in HOL. In: M. Archer, et al. (eds.) Higher Order Logic Theorem Proving and its Applications. IEEE Computer Society Press (1992)

[7] Paulin-Mohring, C.: Inductive Definitions in the System Coq – Rules and Properties. In: M. Bezem, J.F. Groote (eds.) Proceedings of the conference Typed Lambda Calculi and Applications, *LNCS*, vol. 664. Springer-Verlag (1993)

[8] Paulson, L.C.: The foundation of a generic theorem prover. Journal of Automated Reasoning **5**(3) (1989)

[9] Paulson, L.C.: Isabelle: the next 700 theorem provers. In: P. Odifreddi (ed.) Logic and Computer Science. Academic Press (1990)

[10] Paulson, L.C.: A fixedpoint approach to (co)inductive and (co)datatype definitions. In: G. Plotkin, C. Stirling, M. Tofte (eds.) Proof, Language, and Interaction: Essays in Honour of Robin Milner. MIT Press (2000)

[11] Rudnicki, P.: An overview of the MIZAR project. In: 1992 Workshop on Types for Proofs and Programs. Chalmers University of Technology, Bastad (1992)

[12] Trybulec, A.: Some features of the Mizar language (1993). Presented at Turin.

[13] Wenzel, M.: Isar — a generic interpretative approach to readable formal proof documents. In: Y. Bertot, et al. (eds.) Theorem Proving in Higher Order Logics (TPHOLs 1999), *LNCS*, vol. 1690. Springer-Verlag (1999)

[14] Wenzel, M.: Isabelle/Isar — a versatile environment for human-readable formal proof documents. Ph.D. thesis, Institut für Informatik, TU München (2002)

[15] Wenzel, M.: Isabelle/Isar — a generic framework for human-readable proof documents. In: R. Matuszewski, A. Zalewska (eds.) From Insight to Proof — Festschrift in Honour of Andrzej Trybulec, *Studies in Logic, Grammar, and Rhetoric*, vol. 10(23). University of Białystok (2007). `http://www.in.tum.de/~wenzelm/papers/isar-framework.pdf`

[16] Wiedijk, F. (ed.): The Seventeen Provers of the World, *LNAI*, vol. 3600. Springer-Verlag (2006)

[17] Wiedijk, F., Wenzel, M.: A comparison of the mathematical proof languages Mizar and Isar. Journal of Automated Reasoning **29**(3-4) (2002)