CDMTCS
Research
Report
Series

Formal Proof: Reconciling Correctness and Understanding

C. S. Calude[1] and C. Müller[1,2]
[1]University of Auckland, NZ
[2]Jacobs University, Bremen, Germany

CDMTCS-354
March 2009

Centre for Discrete Mathematics and
Theoretical Computer Science

Formal Proof: Reconciling Correctness and Understanding

Cristian S. Calude[1] and Christine Müller[1,2]

[1] Department of Computer Science, University of Auckland, New Zealand
[2] Department of Computer Science, Jacobs University, Bremen, Germany
www.cs.auckland.ac.nz/∼cristian, kwarc.info/cmueller

A good proof is a proof that makes us wiser. Manin [41, p. 209].

Abstract. Hilbert's concept of formal proof is an ideal of rigour for mathematics which has important applications in mathematical logic, but seems irrelevant for the practice of mathematics. The advent, in the last twenty years, of proof assistants was followed by an impressive record of deep mathematical theorems formally proved. Formal proof is practically achievable. With formal proof, correctness reaches a standard that no pen-and-paper proof can match, but an essential component of mathematics — the insight and understanding — seems to be in short supply. So, what makes a proof understandable? To answer this question we first suggest a list of symptoms of understanding. We then propose a vision of an environment in which users can write and check formal proofs as well as query them with reference to the symptoms of understanding. In this way, the environment reconciles the main features of proof: correctness and understanding.

1 Introduction

From Pythagoras and Euclid to Hilbert and Bourbaki, mathematical proofs were essentially based on axiomatic-deductive reasoning. This view was repeatedly expressed by the most prominent mathematicians. For Bourbaki [11], *Depuis les Grecs, qui dit Mathématique, dit démonstration*, and for Mac Lane [37], *If a result has not yet been given valid proof, it isn't yet mathematics: we should strive to make it such.*

A *formal proof* is written in a formal language consisting of certain strings of symbols from a fixed alphabet. Formal proofs are precisely specified without any ambiguity because all notions are explicitly defined, no steps (no matter how small) are omitted, no appeal to any kind of intuition is made. They satisfy Hilbert's criterion of mechanical testing:

> *The rules should be so clear, that if somebody gives you what they claim is a proof, there is a mechanical procedure that will check whether the proof is correct or not, whether it obeys the rules or not.*

By making sure that every step is correct, one can tell once and for all whether a proof is correct or not, i.e. whether a theorem has been proved.

Hilbert's concept of formal proof is an ideal of rigour for mathematics which has important applications in mathematical logic (computability theory and proof theory), but seems irrelevant for the practice of mathematics.

An *informal* (pen-on-paper) *proof* is a rigorous argument expressed in a mixture of natural language and formulae (for some mathematicians an equal mixture is the best proportion) that is intended to convince a knowledgeable mathematician of the truth of a statement, the theorem. Routine logical inferences are omitted. "Folklore" results are used without proof. Depending on the area, arguments may rely on intuition. Informal proofs are the standard of presentation of mathematics in textbooks, journals, classrooms, and conferences. They are the product of a social process.

In theory, each informal proof can be converted into a formal proof. However, this is rarely, almost never, done in practice[3]. Bourbaki, who came closer to formal proving than most mathematicians, still declared that *formalized mathematics cannot in practice be written down in full*, a goal that is an *absolutely unrealizable* program.

Gödel's Incompleteness Theorem [25] shows that in every formal system satisfying a modicum of natural assumptions certain statements are true but not provable. In this sense, the formal approach to mathematics is not universal, not everything can be formally proved. Still, no universal alternative is available. Although a formal proof cannot guarantee 100% correctness because, for example, one cannot prove the correctness of the formal prover itself (a well-known result in computability theory, [24]) the certainty achieved is close to "certain"[4].

The advent, in the last twenty years, of proof assistants was followed by an impressive record of deep mathematical theorems formally proved. The list includes Gödel Incompleteness Theorem (1986)[5], the Fundamental Theorem of Calculus (1996), the Fundamental Theorem of Algebra (2000), the Four Colour Theorem (2004), Jordan's Curve Theorem (2005), the Prime Number Theorem (2008), see [32]. The December 2008 issue of the *Notices of AMS* includes four papers on formal proof: three general overviews [32, 33, 66] and one study case, the formal proof of the Four-Colour Theorem [29]. *Hilbert's standard of proof is practicable, it's becoming reality.*

An automatic prover can be used not only to check the validity of a formalised proof of a known mathematical result (as in the list of famous theorems enumerated above), but also to interactively help to "prove" new theorems. The informal proof of the main result in [19] benefited substantially from the process

[3] Russell and Whitehead 2,500-page opus *Principia Mathematica* [65] is a famous exception: a fully formalised mathematical book. Russell believed that *no human being will ever read through it.*

[4] Not all agree. *Practically, I am "certain" that the HW+OS+ML/Compiler/Runtime + Isabelle implementation is* not *fully trustable,* [62].

[5] It's ironic to have this theorem — which limits the power of formal proving — as the first formally proved important theorem.

of formalisation in the interactive theorem prover Isabelle [48] of one of the key results in algorithmic information theory, the Kraft-Chaitin Theorem.

The correctness achieved by formal proofs cannot be matched by pen-and-paper proofs. However, an essential component of mathematics — the insight and understanding — seems to be in short supply. So, what makes a proof understandable? While correctness can be formally defined, understanding is subjective, so much more difficult to pin down. Our solution is to suggest a list of symptoms of understanding and, with reference to these symptoms, to propose a framework that reconciles correctness and understanding.

The paper is organised as follows. In Section 2 we discuss a list of symptoms of understanding. In Section 3 we present an envisioned environment that provides services regarding these symptoms. Section 4 concludes the paper with a summary of services supporting symptoms and describes future work.

2 Understanding Mathematical Proof

The gap between correctness and understanding seems to be widening (see [15, 18, 16]). Should one abandon the axiomatic-deductive model, should one sacrifice understanding for efficiency, should one try other avenues?

Although most mathematicians agree that understanding is paramount to mathematics there is little consensus regarding the understanding of understanding. Understanding in mathematics may mean many things, but, usually, mathematicians have no difficulties in recognising it. In contrast with correctness, understanding is subjective and probably cannot be rigorously defined.

Inspired by the analysis in [8, p. 9–10] we propose a list of *symptoms* for detecting the understanding of a proof. We use the term *symptom* in analogy with its medical meaning. The list is not exhaustive, not all symptoms are necessary to identify understanding, and some symptoms overlap. Not all symptoms are equally important and ranking seems almost impossible. Many mathematicians may argue that the first two symptoms are the most important ones. Symptoms are discussed and illustrated sometimes with reference to the following lemma, which is presented with three proofs, one informal and two formal ones. The formal proofs were generated with Isabelle. The complete proof script, written by N. Hay, appears in [34]; it is part of a more complex proof for the Kraft-Chaitin Theorem [19]. The formal proof in Isar [64] was written by M. Wenzel [62].

Lemma 1. *For all (binary) strings x, y, xy extends x.*

Proof. (Informal) The relation 'n extends v' (written $u \supset v$) is defined by the following two rules: a) for every string u, $u \supset u$, b) for every strings u,v, if $u \supset v$ then $ui \supset v$, for every $i \in \{0, 1\}$.

Take two strings x, y. If y is the empty string then $xy = x \supset x$ by a). If $xy \supset x$ and $i \in \{0, 1\}$, then $x(yi) = (xy)i \supset x$ by associativity of concatenation, hypothesis, and b). □

Proof. (Formal: Isabelle proof script using recursive definitions)

```
fun extends :: "'A list ⇒ 'A list ⇒ bool"
```

```
where
"extends [] [] = True"
| "extends [] (y#ys) = False"
| "extends x [] = True"
| "extends (x#xs) (y#ys) = ((x=y) & (extends xs ys))"

lemma extends1 : "extends A [] "
  apply (induct A) apply(simp_all)
done
lemma extends2 : "extends (A@B) A "
  apply (induct A) apply(simp_all) apply(simp only: extends1)
done
```

Proof. (Formal: Isar proof using Natural Deduction rules)

```
theory Extends
imports Main
begin
definition extends :: "’a list ⇒ ’a list ⇒ bool"
  where "extends A B ⟷ (∃C. A = B @ C)"
lemma extendsI [intro]:
  assumes "A = B @ C" shows "extends A B"
  using assms unfolding extends_def by blast
lemma extendsE [elim]:
  assumes "extends A B" obtains C where "A = B @ C"
  using assms unfolding extends_def by blast
lemma extends0: "extends A A"
proof
  show "A = A @ []" by simp
qed
lemma extends1: "extends A []"
proof
  show "A = [] @ A" by simp
qed
lemma extends2: "extends (A @ B) A"
proof
  show "A @ B = A @ B" by simp
qed
lemma extends3:
  assumes "extends A’ A"
  shows "extends (A’ @ B) A"
proof -
  have "extends A’ A" by fact
  then obtain C’ where "A’ = A @ C’" ..
  then have "A’ @ B = A @ (C’ @ B)" by simp
  then show ?thesis ..
qed
lemma "extends A A" by auto
lemma "extends A []" by auto
lemma "extends (A @ B) A" by auto
end
```

□

The following list describes symptoms of understanding a mathematical proof as the ability to perform various tasks. These symptoms have been motivated by the understanding of mathematics in general.

Symptom 1: *Fill in simple details of the proof, like explication of notation and definitions.* Understanding implies the ability to answer questions about concepts, their properties, and relations: What is the domain of variables x, y used in the informal proof? What happens when x has a value outside its domain, say $x = 102, y = 10$? For the first formal proof one can query: What is `bool`? What is the relation between `True` or `False` or 0 or 1? What is `list`, `@`, `[]`, etc.? What is `=` and `#` and what properties do these two relations have?

The level of detail in definitions and concepts is different in the two proofs. For example, one can query the definitions and properties of =, #, or *lists* in the Isabelle formal proof, but hardly in the informal proof.

Symptom 2: *Justify other results implicitly used in the proof and inferences.* The property of associativity and the proof by induction are assumed to be known in the informal proof for Lemma 1 above.

Symptom 3: *Give presentations of the proof for different audiences having various degrees of expertise.* Users can be experts in the subject, experts in the area but not in the subject, professional mathematicians, graduate students, undergraduate students, non-mathematicians with interest in the subject, readers of a science magazine, etc. For example, for an expert the proof of Lemma 1 above is too detailed, in fact the lemma itself may be omitted. For a beginner, the detailed proof for the irrationality of $\sqrt{2}$ is suitable, see [30, p. 37]. *A mathematical theory is not to be considered complete until you have made it so clear that you can explain it to the first man whom you meet on the street* says Hilbert.

Symptom 4: *Cast the proof in different terms.* For example, by varying the proportion of natural language and formulae, by varying the level of detail, or by using the language of a different area of mathematics[6]. The irrationality of the golden ratio can be presented from various perspectives, geometrical, algebraic, [30, p. 41–45].

Symptom 5: *Motivate the proof.* Explain why certain notions/constructions are natural, necessary, in contrast with other potential candidates. For example, one may ask what is the natural representation of strings in Isabelle and what are the basic operations with strings [19].

Symptom 6: *Indicate key or novel points in the argument.* The solution of Post's Problem [24, p.237] requires a new ingredient, the priority argument. The argu-

[6] The word language does not only refer to the terminology and notation only, but to the whole "spirit" of an area of mathematics.

ment is highlighted several times in the proof of the Friedberg-Muchnik Theorem presented in [24, p.238].

Symptom 7: *Give natural examples and counter-examples for various notions used in the proof.* Follow the proof of Lemma 1 and justify why $xy \supset x$ for various groups of strings like $x = 10111, y = 10$ or $x = 102, y = 10$.

Symptom 8: *Indicate where certain hypotheses are needed.* The Banach-Tarski Paradox shows how to cut a solid sphere into pieces and then reassemble them without bending, stretching or distorting them to finally obtain two (or ten) solid spheres equal in volume with the original one — the analog of the "Ponzi-type" effect in economics. This is possible because by cutting the sphere into non-measurable pieces one loses information about the initial volume. Is this possible? Solovay [58] proved that if one doesn't use the Axiom of Choice one can construct a set theory in which the Banach-Tarski Paradox is impossible because every set of reals is measurable. However, in the standard set theory with the Axiom of Choice the answer is affirmative.

Symptom 9: *View the proof in a broader context, for example, as a generalisation or adaptation of another proof.* Many results in different areas of mathematics, from theoretical computer science to dynamics, can be seen as some kind of fixed-point construction [31] and, as a consequence, their proof can be phrased in this general type of argument.

Symptom 10: *Discuss interesting generalisations of the proof.* Category theory is one of the important tools for generalisations. Goguen [28] showed that the construction of the minimal Moore automaton can be lifted to a pair of adjunct functors between the category of Moore automata and the category of their behaviours, a more general/deep presentation of minimisation.

Symptom 11: *Discuss interesting modifications of hypotheses and their corresponding modifications of conclusions.* Solovay's result discussed above shows the existence of two set theories, one in which there are non-measurable sets of reals, and another one in which all sets of reals are measurable.

Symptom 12: *Explore alternative proofs.* One correct proof is enough to justify a theorem, but different proofs illustrate the same mathematical phenomenon from different angles. Pythagoras' Theorem has at least 367 essential different proofs [40], Pythagoras' proof, Euclid's proof, algebraic proof, various types of geometric proof, proof by re-arrangement, proof using differential equations, even a proof by an American President, James A. Garfield. There are four different proofs for the completeness of the predicate calculus, leading to four techniques to build models.

Symptom 13: *Discuss analogies between notions involved in the proof, between proofs, between theories, analogies between analogies.* This symptom was discussed by many authors, see for example [51]. The notion of Hilbert space — which evolved into a branch of mathematics [68] — appeared when David Hilbert realised that some important mathematical proofs were structurally the same, so at an appropriate level of generality they could be regarded as the same type of argument. Algorithmic information theory shows that the quantity of information can be equally defined in complexity terms, without using Shannon's probabilistic approach [20, 14].

Symptom 14: *Calculate a quantity used in the proof.* Chaitin's Omega number is a well-defined mathematical real which is random, hence non-computable, and, as a consequence, only finitely many digits of its binary expansion *may* be calculated; in [17] exact values for the initial 40 bits of an Omega number have been calculated offering a new understanding of its uncomputability.

Symptom 15: *Provide an explicit description of an object whose existence is guaranteed by the theorem.* Chaitin's Omega numbers are computably enumerable and (algorithmically) random, but are there other such reals? The answer is negative, every computably enumerable random real is the Omega number of a prefix-free Turing machine [14], a more "concrete" description of a general notion.

Symptom 16: *Provide a diagram or visual argument illustrating the proof.* The diagram used in in [30, p. 49] for illustrating a short proof of Pythagoras' Theorem is very useful in understanding the proof.

Symptom 17: *Identify the main idea of the proof and use it in other contexts.* For example, the standard proof of the irrationality of $\sqrt{2}$, a widely discussed proof, can be easily adapted for infinitely many other reals, $\sqrt{3}$, $\sqrt{5}$, etc., but it fails for π (why?).

Symptom 18: *Apply the theorem in different contexts.* Solovay's Theorem uses "forcing" — a technique invented by Cohen [23] for proving consistency and independence results in set theory — for a different type of problem. In fact, the important results in mathematics re-appear in contexts different from the original one. Group theory [55] sprang from number theory into the theory of algebraic equations, and from geometry, developed as an abstract subject, and has many applications not only in mathematics, but also in physics and chemistry, even in image processing and arts.

Symptom 19: *Recognise the constructive or non-constructive character of a proof.* A constructive proof gives more insight than a non-constructive argument [12]. To illustrate this delicate point we consider the following

Theorem 1. *There exist two irrationals $x, y > 0$ such that x^y is rational.*

Proof. The proof indicates how to 'construct' the reals x and y subject to the conditions of Theorem 1. We distinguish two cases: a) the real $\sqrt{2}^{\sqrt{2}}$ is irrational, b) the real $\sqrt{2}^{\sqrt{2}}$ is rational.
In case a) we choose $x = \sqrt{2}^{\sqrt{2}}, y = \sqrt{2}$; in case b) we choose $x = y = \sqrt{2}$. To verify that our choice is correct we proceed again by cases. In case a) x is irrational by hypothesis, $y = \sqrt{2}$ is well known to be irrational and $x^y = (\sqrt{2}^{\sqrt{2}})^{\sqrt{2}} = \sqrt{2}^2 = 2$. In case b) $x = y = \sqrt{2}$ are irrationals and $x^y = \sqrt{2}^{\sqrt{2}}$ is rational by hypothesis. □

The proof depends on whether $\sqrt{2}^{\sqrt{2}}$ is irrational or not, and for the time being this is an open problem. This proof is not constructive as it doesn't return values for the reals x, y: the proof produces two pairs of reals, exactly one satisfying the requirements of the theorem. The proof above is less non-constructive than the following one [39]:

> *Proof.* Take the equation $x^y = 2$, and let x run through all irrational numbers greater than 1. This gives uncountably many corresponding values of y, which are all different (as x increases y decreases). The conclusion follows as it is not possible for all these values of y to be rational because there are only countably many rationals. □

The second proof gives more information than the first one, as it shows that there are infinitely many pairs satisfying the conditions of the theorem.

Symptom 20: *Program (parts of) the proof in a programming language.* Formalisation of a proof requires full understanding; once formalised, the correctness of the proof can be verified and checked. In the process of formalisation, authors can debug their proof, e.g. they can fix syntactical mistakes, check whether they used the correct definition and description of all symbols, and whether all symbols have been used correctly and consistently. *To me, you understand something only if you can program it. (You, not someone else!) . . . programming something forces you to understand it better, it forces you to really understand it, since you are explaining it* **to a machine** says Chaitin [21, p. xiiii].

The symptoms discussed above are just illustrative for the diversity of meaning of understanding of proof. Even by contrasting our extremely simple proofs for Lemma 1 one can see that the informal proof is more intuitive while the formal proofs are more rigorous. The first formal proof is more compact than the second one, which is closer in spirit to the informal proof. For deeper proofs this divide is sharper: see for example the informal proof of the Kraft-Chaitin Theorem in [14] and the Isabelle proof in [34]. It is worth observing that the Kraft-Chaitin Theorem has two “roles”: one to be executed as an algorithm, the other to be analysed and validated. Previous formalisation efforts focused only on the first part [20]; the work in [19] was directed towards the second.

Informal proofs have many problems (correctness, for example), but also a glorious history of achievements. What are the problems with formal proofs? Some may argue that there is no problem whatsoever. Nelson [46] makes a strong point that syntax is all:

> *As to whether or not a string of formulas is a proof there is no dispute: one simply checks the rules of formation. This is the syntax of mathematics. Is that all there is to mathematics? Yes, and it is enough.*

We believe that *understanding* of formal proofs is the main obstacle. Because of high complexity, most formal proofs cannot be checked by humans, so we can ask with Graham: *If no human being can ever hope to check a proof, is it really a proof?* Bluntly, can we understand formal proofs to the extent they can be used instead of pen-on-paper proofs in the practice of mathematics?

3 An Environment for Correctness and Understanding

Our answer to the last question is emphatically affirmative and is described in the form of an envisioned environment, called *active proof environment* (APE), in which users can write and check formal proofs as well as query them with reference to the symptoms of understanding. In particular, an APE supports the following services:

- discover or verify a formal proof,
- query for theorems and their proofs,
- formalise informal proofs for verification and checking,
- explore proofs at different levels of abstraction and in various forms of presentation,
- support queries corresponding to our symptoms of understanding,
- publish mathematical results at an appropriate level of detail and formality.

Various technologies supporting the above services already exist, but no unique system providing *all* services. An APE includes a *proof assistant* and an *intelligent interface*

Much research is done in the field of proof assistants, such as Mizar [44], Isabelle, Coq [10], or *Ω*MEGA [56], which include libraries of highly interlinked formal proofs and theorems [44, 1], the backbone of our environment.

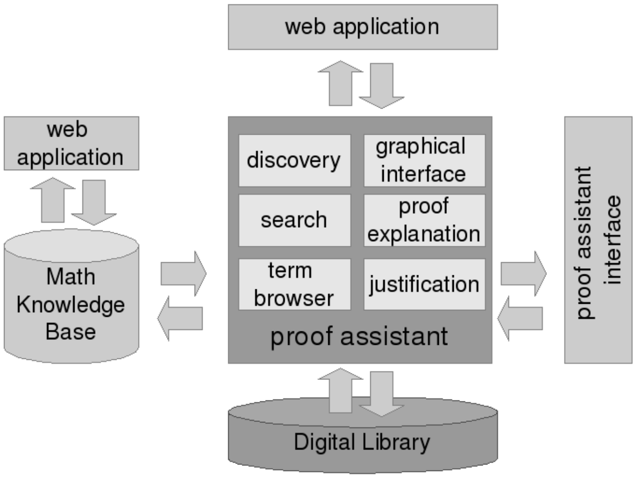

Fig. 1. An Active Proof Environment

Fig. 1 illustrates three alternative intelligent interfaces: proof assistant interfaces, web applications that are integrated with the proof assistant, and web applications that rely on a mathematical knowledge base that communicates with a proof assistant. Proof assistant interface implement some features of an APE's intelligent interface: Isabelle uses the *Proof General* Emacs Interface [2] and the

Intelligible semi-automated reasoning language (Isar) [64] producing structured proof documents. Coq provides a TeXMacs [59] interface for authoring [3] and a MathML integration for publishing mathematics on the web. The mediator PlatΩ [6] connects TeXMacs with Ωmega and allows users to develop, publish, formalise, and query mathematical proofs.

However, the list of symptoms discussed in the previous section shows the necessity of enhancing existing interfaces to implement services for understanding. Web application implement additional features that can support some of our symptoms. Users of the `vdash` wiki [60] can collaboratively formalise mathematical proofs. Initial submissions can be verified or marked as sketches to be fully formalised later. Other mathematical web applications, such as the semantic wiki SWiM [38] and the document reader *panta rhei* [45], are not integrated with proof assistants but work with a central repository of mathematical knowledge represented in OpenMath [50] and OMDoc [35].

In the following we enumerate available services of proof assistants and propose new services, focusing on web-based technologies for OMDoc.

Service 1: *Query different levels of details for a proof.* (Symptom 1)
Authors of formal proofs write *proof scripts*, including instructions and definitions, for the proof assistant with just enough information to generate a formal proof for a given theorem. Users are usually presented with an extract of the fully formalised proof, but can explore the proof on different levels of abstractions (e.g. see the *proof plan* data structure PDS [7, 22] in Ωmega [4, 56] or *high-level proofs* in [13]), or request the system to print all steps, declarations, and definitions.

OMDoc representations can include the fully formalised proof. Folding and elisions of proofs allow one to hide and display different steps and to interactively adapt the level of detail. SWiM includes static cross-links between OMDoc and OpenMath representations of symbols and their definitions.

Service 2: *Query for justification of proving steps.* (Symptom 2)
Proof assistants can automatise the generation of justification for any proving step. OMDoc uses a markup of informal justification that allows users to link manually supplied justification with automatically generated inferences.

Service 3: *Write the proof in different levels of expertise.* (Symptom 3)
Isabelle/HOL generates formal representations of formal proofs for further computation and export the proof into human-readable formats. Ωmega provides a graphical map of the proof tree, a linearised presentation of the proof nodes with their formulae and justifications, a term browser, and a natural language presentation of the proof [26], [57, p. 370]. Furthermore, interactive natural language explanation and justifications of proofs can be generated. PlatΩ supports the presentations of proofs (generated in Ωmega) for different audiences and can adapt the level of detail, the proportion of formal and natural languages, or the mathematical notations [6, 54], as well as interactive stepwise explorations of mathematical proofs [7, 6].

Service 4: *Produce the proof in a specific natural language with different proportions of text and formulae.* (Symptoms 3, 4)

Isabelle is a generic framework for human-readable formal proof documents generated by Isar [63]. PLATΩ supports automatic translations from programming code to natural language [6, 54]. OMDoc includes a multilingual markup for proofs [35]. The conversion from OMDoc to XHTML is based on a collection of XSLT [36] stylesheets, which can be parametrised with the user's preferred language or level of formality.

Service 5: *Query the motivation for the proof.* (Symptom 5)
In OMDoc all fragments of a document are uniquely identified with an *Uniform Resource Identifier* [9] and are classified and interlinked according to mathematical categories [35]. This system allows to interlink fragments of a proof with complementary information, such as a motivation, and can answer corresponding queries. For example, SWiM represents these categories and relations as RDF triples [42] and uses SPARQL [52] to process the queries.

Service 6: *Query novel points.* (Symptom 6)
OMDoc can be extended by rhetorical markup for "novel points" or "obstacle" [27, 35]; XSLT stylesheets can be extended with appropriate visual markers.

Service 7: *Query examples and counter examples.* (Symptom 7)
The *Archive of Formal Proofs* (AFP) [1] is a collection of proof libraries, including examples, formally verified with Isabelle. Isabelle/HOL provides a counter-example search based on Quickcheck [47] and Refute [61]. OMDoc supports the annotation of proofs with examples and counter-examples. SWiM includes static cross-links to examples and dynamically embeds a list of examples into a page.

Service 8: *Query why certain hypotheses are needed and explore consequences if they are changed or omitted.* (Symptoms 8, 11)
Isabelle can be used to experiment and explore consequences of changing various hypotheses.

Service 9: *Produce other proofs that relate to the proof and apply theorems in different contexts.* (Symptoms 9, 10, 18)
Proof assistants organise their mathematical theorems and proofs into contexts (or mathematical theories) and morphisms, which are used to transfer entities from one context to another. Isabelle theories are organised as a graph, so users can apply a theorem in a new context and explore related theorems and proofs.

OMDoc uses the markup of mathematical theories and theory morphisms [35, 53, 49]. Theories are interlinked via theory morphisms, supporting the reuse of previously defined concepts. These logical dependencies build up a theory graph from which one can infer relations between theorems, proofs, or examples.

Service 10: *Query for alternative proofs.* (Symptom 12)
The formal proofs for Lemma 1 use different techniques: the first one recursion, the second one natural deduction. Proof assistants can be used to discover or verify alternative proofs. In OMDoc requests such as "retrieve all proofs for lemma X" can be processed (Service 6).

Service 11: *Query for analogies.* (Symptom 13)
Analogies are very important in mathematics and more work, such as [43], has to be invested to deliver them in APE.

Service 12: *Query for calculations.* (Symptom 14)
ΩMEGA integrates external systems such as *computer algebra systems* (CAS) for symbolic computation; see [5] for an overview of further technologies.

Service 13: *Produce visual illustrations.* (Symptom 16)
Some proof assistant generate diagrams and other visual illustrations [67].

Service 14: *Query for the main idea.* (Symptom 17)
For OMDoc, [27] proposes the markup of "nucleus", a compact presentation of the proof from which the full argument can be easily reconstructed. For example, the nucleus of the second proof of Theorem 1 is "examine the cardinality of the solutions of the equation $x^y = 2$, when x runs through all irrational positive numbers greater than one".

Service 15: *Query whether a proof is constructive or not.* (Symptom 19)
For Coq, which uses constructive logic, this query is easy to answer. In general, partial answers can be obtained by identifying non-constructive rules of inference.

Service 16: *Program parts of the proof.* (Symptom 20)
This symptom is automatically satisfied by formal proofs.

4 Conclusion

The main features of mathematical proof are correctness and understanding. Correctness is easy to define, but there is little consensus regarding the understanding of understanding. To address this, we have proposed a list of symptoms for detecting the understanding of proofs. We have presented a vision of an environment that provides services addressing the symptoms of understanding, in which users can write, check, an query formal proofs. In such an environment, formal proofs are not only theoretical concepts. Because they guarantee a high level of certainty and provide understanding, formal proofs can become the standard of mathematical proof.

In the table below we summarise the technologies that support understanding of formal proofs, which should be integrated into a *unique* system. The proposal is preliminary and needs more extensive experimentation, implementation, and evaluation. Our analysis mainly refers to the proof assistants Isabelle and ΩMEGA, and the OMDoc projects. Our choice doesn't imply any value judgement on technologies.

The second author is developing an active document environment [45] in which users can produce, edit, query their documents, and use the following services:

- configurable layouts and output formats,
- presentations with varying level of detail, expertise, or formality, including multilingual presentations and consistent use of mathematical notations,

Symptom	Services
1	PDS/ Ωmega, High-level proofs, OMDoc
2	proof assistants, OMDoc
3	Isabelle/HOL, PlatΩ/Ωmega
4	Isabelle/Isar, PlatΩ/Ωmega, OMDoc/ XSLT
5	OMDoc/ SWiM/ RDF/ SparQL
6	extension of OMDoc
7	AFP, Isabelle/HOL, OMDoc/ SWiM
8	Isabelle
9	Isabelle, OMDoc
10	Isabelle, OMDoc
11	Isabelle
12	proof assistants, OMDoc (RDF/SparQL)
13	[43]
14	CAS
16	[67]
17	OMDoc
18	Isabelle, OMDoc
19	Coq
20	formal proofs

– enrichment of the documents with definitions, motivations, examples, or justifications and clarification of novel points and main ideas.

Acknowledgements We thank Greg Chaitin, Fulya Horozal, Michael Kohlhase, Christoph Lange, Charles Leedham-Green, Radu Nicolescu, Marc Wagner, Makarius Wenzel for enlightening discussions, suggestions and critical comments. We also thank the anonymous referees for useful suggestions. Our work was supported by the Centre for Discrete Mathematics and Theoretical Computer Science, Auckland, New Zealand. The second author has been partially funded by JEM-Thematic-Network ECP-038208.

References

[1] AFP. Archive of formal proofs. http://afp.sourceforge.net. seen March 2009.

[2] D. Aspinall. Proof general: A generic tool for proof development. In *Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, 38–42. Springer, 2000.

[3] P. Audebaud and L. Rideau. TeXMacs as authoring tool for publication and dissemination of formal developments. In D. Aspinall and C. Lueth, editors, *User-Interface for Theorem Provers. Electronic Notes in Theoretical Computer Science 103*, 27–48, 2004.

[4] S. Autexier, C. Benzmüller, D. Dietrich, A. Meier, and C.-P. Wirth. A generic modular data structure for proof attempts alternating on ideas and granularity. In J. Borwein and W. M. Farmer, editors, *Mathematical Knowledge Management*, *LNAI* 4108, 126–142. Springer, 2006.

[5] S. Autexier, J. Campbell, J. Rubio, V. Sorge, M. Suzuki, and F. Wiedijk, editors. *Intelligent Computer Mathematics*, *LNAI* 5144. Springer, 2008.

[6] S. Autexier, A. Fiedler, T. Neumann, and M. Wagner. Supporting user-defined notations when integrating scientific text-editors with proof assistance systems. In M. Kauers, M. Kerber, R. Miner, and W. Windsteiger, editors. *MKM/Calculemus 2007*, 176–190. *LNAI* 4573. Springer Verlag, 2007.

[7] S. Autexier, C. Benzmüller, D. Dietrich, and M. Wagner. Organisation, transformation, and propagation of mathematical knowledge in Ωmega. *Journal of Mathematics in Computer Science*, 2009. accepted.

[8] J. Avigad. Understanding proofs. In Paolo Mancosu, editor, *The Philosophy of Mathematical Practice*, 317–353. Oxford University Press, 2008.

[9] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, Internet Engineering Task Force, 2005.

[10] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development – Coq'Art: The Calculus of Inductive Constructions*. Springer Verlag, 2004.

[11] N. Bourbaki. *Theory of Sets. Elements of Mathematics.* Springer Verlag, 1968.

[12] D. Bridges and F. Richman. *Varieties of Constructive Mathematics.* Cambridge University Press, 1987.

[13] A. Bundy. A science of reasoning. In J.L. Lassez and G. Plotkin, editors, *Computational Logic – Essays in Honor of A. Robinson*, 178–198. MIT Press, 1989.

[14] C. S. Calude. *Information and Randomness: An Algorithmic Perspective.* Springer Verlag, 2002, 2nd edition.

[15] C. S. Calude, E. Calude, and S. Marcus Passages of proof. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, 84:167–188, 2004.

[16] C. S. Calude, E. Calude, and S. Marcus. Proving and programming. In C. Calude, editor, *Randomness and Complexity, From Leibniz to Chaitin*, 310–321. World Scientific, 2007.

[17] C. S. Calude and M. J. Dinneen. Exact approximations of Omega numbers. *Int. Journal of Bifurcation & Chaos*, 17(6):1937–1954, 2007.

[18] C. S. Calude and S. Marcus. Mathematical proofs at a crossroad? In J. Karhumäki, H. Maurer, G. Păun, and G. Rozenberg, editors, *Theory Is Forever*, *LNCS* 3113, 15–28. Springer-Verlag, 2004.

[19] C. S. Calude and N. J. Hay. Every Computably Enumerable Random Real Is Provably Computably Enumerable Random. Research Report of CDMTCS-328, July 2008.

[20] G. J. Chaitin. *The Limits of Mathematics.* Springer-Verlag, 1998.

[21] G. J. Chaitin. *Meta Math!* Pantheon, 2005.

[22] L. Cheikhrouhou and V. Sorge. PDS – a three-dimensional data structure for proof plans. In *Proceedings of the International Conference on Artificial and Computational Intelligence for Decision, Control and Automation in Engineering and Industrial Applications (ACIDCA'2000)*, 144–149, 2000.

[23] P. J. Cohen. *Set Theory and the Continuum Hypothesis.* Addison-Wesley, 1966.

[24] S. B. Cooper. *Computability Theory.* Chapman &Hall/CRC, 2004.

[25] S. Feferman, J. Dawson Jr., S. C. Kleene, G. H. Moore, R. M. Solovay, J. van Heijenoort, and Daniel J. Velleman, editors. *Kurt Gödel Collected Works.* Oxford University Press, 1986.

[26] A. Fiedler. *User-adaptive Proof Explanation.* PhD Thesis, Universität des Saarlandes, Saarbrücken, Germany, 2001.

[27] J. Giceva. Capturing Rhetorical Aspects in Mathematical Documents using OMDoc and SALT. Technical Report, Jacobs University, Germany, 2008.

[28] J. A. Goguen. Realization is universal. *Theory of Computing Systems*, 6(4):359–374, 1973.

[29] G. Gonthier. Formal proof – The Four-Color Theorem. *Notices of the AMS*, (11):1382–1393, 2008.

[30] T. Gowers. *Mathematics. A Very Short Introduction.* Oxford University Press, 2002.

[31] A. Granas and J. Dugundji. *Fixed Point Theory.* Springer, 2003.

[32] T. C. Hales. Formal proof. *Notices of the AMS*, (11):1370–1380, 2008.

[33] J. Harrison. Formal proof – theory and practice. *Notices of the AMS*, (11):1395–1406, 2008.

[34] N. J. Hay. Formal Proof for the Kraft-Chaitin Theorem. http://www.cs.auckland.ac.nz/~nickjhay/KraftChaitin.thy, seen March 2009.

[35] M. Kohlhase. OMDoc – *An Open Markup Format for Mathematical Documents [Version 1.2]*. *LNAI* 4180. Springer Verlag, 2006.

[36] M. Kohlhase. XSLT Stylesheet for converting OMDoc documents into XHTML. http://kwarc.info/projects/xslt, seen January 2008.

[37] S. M. Lane. Despite physicists, proof is essential in mathematics. *Synthese*, 111(2):147–154, 1997.

[38] C. Lange. SWiM – a semantic wiki for mathematical knowledge management. In Sean Bechhofer, M. Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, editors, *ESWC*, *LNCS* 5021, 832–837. Springer, 2008.

[39] C. Leedham-Green. Personal communication to C. Müller, March 7 2009.

[40] E. S. Loomis. *The Pythagorean Proposition*. Oxford University Press, 1968, 2nd edition.

[41] Y. I. Manin. *Mathematics as Metaphor*. American Mathematical Society, 2007.

[42] F. Manola and E. Miller. RDF Primer. W3C Recommendation, World Wide Web Consortium, February 2004.

[43] E. Melis and J. Whittle. Analogy in inductive theorem proving. *Journal of Automated Reasoning*, 22(2):117–147, 1999.

[44] Mizar. http://web.cs.ualberta.ca/~piotr/Mizar, seen March 2009.

[45] C. Müller and M. Kohlhase. panta rhei. In Hinneburg, editor, *LWA Conference Proceedings*, 318–323, Martin-Luther-University, 2007.

[46] E. Nelson. Syntax and Semantics. http://www.math.princeton.edu/~nelson/papers/s.pdf, seen March 2009.

[47] S. Berghofer and T. Nipkow. Random testing in Isabelle/HOL. In J. Cuellar and Z. Liu, editors, *Software Engineering and Formal Methods (SEFM 2004)*, 230–239. IEEE Computer Society, 2004.

[48] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL – A Proof Assistant for Higher-Order Logic*, *LNCS* 2283. Springer, 2002.

[49] I. Normann. *Theory Morphisms*. PhD Thesis, Jacobs University, Germany, 2008.

[50] OpenMath Home. http://www.openmath.org, seen March 2007.

[51] G. Pólya. *Mathematics and Plausible Reasoning Volume I: Induction and Analogy in Mathematics*. Princeton University Press, 1969.

[52] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/, seen March 2009.

[53] F. Rabe. *Representing Logics and Logic Translations*. PhD Thesis, Jacobs University, Germany, 2008.

[54] M. Schiller, D. Dietrich, and C. Benzmüller. Proof step analysis for proof tutoring – a learning approach to granularity. *Teaching Mathematics and Computer Science*, 2009. in press.

[55] W. R. Scott. *Group Theory*. Dover, New York, 1987.

[56] J. Siekmann, C. Benzmüller, A. Fiedler, A. Meier, I. Normann, and M. Pollet. Proof development in OMEGA: The irrationality of square root of 2. In F. Kamareddine, editor, *Thirty Five Years of Automating Mathematics*, 271–314. Kluwer, 2003.

[57] J. Siekmann, C. Benzmüller, A. Fiedler, A. Meier, and M. Pollet. Proof development with Omega: Sqrt(2) is irrational. In M. Baaz and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 9th International Conference, LPAR 2002*, *LNAI* 2514, 367–387. Springer, 2002.

[58] R. M. Solovay. A model of set-theory in which every set of reals is Lebesgue measurable. *Annals of Mathematics*, 38(3):1–56, 1970.

[59] Gnu TeXMacs. web page at http://www.texM.s.org, seen March 2009.

[60] vdash: A Formal Math Wiki. http://www.vdash.org, seen March 2009.

[61] T. Weber. Bounded model generation for isabelle/hol. *Electronic Notes in Theoretical Computer Science*, 125(3):103 – 116, 2005.

[62] M. Wenzel. Personal communication to C. Müller, March 7 2009.

[63] M. Wenzel. Isabelle/Isar – a generic framework for human-readable proof documents. In R. Matuszewski and A. Zalewska, editors, *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*, 10(23) *Studies in Logic, Grammar and Rhetoric*, 277–298. University of Białystok, 2007.

[64] M. Wenzel. Isar – a generic interpretative approach to readable formal proof documents. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Théry, editors, *Theorem Proving in Higher Order Logics: TPHOLs'99*, *LNCS* 1690, 167–184. Springer Verlag, 1999.

[65] A. N. Whitehead and B. Russell. *Principia Mathematica*, volume I. Cambridge University Press, Cambridge, 1910, 2nd edition.

[66] F. Wiedijk. Formal proof – getting started. *AMS Notices*, (11):1408–1414, 2008.

[67] D. Winterstein, A. Bundy, C. Gurr, and M. Jamnik. An experimental comparison of diagrammatic and algebraic logics. In *International Conference on the Theory and Application of Diagrams*, 2980, 432–434. Springer, 2004.

[68] N. Young. *An Introduction to Hilbert Space*. Cambridge University Press, 1988.