Hilbert II

Presentation of
Formal Correct
Mathematical Knowledge

Basic Concept

Michael Meyling

October 16, 2006

Contents

Executive Summary

The project **Hilbert II** deals with presentation and documentation of mathematical knowledge. Therefore **Hilbert II** supplies a program suite for the realization of the related tasks. Also the documentation of basic mathematical theories is a main purpose of this project.

This document is a service description of the program suite and its main features. This roughly concept should enable a mathematician to understand the vision and the contents of **Hilbert II**.

The goals of this project are as follows.

Formal correct but *readable* mathematical knowledge should be made *freely accessible* in *decentralized* manner within the internet.

- *Formal correct* means checkable by a proof verifier. For this reason the mathematical formulas are written in a formal language that includes a first order predicate calculus. This makes a mechanical analysis possible. For example the enquiry if a theorem depends from a certain axiom could be answered automatically.
- The presentation shall be *readable* like an ordinary mathematical textbook. This means text and common informal proofs. There are even different detail levels possible. One of the most detailed form of a proof is a formal proof.
- Manifestations of these textbooks in LaTeX files or HTML pages are *freely accessible* in the world wide web. It also stands for "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non commercially.
- The knowledge is organized *decentralized* because it is spread over the internet with or without cross references to each other. So already proven theorems could be used elsewhere.

To achieve these objectives the mathematical knowledge is organized in so called *qedeq modules*. Such a module is a XML file that is in principle already structured like a common LaTeX file. It contains LaTeX text for different detail levels, LaTeX templates to display the formal contents and the formal contents itself. The proof checker only address the formal content. Other programs could generate LaTeX and HTML files for given detail levels out of the qedeq modules.

There should be also a *qedeq viewer* that can directly view qedeq modules and switch between the different explanation levels. It can also analyze the dependencies between the theorems and show the derivation of a proposition to its axomatic roots.

This document is already generated out of the following XML file:

http://www.qedeq.org/0_01_07/qedeq_basic_concept.xml.

This is still a “living document” and is updated from time to time. Especially at the locations marked with “+++” additions and improvements are planed.

Preface and *Introduction 1* describe the project background and vision. Chapter **??** gives more details about the functional requirements. Other requirements are listed in chapter *3*. Chapter *⌈4* provides some information about the technichal specifications and software architecture. Last but not least a very rudimentary project plan shows the different development phases.

Preface

This document is the result of a lifelong dream. No more insecurity about the correctness of mathematical proofs. The goal of **Hilbert II** is decentralized access to verified and readable mathematical knowledge. As it's name already suggests, this project is in the tradition of Hilbert's program.

During my mathematical education I found it difficult to balance the detail deepness of my proofs. Sometimes I needed even for simple steps several lemmata. Occasionally my argumentation was too short and from time to time even incorrect.

Once in a while I tried to write down nearly formal proofs. That often had the high danger of not seeing the wood for the trees. Formal proofs kill the mathematical spirit and dry mathematics out into a dead skeleton.[1]

Some parts of this text were written within the great insular landscape of *Amrum*. The sea, the sand and the wind created such an inspirational environment.

But living flesh needs a strong skeleton to give you stability and to make the muscles work. Even if the skeleton is essential it must not be directly visible. So only the combination of lively mathematical texts with absolutely reliable formal background develops the full potential of mathematical knowledge.

I am deeply grateful to my wife *Gesine Dräger* and our son *Lennart* for their support and patience.

Hamburg, August 2005

Michael Meyling

[1] After a text from *Richard Courant*:

> We must not accept the old blasphemous nonsense that the ultimate justification of mathematical science is the "glory of the human mind". Abstraction and generalization are not more vital for mathematics than individuality of phenomena and, before all, not more than inductive intuition. Only the interplay of these forces and their synthesis can keep mathematics alive and prevent its drying out into a dead skeleton.

.

Chapter 1

Introduction

This chapter gives an overview of the project purpose and goals.

1.1 Motivation

Mathematics is a science with a structure that achieved enormous dimensions in the course of time. This huge stronghold has only a small set theoretic foundation and its firmness rests upon simple predicate calculus mortar. In principle the assembly could be comprehended by any mathematician. From every newest turret of mathematical cognition each path of logical dependency could be followed all the way down to the set theoretic roots.

But this is practically impossible. It simply costs too much time to follow every single step in all it's details. Common practice for a mathematician is the use of references to more or less basic theorems that are proved elsewhere. Hopefully all of these referencing chains will end at axioms. The large number of referencing chains together with the experience that even standard works contain mistakes increase the error probability. Furthermore top level results are often verified by few people only.

One must be even more confident that all references match, that every single precondition is fulfilled to apply the theorem. Often preconditions are well hidden, e.g. "note that from this point on it will be assumed that every ring is commutative" as mentioned in the third chapter. This increases the difficulties for a mathematician who crosses the boarder of her discipline or a student to use mathematical results. The understanding can also be aggravated by unknown nomenclature, field specific conventions and definitions and special proof techniques. One has to acquire their meanings and learn their usage. It simply costs a lot of time to be cocksure.

Another aspect is the question of free access to mathematical knowledge. If mathematical textbooks are still buyable their price is high and access to a relevant and nearby library is often limited. But mathematical knowledge belongs to the worldwide cultural assets. This knowledge should be freely available for everybody.

1.2 Gödel's Incompleteness Theorem

A consistent and complete axiomatization of mathematics is impossible. If an recursive axiomatizable theory is sufficiently strong and consistent it will have undecidable sentences.

So there is clearly no hope for a complete formalization of mathematics? In a certain sense this is false! The proof of Gödel's incompleteness theorem could be formulated within a formal language too. One can prove, within this system, that a similar theory[1] is incomplete. Because even incompleteness can be proved within a formal system one could propose that the complete world of mathematics is formalizable.

1.3 Goals

To solve the problem described above, the following demands are made:

- Proposition formulas should be written in a standardized language.
- References should be easily resolvable.
- Theorems should be checkable by a proof verifier.
- Mathematical standard works should be freely accessible.

Students and professional mathematicians could benefit from **Hilbert II**. First of all this project provides a compilation of common mathematical textbooks. These textbooks are available for free and are easily accessible by internet. They come in different formats like LaTeX, PDF and HTML. They are highly linked and enable effortless reference resolution.

Furthermore there will be additional textbooks which contain formal proofs for the theorems. There could also be supplementary texts and documents in other languages.

So you could start with a mathematical theorem and read a short non formal proof. If you are puzzled with that proof there might be a more detailed version and even a formal proof to support your comprehension.

Needless to say **Hilbert II** offers a publishing framework for mathematical texts. Starting with a common LaTeX text file the mathematical contents is transferred step by step into a formal language. In the first phase it is not necessary to provide a formal proof, only a formal notation for formulas is required. The resulting XML file contains theorems and definitions written in a formal language and their LaTeX visualization. An equivalent to the original textbook could be generated. Additionally it is possible to analyze the formulas, even a theorem prover could be attached.

The addition of formal proofs in the second phase might be a little bit painful. In principle a formal proof is a sequence of formulas which follow logically from previous proved theorems or proof lines. The last proof line is equal with the theorem to prove. To make the derivation easily checkable by a proof verifier these steps must be very small. A common mathematical proof technique is the usage of assumptions. The so called *deduction theorem* is a new meta rule. There are many others and the more are understood by the proof checker the easier writing formal proofs gets. See also under *Mathematics 2.1.1*.

There exists a working prototype called *Principia Mathematica II*. It is fully capable of first order predicate logic and shows the main features and basic functionality of **Hilbert II**. It can verify (prototype) qedeq module files located anywhere in the internet. The prototype has a GUI and can transfer qedeq modules into HTML and LaTeX files. You can create and edit your own new qedeq module and publish it in the internet. In the web already existing qedeq modules could be used just by referencing them.

[1] So the formal system analyzes a theory similar to itself

Chapter 2

Functional Specification

The following is a description of what a **Hilbert II** does or should do. A functional specification describes how a product will work entirely from the user's perspective. It doesn't care how the thing is implemented. It just talks about features.

2.1 Functional and Data Requirements

The following contains a specification for each individual functional requirement.

2.1.1 Mathematics

In **Hilbert II** a formal language is used which enables us to describe most domains of mathematics. It is a first order predicate calculus based on the text *Elements of Mathematical Logic* from P. S. Novikov. The logical axioms and basic rules originate from the book *Principles of Mathematical Logic (Grundzüge der theoretischen Logik)* (1928) by D. Hilbert and W. Ackermann.

Beside logical ones the only axioms in **Hilbert II** are those of axiomatic set theory. As usual for mathematics the axioms of all other theories could be expressed as simple predicate constant definitions. The set theoretic axiom system used here is the extended form of Neumann-Bernays-Gödel (extended NBG, also called Morse-Kelley), which fits the needs of the working mathematician.

2.1.2 Qedeq Format

The mathematical knowledge of this project is organized in so called *qedeq* modules. Such a module can be read and edited with a simple text editor. It could contain references to other qedeq modules which lay anywhere in the world wide web.

A qedeq module is built like a mathematical text book. It contains chapters which are composed of paragraphs each with an axiom, abbreviation, definition or proposition. Every paragraph has a label and could be referenced by that label. Essential formal element of a paragraph are formulas. The formulas are written in a first order predicate calculus, also the proofs are in this language. Therefore a proof verifier can check the formulas and their proofs for formal correctness. In this manner linked mathematical text books could be typed which have the extended analytic possibilities of the formal language. Beside the

assured correctness of formulas and proofs there is for example a dependency analyze easily done.

In addition to the basic rules also other derived rules, so called *meta rules*, could be used. A proof that uses meta rules could be automatically transformed into a proof which only uses the basis rules. Some other language extensions, for example abbreviations, are established for shorter writing and convenient argumentation. These extensions can also be automatically removed and transformed into the original system.

We are aware of the fact that this transformation is not in each case practically realizable. For example it is not possible to write down the natural number 1000000000000000 completely in set notation: $\{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}, \{\{\}, \{\{\}\}, \{\{\}, \{\{\}\}\}\}, \ldots\}$.

The comprehension of mathematics is not promoted by formal languages. Hence descriptive texts written in the "colloquial language LaTeX" are of great importance. Lastly those texts carry the mathematical contents for humans. In the qedeq modules of *Hilbert II* those texts are regular parts. There can also be different detail levels of texts and proofs. The first levels should be non formal proofs but common mathematical texts like "trivial", "follows directly from definition" or something more elaborated. Then the highest levels are formal correct proofs. It is also possible to give different proofs, for instance an elegant short one using the foundation axiom and a long and laborious one without the foundation axiom.

Out of the qedeq module hyperlinked LaTeX, HTML or PDF documents can be generated. These documents look basically like a common mathematical document. Before the generation the wanted detail level must be given.

2.2 Use cases

Students and professional mathematicians are the intended audience of **Hilbert II**. This project wants to present mathematical knowledge in formal correct but readable form. In this section the system is described by *use cases*. Such a use case gives an example how the system is going to be used. Each use case has an short name which is written in *italics*.

2.2.1 *REMAPDF* Reading mathematical text.

The user is interested in a certain mathematical subject. With an internet browser she chooses the subject from the **Hilbert II** web page and finds a mathematical textbook in PDF format. After flipping some pages online she saves the document prints it and reads the paper.

2.2.2 *REMAHTML* Reading mathematical text.

In extension to *REMAPDF* the mathematical textbook is visible in HTML format. A fromula shows itself in formal form if the user clicks a certain symbol. It is also possible to change the detail level or text language.

2.2.3 *REMAJAVA* Reading mathematical text.

Similar to *REMAHTML* the browsing is done with an Java pplet or a web started Java program. Some dependency analyzing capablities are included.

2.2.4 *CHECKPRE* Check preconditions for applying.

The user wants to apply a theorem in one of her own proofs. She writes down the preferencees within her proof situation and compares it to those of the theorem visible in *READHTML*.

2.2.5 *TRLATEX* Transformation of LaTeX files.

The user wants to transform her ordinary LaTeX files with mathematical contents into the **Hilbert II** specific qedeq format. She skips through the text files to gather some information about the used mathematical symbols. Text areas that should be transformed into formal language formulas are marked with a specal tag. A translator program is started and transforms the LaTeX files into qedeq modules. The translator program must have access to some information about the used function symbols and their arguments. After some manual corrections the qedeq module files have no syntactical errors. Nevertheless formal proofs for theorems are still missing.

2.2.6 *GENLATEX* Generation of LaTeX files.

The user takes an qedeq module file, e.g. one created with *TRALATEX*, and starts the creation process for a certain language and level. The result is a LaTeX presentation of her qedeq module.

2.2.7 *GENHTML* Generation of HTML files.

The user generates HTML presentation of her qedeq files.

2.2.8 *CHECKTEO* Formal verification of theorem.

The user checks if a theorem is formal correct.

2.3 Non Goals

Although **Hilbert II** is no proof finder in the strong sense it tries to support common mathematical proof techniques.[1]

This means that a very detailed informal proof should be easily transferable into a formal proof that **Hilbert II** accepts. And even one simple step in an mathematical proof could mean hard work for a theorem prover.

The focus lies on simple steps for an *mathematican*. If the step is no problem for an advanced theorem prover but for humans it is not easy to draw the conclusion there is also no need for **Hilbert II** to be able to do that too.

[1] These meta rules could always be replaced by a sequence of simple basic rule applications.

Chapter 3

Other Requirements

Although English is the project language and many mathematicians can read English texts about their special subject **Hilbert II** supports different text languages.

The data of **Hilbert II** can be completely presented in XML documents. The current XML schema specification can be found here:

http://www.qedeq.org/current/xml/qedeq.xsd.

And it's documentation is here:

http://www.qedeq.org/current/xml/qedeq.html.

The data access works with the common internet protocols **http** and **ftp**. This defines platform independence and enables different software implementations. Everybody can implement her own program suite that operates with qedeq modules. So independent proof checkers, document generators, analyzers and so on can be developed. Common interface for all these programs is the XML specification with it's additional semantical restrictions.[1]

The reference software is written in Java and should run on most operating systems.

As time goes by **Hilbert II** will expand. This includes the format of data presentations. The old format must be supported further on.

[1] As there are for example: quantification over already bound variables, unknown references, impoper use of logical laws and so on.

Chapter 4

Technical Specification

This chapter gives some information about the reference implementation. It talks about architecture, data structures, software architecture, algorithms and tools.

4.1 Software architecture

The mathematical knowledge of this project is organized in XML files that are called *qedeq modules*. Such a qedeq module could have references to other qedeq modules which are somewhere in the world wide web. It's main structure looks like an LaTeX book file. There exist a special kind of subsections called *node* that contain an abbreviation, axiom, definition or proposition. Each node is labeled and could be referenced by that label.

The qedeq modules stand under the GNU Free Documentation License (GFDL), the software of this project under the GNU General Public License (GPL). The reference implementation is programmed in Java. Current development environment is eclipse.

4.2 Third party tools and libraries.

The following tools and libraries are used in the development process.

Eclipse	Java IDE
Ant	apache build tool
Xerces	apache XML parser
Checkstyle	coding standard checker
JUnit	a simple framework to write repeatable tests
Clover	code coverage analysis tool

Chapter 5

Project Plan

In contrast to the well developed prototype the main project has not reached alpha stage, but the mathematical grounding of set theory has made good progress. Only the derivation of elementary propositions and definition of necessary notations will be done. The propositions are always written as formulas, the proofs are informal as usual. The outcome of this is a script of axiomatic set theory.

With common mathematical practice in mind, the set theory used in **Hilbert II** is not ZFC but MK (by J. L. Kelley (1955), also called extended NBG).

For the current stage see (text still in German):

http://www.qedeq.org/current/doc/math/mengenlehre_1.pdf.

During the completion of the set theory script the qedeq format will be extended to be suitable for formal correct notations and proofs of that script. The syntax of this formal language should be very near to the common symbolic mathematical language. The script will be translated into this new formal language and will be complemented with formal proofs. After this process an automatic proof verification for the newly created qedeq module is possible. The old informal proofs are also part of the qedeq module and enable a human access to the mathematical contents.

The next major milestone is the release of the version 1.00 which has the following specification:

- The syntax of qedeq module files is so rich, that the notations and formulas of `mengenlehre_1.pdf` could be expressed.
- There is a kernel, which could check qedeq module files on a syntactic basis. For example it should recognize, that a formula is not well formed if it was quantified twice over the same subject variable. The kernel still couldn't check a proof (that means it couldn't decide if a formula derives logically from others).
- The generation of LaTeX files out of qedeq modules is possible.
- The transfer of basic set theory from script into qedeq module files is finished. Starting with the elementary axioms, definitions and notations (as mentioned in `mengenlehre_1.pdf`) the beginning of axiomatic set theory is fully formalized. Ideally the mathematical description texts are written in different detail levels and in the languages English and German.

Index