Comparing mathematical provers

Freek Wiedijk

University of Nijmegen

Abstract. We compare fifteen systems for the formalizations of mathematics with the computer. We present several tables that list various properties of these programs. The three main dimensions on which we compare these systems are: the size of their library, the strength of their logic and their level of automation.

1 Introduction

We realize that many judgments in this paper are rather subjective. We apologize in advance to anyone whose system is misrepresented here. We would like to be notified by e-mail of any errors in this paper at `<freek@cs.kun.nl>`.

1.1 Problem

The QED manifesto [6] describes a future in which all of mathematics is encoded in the computer in such a way that the correctness can be mechanically verified. During the years the same dream has been at the core of various proof checking projects. Examples are the Automath project [17], the Mizar project [15, 23], the NuPRL project [8], and the Theorema project [7]. Recently, the checking of mathematical proofs has become popular in the context of verification of hardware and software: important systems in this area are ACL2 [13] and PVS [18]. Because of this, the field of proof verification currently focuses on computer science applications. The study of formal proof in mathematics is still not widespread.

We have compiled a list of 'state of the art' systems for the formalization of mathematics, systems that one might seriously consider when thinking of implementing the QED dream. We were not so much interested in experimental systems (systems that try out some new idea), as well as in 'industrial strength' systems (systems that are in at least some aspects better at the formalization of mathematics than all other existing systems). We ended up with a list of fifteen systems. For each of these systems we asked a user of the system to formalize the same small theorem: the Pythagorean proof of the irrationality of $\sqrt{2}$.[1] These

[1] This proof is mentioned in Aristotle's *Prior Analytics*, as follows [10]:

> *For all who argue* per impossibile *infer by syllogism a false conclusion, and prove the original conclusion hypothetically when something impossible follows from a contradictory assumption, as, for example, that the diagonal [of*

fifteen formalizations will be published elsewhere.[2] We did not in advance specify to the user a very specific proof problem to be solved, because we wanted formalizations that were the most natural for the given system.

In this paper we compare these fifteen systems according to various criteria. We do not try to establish which of these systems is 'best': they are too different to say something like that. All of these fifteen systems clearly show the dedication of their creators, they all contain important ideas, and they all merit to be studied and used. The main purpose of this paper is to show how different this kind of system can be. When one only knows a few of these systems, it is tempting to think that all systems for formal mathematics have to be of a similar nature. Instead, it is surprising how diverse these systems are.

1.2 Approach

This paper is primarily a collection of tables that show various properties of the systems. It is something like a 'consumer test'.

To illustrate three of the most important dimensions for comparing these systems (the size of their library, the strength of their logic, and their level of automation), at the end of the paper we show them together in a two-dimensional diagram. We realize that this diagram is highly subjective. We list some aspects of the systems that we have used to determine the positions in this diagram. (Readers will probably disagree with the details of this diagram and are encouraged to make their own variant.)

The order in which we list the systems in this paper is the order in which we received the formalizations of the irrationality of $\sqrt{2}$. In this way we wish to express our gratitude for the help of all the people who wrote these formalizations.

1.3 Related Work

On page 1227 of [4] there appears a comparison similar to the one in this paper. However, it only compares nine instead of fifteen systems, and the comparison takes only one and a half pages.

There are several other comparisons between provers in the literature, but those generally compare only two systems, and often compare the systems for applications in computer science instead of for mathematics. For instance, there are comparisons between NuPRL and Nqthm [5], HOL and Isabelle [2], HOL and ALF [1], Coq and HOL [12], and HOL and PVS [11].

---

a square] is incommensurable [with the side] because odd numbers are equal to even if it is assumed to be commensurate.

It was interpolated in Euclid's *Elements* as Proposition x. 117. Due to the oral tradition of the Pythagorean School, the origins of this proof have been covered in 'complete darkness' [22]. There is a legend that Pythagoras' pupil Hippasus discovered this proof and was drowned at sea to keep it a secret.

[2] The current draft of this document is on the Web at `<http://www.cs.kun.nl/~freek/comparison/comparison.ps.gz>`

There also has been work done on how to embed the proofs of one system in another one. As an example, there is the work by Doug Howe and others on how to translate HOL proofs to classical NuPRL [16]. Surprisingly, mapping mathematics between systems is more difficult than one would expect.

1.4 Outline

In Section 2 we list the fifteen systems that are compared in this paper. We explain why we selected these systems and also why we did not select some other systems. In Section 3 we investigate various proof representations that are used by the fifteen systems. We give a classification of these representations into seven categories. Then we investigate the sizes of the irrationality of $\sqrt{2}$ proofs. Finally we compare the sizes of the libraries of the systems. In Section 4 we compare the logical foundations of the systems. We also look at their architecture according to the so-called 'de Bruijn criterion'. In Section 5 we compare the level of automation of the systems. We study whether they satisfy the 'Poincaré principle', whether they have an open architecture that allows user automation, and whether they come with strong built-in automation. Finally in Section 6 we put all systems in one diagram.

2 From Alfa to Ωmega: the fifteen provers of the world

The systems that are compared in this paper are listed in the following table:

1. **HOL**
   Web page: `<http://www.cl.cam.ac.uk/Research/HVG/HOL/>`
   Implementation language: ML
   Main person behind the system: Mike Gordon
   People who did $\sqrt{2} \notin \mathbb{Q}$*:* John Harrison, Konrad Slind
2. **Mizar**
   Web page: `<http://mizar.org/>`
   Implementation language: Pascal
   Main person behind the system: Andrzej Trybulec
   Person who did $\sqrt{2} \notin \mathbb{Q}$*:* Andrzej Trybulec
3. **PVS**
   Web page: `<http://pvs.csl.sri.com/>`
   Implementation languages: Lisp, ML
   Main people behind the system: John Rushby, Natarajan Shankar, Sam Owre
   People who did $\sqrt{2} \notin \mathbb{Q}$*:* Bart Jacobs, John Rushby
4. **Coq**
   Web page: `<http://pauillac.inria.fr/coq/>`
   Implementation language: ML
   Main people behind the system: Gérard Huet, Thierry Coquand, Christine Paulin
   Person who did $\sqrt{2} \notin \mathbb{Q}$*:* Laurent Théry

5. **Otter/Ivy**
   Web pages: <http://www.mcs.anl.gov/AR/otter/> and <http://www-unix.mcs.anl.gov/~mccune/acl2/ivy/>
   Implementation language: C
   Main people behind the system: William McCune, Larry Wos, Olga Shumsky
   People who did $\sqrt{2} \notin \mathbb{Q}$*:* Michael Beeson, William McCune
6. **Isabelle/Isar**
   Web pages: <http://www.cl.cam.ac.uk/Research/HVG/Isabelle/> and <http://isabelle.in.tum.de/>
   Implementation language: ML
   Main people behind the system: Larry Paulson, Tobias Nipkow, Markus Wenzel
   People who did $\sqrt{2} \notin \mathbb{Q}$*:* Markus Wenzel, Larry Paulson
7. **Alfa/Agda**
   Web page: <http://www.cs.chalmers.se/~catarina/agda/> and <http://www.math.chalmers.se/~hallgren/Alfa/>
   Implementation language: Haskell
   Main people behind the system: Thierry Coquand, Catarina Coquand, Thomas Hallgren
   Person who did $\sqrt{2} \notin \mathbb{Q}$*:* Thierry Coquand
8. **ACL2**
   Web page: <http://www.cs.utexas.edu/users/moore/acl2/>
   Implementation language: Lisp
   Main person behind the system: J Strother Moore
   Person who did $\sqrt{2} \notin \mathbb{Q}$*:* Ruben Gamboa
9. **PhoX**
   Web page: <http://lama-d134.univ-savoie.fr/sitelama/Membres/pages_web/RAFFALLI/phox.html>
   Implementation language: ML
   Main person behind the system: Christophe Raffalli
   People who did $\sqrt{2} \notin \mathbb{Q}$*:* Christophe Raffalli, Paule Rozière
10. **IMPS**
    Web page: <http://imps.mcmaster.ca/>
    Implementation language: Lisp
    Main people behind the system: William Farmer, Joshua Guttman, Javier Thayer
    Person who did $\sqrt{2} \notin \mathbb{Q}$*:* William Farmer
11. **Metamath**
    Web page: <http://metamath.org/>
    Implementation language: C
    Main person behind the system: Norman Megill
    Person who did $\sqrt{2} \notin \mathbb{Q}$*:* Norman Megill

12. **Theorema**
    Web page: <http://www.theorema.org/>
    Implementation language: Mathematica
    Main person behind the system: Bruno Buchberger
    People who did $\sqrt{2} \notin \mathbb{Q}$*:* Markus Rosenkranz, Tudor Jebelean, Bruno Buchberger
13. **Lego**
    Web page: <http://www.dcs.ed.ac.uk/home/lego/>
    Implementation language: ML
    Main person behind the system: Randy Pollack
    Person who did $\sqrt{2} \notin \mathbb{Q}$*:* Conor McBride
14. **NuPRL**
    Web page: <http://www.cs.cornell.edu/Info/Projects/NuPrl/nuprl.html>
    Implementation languages: ML, Lisp
    Main person behind the system: Robert Constable
    Person who did $\sqrt{2} \notin \mathbb{Q}$*:* Paul Jackson
15. **Ωmega**
    Web page: <http://www.ags.uni-sb.de/~omega/>
    Implementation language: Lisp
    Main person behind the system: Jörg Siekmann
    People who did $\sqrt{2} \notin \mathbb{Q}$*:* Christoph Benzmüller, Armin Fiedler, Andreas Meier, Martin Pollet

For most systems it is clear why they are in this list, but a few need explanation.

Otter is not designed for the development of a structured body of mathematics in the QED style, but instead is used in a 'one shot' way to solve logical puzzles. Also it is only one of the members (although the best known) of the large class of *first order theorem provers*. The reason that we still have included Otter in this list (and only Otter) is that Art Quaife has used Otter to develop a body of mathematics in Euclidean geometry and set theory [20]. Also Otter is the only program in the list that has been used for the solution of open mathematical problems, as listed in <http://www-unix.mcs.anl.gov/AR/new_results/>. A reason to single out Otter from the first order provers is that Otter has the Ivy program for separate checking of its proofs.

ACL2 is not primarily designed for mathematics. In particular it has a rather weak logic, without an explicit existential quantifier. However the Nqthm system (a predecessor of ACL2, which is very similar) has been used to formalize significant theorems like Gödels first incompleteness theorem [21]. Also Nqthm was the system that the authors of the QED manifesto had in mind.

The Metamath system [14] maybe should not be counted as an 'industrial strength' system: it only has one user. However, the system is beautifully executed and differs in many respects from the other systems. For one thing it is very fast: it can check its full (non-trivial) library in only a few seconds. Also it really makes the logical structure of the mathematics completely transparent.

Some of the systems in the list have predecessors or (recent) successors:

ALF, Half	→	**Agda**
Nqthm	→	**ACL2**
Imps	→	MathScheme
Lego	→	Oleg, Plastic
NuPRL	→	MetaPRL

We did not include any of those. We expect systems from the same origin to be reasonably close to each other.

Some recent provers, like the KIV system and the 'B method', have been especially designed for verification of hardware and software. These systems can also be used for the formalization of mathematics, but we have not included them in our comparison. Other systems, like the Twelf system and the Typelab system, are more for formalizing logic than for mathematics. After some discussion with the authors of these systems we decided to omit these as well. Finally there is the TPS system which is similar to Otter but for higher order logic. However, it misses what makes Otter interesting for this comparison, so it was left out too.

3 Files containing mathematics

3.1 Seven ways to represent mathematics

When we were editing the fifteen proofs of the irrationality of $\sqrt{2}$ for presentation on paper, it turned out that often it was difficult to present the proofs in such a way that it was clear what was going on. For various systems we needed to show the proof in multiple representations. Some reflection showed that these representations could be divided into seven groups:

	HOL	Mizar	PVS	Coq	Otter/Ivy	Isabelle/Isar	Alfa/Agda	ACL2	PhoX	IMPS	Metamath	Theorema	Lego	NuPRL	Ωmega
definitions & statements of lemmas	*i*	*i*	*i*	*i*	*i*	*i*	*i*	*i*	*i*	*i*	*i*	*x*	*i*	*i*	*i*
proof scripts	=	=	*i*	=		=	=		=	=	=		=	*i*	*i*
trace of interactive session	*o*		*o*	*o*	*o*	*o*		*o*			*o*	=	*o*	*o*	*o*
representation with symbols						=	=		*x*		*o*	=		=	*o*
representation in natural language		=						=	=			=			*o*
stored formalization state			*o*											*o*	*o*
λ-term or other 'proof object'				*o*	*o*	*o*	=						*o*		=

In this table an '*i*' means 'input file' that has been made by the user. A '*o*' means 'output file' that has been generated by the system. An '*x*' means that it is a mixture of input from the user and output from the system. An '=' sign means that this is part of the same file as the item above it in the same column. For instance, in the Mizar system the statement of the lemmas, the 'proof scripts' that the user enters, and the 'natural language' representation, all are in one file (the `.miz` file) which is written by the user.

In this table we only included files that can be displayed on paper. For some systems the stored formalization state is not represented in this table because it is a binary file which is not humanly readable.

3.2 Comparing the sizes of the input files

We now compare the sizes of the formalization of the irrationality of $\sqrt{2}$ in the fifteen systems. This not only compares the systems but also the styles of the people who did the formalization and the complexity of the proof that they selected for formalization. Therefore it only gives a rough indication of the 'compactness' of the systems.

Also, comparing systems based on a single proof problem is statistically not meaningful. Having the same proof in fifteen systems does show the proof styles of the systems surprisingly well, but it does not say much about the quality of the provers. Still, we will list the sizes of the files.

Here are the precise statements that were proved (or, in the case of Otter, disproved) in the fifteen systems:

HOL	`~rational(sqrt(&2))`
Mizar	`sqrt 2 is irrational`
PVS	`NOT Rational?(sqrt(2))`
Coq	`(irrational (sqrt (S (S O))))`
Otter	`m(a,a) = m(2,m(b,b))`
Isabelle/Isar	$sqrt\ (real\ (2{::}nat)) \notin \mathbb{Q}$
Alfa/Agda	$prime\, p \rightarrow noether\, A\ (multiple\, p) \rightarrow isNotSquare\, p$

ACL2	`(implies (equal (* x x) 2)`
	`(and (realp x)`
	`(not (rationalp x))))`

PhoX	`/\m,n : N (m^ N2 = N2 * n^ N2 -> m = N0 & n = N0)`
IMPS	`not #(sqrt(2),qq)`
Metamath	`$p |- ( sqr ‘ 2_10 ) e/ QQ`
Theorema	$\neg \mathrm{rat}\left[\sqrt{p}\right]$

Lego	`{b|nat}{a|nat}`
	`(Eq (times two (times a a)) (times b b))->`
	`(Eq a zero /\ Eq b zero)`

NuPRL	$\neg(\exists u{:}\mathbb{Q}.\ u *_q u = 2 / 1)$
Ωmega	`(not (rat (sqrt 2)))`

Some people proved the statement of the irrationality in the real numbers:

$$\sqrt{2} \notin \mathbb{Q}$$

Others did not have a library of real numbers (or did not want to use it) and only proved a statement about natural numbers:

$$m^2 = 2n^2 \iff m = n = 0$$

The Agda proof by Thierry Coquand proves something still more basic. It does not talk about the number two in the natural numbers, but instead about any element in a commutative monoid that satisfies some conditions. The Otter proof has a similar structure.

Most people proved the irrationality of the square root of two, but some only proved the irrationality of an arbitrary prime number (where 'prime' means that the number divides a product if and only if it divides one of the factors). This might sound stronger, but Conor McBride noted that it is the other way around: it is also needs some non-trivial work to prove that two is prime.

Most people proved the statement using the library of their system, but not all systems have a library. In that case some lemmas were proved from statements that were taken as axioms. The Ωmega system does have a standard library, but not all statements in this library have been proved using the system. For the irrationality of $\sqrt{2}$ four lemmas were added to this library, but only one was proved. The Agda system does not have a library, but the formalization did not use any unproved statements. It defined everything that was used, including the logic.

The IMPS proof is by far the largest in the collection. It could have been quite a bit shorter, but William Farmer chose to first prove the more general statement that the square root of any non-square number is irrational.

	lines			*fragment*
Otter	17	monoid	prime	one lemma
HOL	29	$\mathbb{R}$	2	from library
Ωmega	38*	$\mathbb{R}$	2	from library, one lemma out of four
Theorema	39*	$\mathbb{R}$	prime	two lemmas
Mizar	44	$\mathbb{R}$	2	from library
NuPRL	54*	$\mathbb{N}$	2	from library
Coq	68	$\mathbb{R}$	2	from library
PVS	77	$\mathbb{R}$	2	from library
Metamath	81	$\mathbb{R}$	2	from library
Isabelle	114	$\mathbb{R}$	2	from library
PhoX	151	$\mathbb{N}$	2	from library
ACL2	206	$\mathbb{R}$	2	from library
Agda	230	monoid	prime	stand alone
Lego	261	$\mathbb{N}$	2	from library
IMPS	663	$\mathbb{R}$	2	from library

For this table we only counted non-blank non-comment lines.[3] The line counts that have been marked with an asterisk do not refer to a specific file, but instead are combined counts of relevant parts of files. If we restrict ourselves to formalizations that prove the full statement about the number two in the real numbers

[3] The relevant files are on the Web in `<http://www.cs.kun.nl/~freek/comparison/comparison.tar.gz>`

without omitting anything, then the three shortest proofs are the HOL, Mizar and Coq proofs.

3.3 The library

In practice for serious formalization of mathematics a good library is more important than a user friendly system. The Mizar systems has the largest library by far. It proves over 32 thousand lemmas, taking 50 megabytes or 1.4 million lines.

The following systems currently have a large *mathematical* library:

	HOL	Mizar	PVS	Coq	Otter/Ivy	Isabelle/Isar	Alfa/Agda	ACL2	PhoX	IMPS	Metamath	Theorema	Lego	NuPRL	Ωmega
large mathematical library	•	•	•	•		•				•				•	

4 Differences in logical strength

4.1 Logics and type systems

The systems vary in underlying logic and type system. Here is a table that shows the different logics of the fifteen systems:

	HOL	Mizar	PVS	Coq	Otter/Ivy	Isabelle/Isar	Alfa/Agda	ACL2	PhoX	IMPS	Metamath	Theorema	Lego	NuPRL	Ωmega
primitive recursive arithmetic								•							
first order logic					•										
higher order logic	•		•			•	•		•	•		•	•		•
first order set theory		•				•					•				
higher order type theory				•										•	
classical logic	•	•	•		•	•		•	•	•	•	•			•
constructive logic				•			•						•	•	
quantum logic											•				
fixed logic	•	•	•	•	•		•	•	•	•		•	•	•	•
logical framework						•					•				

A logical framework does not just support some given logics, but instead the user is able to define logics of his own. In the case of a logical framework the

first two sections of the table indicate the most commonly used logics of the system.

Coq and NuPRL are implementations of variants of intuitionistic type theory. The 'classical variants' of these systems are equiconsistent with ZFC set theory with countably many inaccessible cardinals.[4] In particular, they are quite a bit stronger than the higher order logics of systems like HOL.

The Mizar system is logically even stronger than this, because it has arbitrarily large inaccessibles. However, Mizar is clearly a first order system.[5] The Coq and NuPRL systems are higher order systems.[6]

Here is a table that shows the type systems of the fifteen systems (a system is only considered typed when the types are first class objects that occur in variable declarations and quantifiers):

	HOL	Mizar	PVS	Coq	Otter/Ivy	Isabelle/Isar	Alfa/Agda	ACL2	PhoX	IMPS	Metamath	Theorema	Lego	NuPRL	Ωmega
untyped					•			•			•	•			
decidable non-dependent types	•					•			•	•					•
decidable dependent types		•		•			•						•		
undecidable dependent types			•											•	

4.2 The de Bruijn criterion

	HOL	Mizar	PVS	Coq	Otter/Ivy	Isabelle/Isar	Alfa/Agda	ACL2	PhoX	IMPS	Metamath	Theorema	Lego	NuPRL	Ωmega
de Bruijn criterion	•			•	•	•	•		•		•		•		•

The de Bruijn criterion states that the correctness of the mathematics in the system should be guaranteed by a *small* checker. Architecturally this generally means that there is a 'proof kernel' that all the mathematics is filtered through. In the HOL Light variant of the HOL system this kernel is extremely small: it

[4] In the NuPRL system the classical variant is officially supported. In the Coq system the equiconsistency actually seems to break down, because the impredicativity of Coq is inconsistent with classical mathematics.

[5] Steps from Mizar proofs correspond directly to first order problems [9]. As part of his PhD research, Josef Urban from the Charles University in Prague is developing software that can export any Mizar step in TPTP format.

[6] The NuPRL type theory is predicative, which means that it does not have just one type for propositions, but one for each type universe. However, in practice NuPRL is higher order logic: it can abstract and quantify over arbitrary higher order types.

consists of only 285 lines of ML. The NuPRL system is just over the border of this criterion. It has a proof checking kernel but this kernel is not small.

The Otter system does not have a proof checking kernel built into the system. However, there is the Ivy system that can export Otter proofs in a form that can be checked by a very small ACL2 program.

5 Proof checking or theorem proving

5.1 Interaction styles

Among the systems one finds three different interaction styles. First, there are the systems in which the user writes the text of the proof and the system checks the correctness afterwards. Second, there are the 'proof assistants' which keep a proof state for the user. The user then modifies this proof state through the application of so-called 'tactics'. Third, there are the automated theorem provers which automatically prove lemmas that the user states. The involvement of the user then only consists of the selection of the lemmas (as 'stepping stones' towards the final result) and the selection of parameters for the prover.

	HOL	Mizar	PVS	Coq	Otter/Ivy	Isabelle/Isar	Alfa/Agda	ACL2	PhoX	IMPS	Metamath	Theorema	Lego	NuPRL	Ωmega
proof checking		•					•								
goal transformation through tactics	•		•	•		•			•	•	•		•	•	•
automated theorem proving					•			•				•			

Note that tactic-based provers can still have powerful automation. For instance the PVS system has powerful decision procedures.

5.2 The Poincaré principle and automation

	HOL	Mizar	PVS	Coq	Otter/Ivy	Isabelle/Isar	Alfa/Agda	ACL2	PhoX	IMPS	Metamath	Theorema	Lego	NuPRL	Ωmega
Poincaré principle	•		•	•	•	•		•	•	•		•	•	•	•
user automation	•		•	•		•								•	•
powerful built-in automation	•		•		•	•		•		•		•			•

An important aspect of a mathematical system is automation of trivial tasks. In particular a user should not need to spell out calculations in detail. A system that can prove the correctness of calculations automatically is said to satisfy the

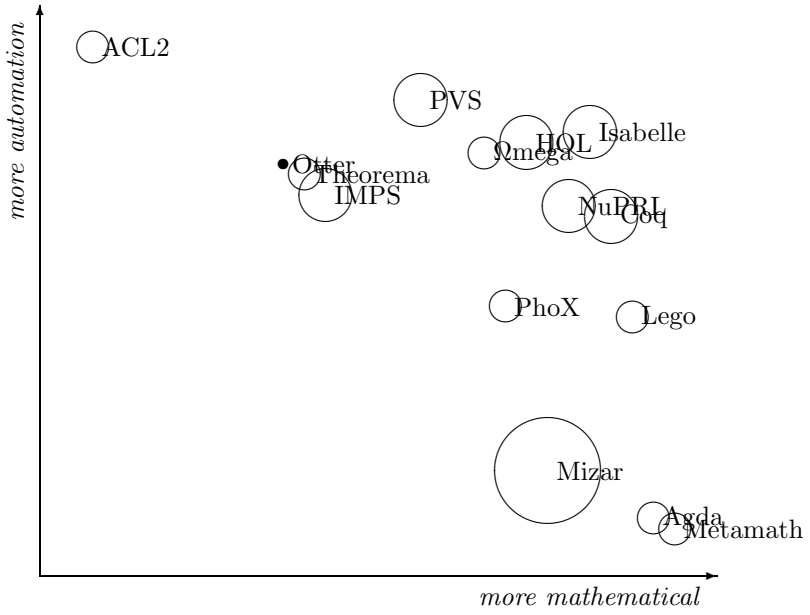
Poincaré principle [3],[7] because in [19] Henri Poincaré wrote about showing the correctness of the calculation $2 + 2 = 4$:

> *'Ce n'est pas une démonstration proprement dite,* [...] *c'est une vérification'.* [...] *La vérification diffère précisément de la véritable démonstration, parce qu'elle est purement analytique et parce qu'elle est stérile.*

An important aspect of a prover is whether it has an 'open' architecture, i.e., whether the user can write programs to solve proof problems algorithmically. Most provers allow *some* automation like this, but in the table we indicate whether this programmability is on the level of the implementation of the system.

Finally there is the aspect whether a prover already has strong automation built-in.[8] Examples of such automation are decision procedures for algebraic problems, proof search procedures, and automation of induction.

6 A rather subjective two-dimensional diagram



[7] Henk Barendregt links the Poincaré principle to reduction in type theory, but this is only partially correct. The Agda system has $\beta\delta\iota$-reduction but it cannot use it for reflection as it lacks the automation to lift the expressions to the syntactical level, so we claim that Agda does not satisfy the Poincaré principle. On the other hand HOL does not have reduction in its logic, but it is easy to program HOL 'conversions' to prove calculations automatically.

[8] We only considered a tactic-based prover to have 'powerful built-in automation' if it has a tactic that is a full first order prover, like Isabelle's *blast*.

We compiled some of the information about the systems into a diagram (like the Herzsprung-Russell diagram classifying stars in astronomy). On the horizontal axis the diagram shows how 'mathematical' the logic of the system is. On the vertical axis it shows how much automation the system offers. The sizes of the circles correspond to the sizes of the respective mathematical libraries.

The positions of the systems in this diagram are rather subjective. Most circles can be argued around quite a bit. To make the diagram somewhat objective we have 'scored' various aspects of the systems. This made the diagram turn out to be surprising to us: we had expected the Theorema and IMPS systems to score more mathematical than they do in this diagram, and the Otter system to score higher on the automation axis.

Items that added to the score for 'more mathematical' were:

- more powerful logic
- logical framework
- dependent types
- de Bruijn criterion

while items that added to the score for 'more automation' were:

- more automated interaction style
- Poincaré principle
- user automation
- powerful built-in automation

7 Future work

The comparison in this paper does not focus in detail on the automation of the systems. It is worthwhile to investigate the various kinds of automation of these systems in more detail, and in particular to investigate their algebraic decision procedures.

References

1. S. Agerholm, I. Beylin, and P. Dybjer. A Comparison of HOL and ALF Formalizations of a Categorical Coherence Theorem. In *TPHOLs'96*, volume 1125 of *LNCS*, pages 17–32. Springer-Verlag, 1996.
2. S. Agerholm and M.J.C. Gordon. Experiments with ZF Set Theory in HOL and Isabelle. In *8th International Workshop on Higher Order Logic Theorem Proving and its Applications*, volume 971, pages 32–45. Springer-Verlag, 1995.
3. Henk Barendregt. The impact of the lambda calculus. *Bulletin of Symbolic Logic*, 3(2), 1997.
4. Henk Barendregt and Herman Geuvers. Proof-Assistants Using Dependent Type Systems. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers B.V., 2001.
5. D. Basin and M. Kaufmann. The Boyer-Moore Prover and NuPRL: An experimental comparison. In *Proceedings of the First Workshop on 'Logical Frameworks', Antibes, France*, pages 89–119. Cambridge University Press, 1991.

6. R. Boyer et al. The QED Manifesto. In A. Bundy, editor, *Automated Deduction – CADE 12*, volume 814 of *LNAI*, pages 238–251. Springer-Verlag, 1994. <http://www.cs.kun.nl/~freek/qed/qed.ps.gz>.
7. B. Buchberger, T. Jebelean, F. Kriftner, M. Marin, and D. Vasaru. An Overview on the Theorema project. In W. Kuechlin, editor, *Proceedings of ISSAC'97 (International Symposium on Symbolic and Algebraic Computation)*, Maui, Hawaii, 1997. ACM Press.
8. Robert L. Constable, Stuart F. Allen, H.M. Bromley, W.R. Cleaveland, J.F. Cremer, R.W. Harper, Douglas J. Howe, T.B. Knoblock, N.P. Mendler, P. Panangaden, James T. Sasaki, and Scott F. Smith. *Implementing Mathematics with the Nuprl Development System*. Prentice-Hall, NJ, 1986.
9. Ingo Dahn and Christoph Wernhard. First Order Proof Problems Extracted from an Article in the MIZAR Mathematical Library. In *Proceedings of the International Workshop on First order Theorem Proving*, number 97-50 in RISC-Linz Report Series, pages 58–62, Linz, 1997. Johannes Kepler Universität.
10. G.P. Goold, editor. *Selections illustrating the history of Greek mathematics, with an English translation by Ivor Thomas*. Harvard University Press, London, 1939.
11. David Griffioen and Marieke Huisman. A comparison of PVS and Isabelle/HOL. In Jim Grundy and Malcolm Newey, editors, *Theorem Proving in Higher Order Logics: 11th International Conference, TPHOLs'98*, volume 1479 of *LNCS*, pages 123–142. Springer-Verlag, 1998.
12. L. Jakubiec, S. Coupet-Grimal, and P. Curzon. A Comparison of the Coq and HOL Proof Systems for Specifying Hardware. In E. Gunter and A. Felty, editors, *International Conference on Theorem Proving in Higher Order Logics: B-Track*, pages 63–78, 1997.
13. Matt Kaufmann, Panagiotis Manolios, and J. Strother Moore. *Computer-Aided Reasoning: An Approach*. Kluwer Academic Publishers, Boston, 2000.
14. Norman D. Megill. Metamath, A Computer Language for Pure Mathematics. <http://metamath.org/>, 1997.
15. M. Muzalewski. *An Outline of PC Mizar*. Fondation Philippe le Hodey, Brussels, 1993. <http://www.cs.kun.nl/~freek/mizar/mizarmanual.ps.gz>.
16. P. Naumov, M.-O. Stehr, and J. Meseguer. The HOL/NuPRL Proof Translator: A Practical Approach to Formal Interoperability. In R.J. Boulton and P.B. Jackson, editors, *The 14th International Conference on Theorem Proving in Higher Order Logics*, volume 2152 of *LNCS*, pages 329–345. Springer-Verlag, 2001.
17. R.P. Nederpelt, J.H. Geuvers, and R.C. de Vrijer. *Selected Papers on Automath*, volume 133 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science, Amsterdam, 1994.
18. S. Owre, J. Rushby, and N. Shankar. PVS: A prototype verification system. In D. Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *LNAI*, pages 748–752, Berlin, Heidelberg, New York, 1992. Springer-Verlag.
19. Henri Poincaré. *La Science et l'Hypothèse*. Flammarion, Paris, 1902.
20. A. Quaife. *Automated Development of Fundamental Mathematical Theories*. Kluwer Academic, 1992.
21. N. Shankar. *Metamathematics, Machines and Gödel's Proof*. Number 38 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1994.
22. Otto Töplitz. *The Calculus: A Genetic Approach*. University of Chicago Press, Chicago, 1963. Translated by Luise Lange.
23. F. Wiedijk. Mizar: An Impression. <http://www.cs.kun.nl/~freek/mizar/mizarintro.ps.gz>, 1999.