Information technology implications for mathematics—a view from the French riviera

Marco Maggesi and Carlos Simpson
Laboratoire J. A. Dieudonné, CNRS UMR 6621 Université de Nice-Sophia Antipolis

...la traduction en langage formalisé ne serait plus qu'un exercise de patience (sans doutes fort pénible).
—Nicolas Bourbaki

Hanc marginis exiguitas non caperet.
—Pierre de Fermat

We feel that this context has now changed. For a long time, "mathematics" and "computer science" have designated widely separated disciplines, overlapping only in areas which were considered "fringe" on either side. This was not really a historical view since Von Neumann, Turing and others were mathematicians who set off into the world of computers. We are now entering a time of reconvergence which has all kinds of implications. A prominent one is the increasing reality of computer proof verification as applied to standard mathematical work. This has been under development for quite a while, but just now it is becoming realistic to imagine applying it to a wide array of mathematical situations.

The idea is to write mathematical documents in a language such that the mathematical accuracy is then verified by a computer program. This should be compared with other different but related utilisations of the computer:
—typesetting: we are looking at having the computer understand the argument, not just the visual expression on the page, nevertheless on a sociological level there are many similarities with the TEXproject;
—computer algebra and other calculation systems: we are looking at the logical or "proof" aspect of mathematics, including also the problem of introducing, defining and manipulating new abstract concepts, which is different from just having the computer do a specific calculation, but of course calculation is one primordial aspect of mathematical proof and one of the big problems is how to integrate the two into a coherent whole;
—computer proof search: we are not, as a basic requirement, asking the computer to *find* the proof of something, only to verify the proof that the mathematician gives. One of our favorite reactions from a colleague (who shall remain nameless) was:

so, how is your project of trivializing mathematics going?

But we are not talking about trivializing mathematics: the mathematician has to supply the proof, because automation can perhaps deal with what we regard as small details but will never (?) replace the human work of concieving the outline of a complicated argument.

A favorite analogy is with the construction trade. At first, everything was constructed by laying stone or other material by hand. If you wanted to be in the construction trade you had to know how to lay a stone wall that was straight (not easy). Nowdays we have new materials and new tools. If you want to be in the construction trade, you should know how to drive a tractor and how to use a circular saw, but you don't really need to know how to build a stone wall unless you specialize in that sort of thing. When we look back at the pyramids we say "Wow! I can't believe they actually built those things without a crane!". In a few hundred years, people will look back at Fermat's last theorem and say "Wow! I can't believe they actually proved that thing by hand!".

Just as the advent of new techniques in the construction industry radically altered the landscape of buildings we build, so the advent of new tools for doing mathematics will radically alter the type of mathematics we find, and in ways we can only barely imagine right now. It seems important that mathematicians join the fray so as to open up our understanding of what the future might hold.

Currently many different *systems* are under experimentation: a "system" is a program which verifies proofs written in a specified language, plus all of the necessary support material including documentation for the language and how to prove things in it; a library of results on which users can build; active support in the sense of continually improving the program to take into account problems which may be encountered or implement new functionalities, together with maintenance of backward-compatibility for the library (this all requires a non-negligable amount of computer science); and a group of users extending beyond the immediate circle of those who concieved and maintained the program.

This subject has recently made the news in several different venues, even an article in the New York Times [Cha04]. One of the more notable efforts at getting the mathematical community involved in the subject is Thomas Hales' "flyspeck project" [Hal].

We will look at the current state of the art with respect to a few of the main systems which are currently available: ACL2, Coq, Isabelle, Mizar, NuPRL, ... and also briefly consider some other projects which are newer or present original aspects such as PVS, Phox, MetaMath-Ghilbert, IMPS, ... together with some transversal web sites such as MKMnet, HELM, Omega, MathWeb/OMDoc. We will also briefly discuss what this might bring for the future, including web-based approaches to mathematical education.

We restrict our attention mostly to open systems. This is partly because it is much more difficult to obtain information about proprietary ones (in particular you have to pay for them!), and also this information may not be all that relevant other than to users of the particular system. Nonetheless we should mention one major proprietary system *Mathematica* which apparently has latent theorem-verification possibilities.

Most of the information here has been gained by searching the internet with standard tools. The interested reader may readily continue that path and find much fuller and deeper information than we could cover. That having been said, we will not always include the proper references for sources because (1) there would be too many, and (2) it is all available on the web (we try to avoid refering to things you would have to pay for). All told, this represents *another* information-technology phenomenon which has an enormous impact on mathematics, which we could discuss at length but that would depass the scope of a short article. (A *third* phenomenon, the possibility of doing mathematical communication by visioconference, doesn't seem to work well enough yet that we could do more than mention it parenthetically.)

1 "I would like to certify my latest theorem"

We expect that the formalization of a non trivial theorem may require several years. To give the idea, we can cite Thomas Hales who plans to certify his proof of the Kepler Conjecture in twenty years.

Moreover, not all subjects of mathematics have an equal probability of success in formalization with today's technology: we may expect that a geometrical argument of, say, knot theory would be intrinsically more difficult to formalize than a result of algebra or combinatorial logic.

Aside from this, a "classical" mathematician would probably need to enrich his mathematical background with a variety of topics from other domains such as logic and computer-science in order to be effective in the production of a formal document readable by a computer. The training may require a non negligible time (choosing a system, understanding its logic and becoming proficient in its use, learning which results are already made available for that system, etc).

On the other hand, this effort can be repaid in several ways. First of all, to reach a *complete* formalization of a theorem can be a very interesting experience, both for young and for senior researchers, that can be rarely done without the aid of a computer, if we exclude trivial examples in toy theories. The passage from "formalizable" to "formal" mathematics can frequently lead to discovery of a deeper structure in the reasoning and reveal unexpected aspects of the mathematical constructions that can stimulate endless related investigations, subject for other researches and links with other disciplines. Finally, we predict that more and more in the future, there will be disciplines that will be easier to approach with the aid of a computer than without it. This is already the case for several topics of logic and algebra strictly related to the subject: the best way for a beginner to learn type theory is to start to play with a computer assistant with a typeful specification language (more or less as for a young algebraic geometer it can be a good thing to play with a program which computes Groebner bases).

The newcomer trying to find his way in this subject (as we have been) is faced with a number of difficult problems, not the least of which is deciding which system to use and how to use it. We feel it would be harmful to suggest one system or another, since progress can only be made if many people experiment with many different things. However it seems essential to have some sort of guidance.

In trying to understand what is going on with one or many systems, several aspects should be taken into account: the *presentation*, the *underlying logic*, the *architecture*, *tactical language* and the available *decision tactics*, the *proof manager*, the *user interface*, the *documentation* and the *user community*. Let us discuss briefly each of these aspects (see [GH98] in the context of a comparison of PVS and Isabelle/HOL along these lines).

Presentation A proof assistant is typically a program which can be run on an input file (usually text), and which certifies that (1) the file adheres to a specified syntax; (2) according to specified inference rules, the document contains the proofs (and constructions) that it purports to; and (3) any errors are located.

Additional functionalities can include: a top level command loop into which commands can be entered one at a time; a goal window display whereby each command which is entered provokes the display of additional information to show the user what the state of the proof is at that point (generally showing the goal to be proven and the known hypotheses); display of process information saying what the prover tried (Boyer-Moore style); and possibly an integrated user-interface which interpolates between the top level loop and the file-compilation mode. More advanced interfaces can include non-textual commands such as "proof by pointing" [**?**].

Logic Each proof assistant implements a different logic. The vast majority of contemporary mathematics is expressed using the language of set theory (like ZF) and most mathematicians can do their work without too much concern about the very basics of mathematics. Doing mathematics formally implies a change of perspective – at least at the present stage – and the choice of the formal system can be a crucial one. Many theorem provers implement logical systems common in theoretical computer science.

The logics which are used range from first-order or higher-order logic with ZF semantics (ACL2, Mizar, Metamath), to classical higher-order logic in the framework of a simple type theory (HOL/Isabelle), to a full constructive type theory of Martin-Lof including dependent types and universes (Coq, Nuprl).

One basic question about the logic implemented in a system is whether you are allowed to quantify over predicates: this is the distinction between *first order* and *higher order*logic. Actually this distinction can be more nuanced, depending on exactly what kinds of higher-order objects can be manipulated, and how. For example, Coq, Lego and NuPRL are even more higher-order than HOL because they use dependent types [Den00]. This is related to the use of the Curry-Howard principle representing proofs as objects in a propositional type, with the same status as any other mathematical objects. Unfortunately dependent types create other problems which are not currently adequately solved.

Another major question is how the system deals with the problem of Grothendieck universes or other ways of getting around Russell's paradox. Common mathematical practice is at its fuzziest on this point, and maybe for that reason no system has presented a solution which seems adequate. The approach of Mizar might be the best one (at the expense of starting with an axiom which implies the existence of infinitely many inaccessible cardinals). In fact mathematicians would prefer to ignore the problem completely, and system designers try to aim for allowing the user to ignore the problem too. But it doesn't want to go away! Consequently this remains an open research problem, as well as a potential stumbler for neophytes.

In general, a proof assistant is designed for a specific goal in mind. For instance, many systems are directed toward application in hardware and software verification. In some cases, the choice of restricting the support to first order logic can be reasonable. However, we feel that for mathematics in general, it is hard to develop non trivial theories in a specification language that does not allow us to quantify over predicates. (One can even note that HOL, one of the first systems available for higher order logic, was born exactly with the problem of hardware verification in mind).

The logic of the system is strictly related to another important aspect: the *specification language*. Put in simple terms, it is not useful to have a proof of a theorem if it is hard to understand what the theorem really says. When we introduce a new concept (say the definition of group), it is important that everybody agree that the formal definition coincide with the one of diffused practice. An expressive specification language is an important premise for a formalized theory to be useful. In turn, a "rich" logic is one important ingredient to build an expressive specification language.

Higher order logic allows us to use the so called *declarative style*, that is, introduce new concepts by definitions. This has an important practical advantage over the other methodology of using axioms to model new concept, called *axiomatic style*: each time a new axiom is introduced we risk breaking the soundness of the theory (if the axiom it is not well thought or even by a simple misspelling). If you introduce new concepts by definitions, you can be lead to prove unuseful theorems but you do not compromise the soundness of the whole environment.

On the other hand, a restriction to first-order logic (at least in part of a system) can increase the number of decidable algorithmic problems and generally open up more possibilities for automation of proof-search. Otter and ACL2 are examples of this phenomenon.

Since the logic implemented by the system is so important, some proof assistants are designed to be generic with respect to the logic. The implements a metalogic in which several *object logics* can be declared. The Isabelle proof assistant is perhaps the first system designed with this idea in mind (see below). Also MetaPRL developed by the PRL group that also wrote NuRPL is designed as an extension of the OCaml programming language to describe and relate several logics. One different, also interesting, attempt in this direction is the HOL-light proof assistant (see below).

Architecture Not surprisingly, a proof assistant is often quite a complex piece of software. The design decisions made by the developers can influence the life of the user in several ways. One important point at this regard is about the size of the so called *trusted kernel* of the system, that is, the part of the code on which the soundness of the system depends. Some proof assistants are designed to have a reasonably small kernel. This means that developers and users are free to add other functionalities to the system without to worry about the correctness of their code. Since the most popular proof assistants are in constant development, a small trusted kernel can dramatically improve both reliability and usability. However, a small kernel comes with price: all tactics should produce proofs that expand the reasoning to small steps simple enough to be checked by the kernel. This can be a computational cost. It is somehow surprising that the technology permits to write a system with small kernel which also shows good performances.

At this point an historical note about the remarkable work of Milner on the LCF proof assistant at the beginning of seventies is unavoidable. The design of LCF had a direct influence on many successful provers like HOL, Isabelle, Nurpl and Coq which can be all considered direct descendants of LCF. One of the most important ideas of Milner was to use a strictly typed sound programming language to ensure the soundness of the system. To this end Milner designed an *ad hoc* programming language, called ML (Meta Language), to develop one of the versions of LCF. Today development of ML is under active research and it is utilized for a wide range of applications.

Another point that could be considered pertinent is the availability of a good system of "modules" (otherwise known as "books" or "locales") for organizing the argument, partitioning the namespace, and re-using definitions and arguments. Unfortunately this is a newer concept in computer science and it doesn't seem that any one system has completely succeeded on this point yet. Similar considerations are the syntax, together with all sorts of possible syntax facilities to reduce the volume of what is written in the file: implicitization of arguments, overloading of symbols, coercions, notation, etc. One good criterion is to what extent the user has control over his syntactical environment.

Tactical language A library of computer formalized mathematics can hardly be reduced to a collection of definitions and theorems. As in usual mathematics, definitions and theorems are coupled with techniques or strategies about how a certain theory can be used in other proofs. Tactics prescribe a set of recipes that have to be applied when we try to solve a certain goal. The tactical language is a kind of specialized programming language to specify these recipes.

A related aspect is the collection of *decision tactics*, that is, tactics provided by the system that are able to automatically solve a certain class of problems like tautologies, linear arithmetics, first order logic, algebraic identities, etc. A rich set of decision tactics may ease a lot the life of the user.

Decision tactics are the point at which the theorem-verification problem meets up with the research in *finding* proofs by computer. This other field is very rich, with many annual contests to award programs which are the most successful at finding proofs ???refs ???. Whereas the problems that can be solved entirely automatically are pretty limited in a mathematical sense, this technology should be important as it is imported into the theorem-verification problem by way of automation of larger and larger chunks of reasoning.

Meta-proof or reflection One of the most powerful methods for increasing the reach of a theorem prover (in particular by creating decision tactics) is to incorporate the idea of *reflection* [Bou97] [Har95]. This means that a separate logical system is set up and implemented *inside* the ambient logic, and results from the inner logic are exported into the outer one. This requires capabilities such as Coq's `quote` tactic which allow the computer to recognize and transform expressions in the ambient logic which come from expressions written in the inner logic. This might be known as "skolemization" ???. It is the basis of the `ring` tactic in Coq: expressions in a ring are transformed into formal expressions on which computation is then performed: the theorem that the computations are justified yields a proof of the computational result which is then exported back to the original setting.

Many people have mentionned projects ("skeptical systems" or "oracles") [HT98] to contract out this process to other computer programs, envisioning for example a connection between symbolic math programs and theorem-proving programs. It is clear that this kind of symbiosis will be a necessary and important ingredient in important future systems.

David Delahaye and Micaela Mayero have implemented a "Maple mode" for Coq, which is an interface combining Maple computation with Coq [DM]. It uses Maple to simplify a goal and then uses the `field` tactic of Coq to prove the equivalence between the simplified goal and the original one. Another example is "Ivy" [MS00] where (quoting from the resumé) "the study shows how non-ACL2 programs can be combined with ACL2 functions in such a way that useful properties can be proved about the composite programs".

One question which seems legitimate is whether it is better to do this by interfacing with external programs or by reconstituting the external programs in the theorem-proving environment.

User interface A good interface that lets the user prove a theorem with a sequence of clicks on sensitive parts of a formula is more attractive than a command line interface where you have to type extensively each command. However, as a matter of facts, our experience is that we never get accustomed to use a highly evolved interface. This might be due to our laziness. However some comments on this aspect are in order. The main point seems the fact that it is a difficult task to keep in sync a sophisticated user interface with the underlying proof engine. As a consequence, you are often forced to choose between a good interface with an old system and a command line interface with the new version of the software. For us, this dilemma has always been resolved in favor of the first option. Proof assistants evolve rapidly, and at each new release you can find decisive improvements like the "feature X that you badly needed": more expressiveness, improved syntax, fixes in the logic, etc.

Related to the user interface is the *proof manager*, that is, the way in which the system lets you organize the tree of the proof and navigate in it. A very commonly used generic interface is *Proof General* [PG], an emacs package that lets you combine edition of the text file with running the proof engine on it and viewing the output. Various attempts have been made to integrate some type of proof by pointing on top of this type of interface [BKT94]. This is very related to the notion of web-based interfaces that we will consider separately below.

Documentation As for any kind of software, documentation is an important point for the usability of a system. For proof assistants, a reference manual is definitively not enough. In order to learn to use a proof assistant a new user need to learn not only various commands and functionality but also several patterns of utilization and commonly useful tricks. A good tutorial can make a big difference in the first months of utilization.

User community The size of the group of users of a system influences how the system develops: a small group leads to a more homogeneous environment, but leads to an uncertain long-term stability whereas a larger group insures that more long-term effort is made to insure continuity, but leads to a widening of the range of style and logic which are used which makes it more difficult to enter the subject. An important aspect for increasing homogeneity as well as long-term stability is to have a good standard for diffusion of user contributions.

A lively user's group with discussion on the mailing lists is generally very helpful for new users.

2 Specific systems

We would like to consider in more detail a small number of systems. The comparison between various theorem-proving systems has become a veritable literary *genre* (comparison of PVS and HOL [GH98], Mizar and Isar [MW02], Coq and HOL [Zam97], . . .). There is a very thorough and well done "road test" of 15 theorem provers [Wie03], which among other things discusses in detail the proof of $\sqrt{2} \notin \mathbf{Q}$ in each of these systems. We will try to synthesize and complete this information with respect to a small number.

We start with ACL2, exemplary of the Boyer-Moore family of provers that have been under developpement for decades. The logic is resolutely first-order [KM98]. The user types in the definitions and statements of theorems, and the computer takes care of the rest: adding in appropriate statements as complements to definitions, searching for the proof of each new theorem, and telling the user what it is doing along the way. The user just has to set up the right sequence of lemmas (which is still a nontrivial task). This is very cool, and it would be great if the algorithms for doing that could be imported into other systems.

It is difficult to invent how to apply ACL2 to higher-order reasoning, for example the library contains books about finite sets but it is harder to find how to treat arbitrary sets. Even quantifiers are handled in a somewhat external fashion (at one point in the documentation, with probably exaggerated pessimism, they say: "Don't expect too much automatic support from ACL2 for reasoning about quantified notions"). This makes it difficult to imagine directly applying ACL2 to highly abstract mathematical reasoning. Of course it is possible, by altering the approach to the *specification language*, to reduce higher-order systems to first-order systems by moving up a level: any system is in the end implemented as a deterministic program, so it is possible, in principle, to give a first-order axiomatization for what the system does. Thus instead of proving a higher-order statement P, you would prove the first-order statement "my higher-order system allows me to prove the statement written as P". This isn't very convenient at least to start with, and it seems to nullify the advantage of having somebody else write the computer program since you end up having to codify some sort of program again. Beyond these remarks, we are not sure whether this possibility has been systematically explored, so in spite of what look like obvious disadvantages, there might be some hidden advantages to this approach that we don't yet know about.

We next consider MIZAR. This implements a classical mathematical set theory with infinitely many Tarski-Grothendieck universes. Proof documents are written in something approaching natural language; and there is a huge library of results which is very well integrated into the system. The library results are included in the *Journal of Formalized Mathematics* which even prints a paper version. The library includes very advanced results on combinatorial mathematics, and also results which are actually quite complete in a wide array of mathematical domains.

Two things that make it difficult to get started are that Mizar doesn't provide the user with a "goal window" showing what is to be proven at any step in the proof; and there is no documentation manual. Indeed, in ???Freek???, Wiedjik says that the best way to learn the system is to spend three weeks with the Mizar group in Poland.

One might say that using interactive theorem provers dulls the mind, in that we get into a habit of floating around in the middle of a proof having forgotten entirely what it is we are trying to prove, but still manage to get out anyway. An habitual user of interactive provers may find it difficult to go back to the mental rigor required of writing a proof in Mizar where you have to keep in mind at each stage what the current goal is.

This may be intentional on the part of the system designers: there is a case to be made for the idea that lists of proof tactics don't give a very good idea of how a proof works, and that if the author is forced to follow his own proof as he is writing it, then the writing will be understandable to other readers. We don't actually think this is true. For example, the Mizar library contains a rather interesting theorem in its foundational material: starting from an axiom of Tarski which more or less says that every set is contained in a Grothendieck-type universe, they *deduce* the axiom of choice. It turns out that this depends a lot on the precise form of the Tarski axiom. We were not able, by looking at the Mizar articles where this happens, to understand how it worked. Basically what was missing was any distinction between important and unimportant steps in the proofs. It is certainly also possible that important steps get hidden inside tactic scripts. However, it seems that the most common situation is that important steps correspond in some way to tactics which look a bit out of the ordinary, and which would stand out under a rapid examination of the tactic script. Or alternatively, much as would be the case for an ACL2 proof, the basic outlines of the proof would come out of the structure of the set of prior lemmas which had to be proven.

Irrespective of the functional question of which form best enables readers to read a proof, there is also a basic question of emphasis: are we most interested in creating proofs which are readable by the human reader? or are we most interested in creating, as quickly and easily as possible, true proofs that are verified by the computer and which we don't subsequently care about? The first approach has the long-term advantage that the existence of the document doesn't rely on the existence of a computer available to read it. Nonetheless, we feel that the greatest benefits will come from the second approach.

Next we come to "HOL". On one side, HOL is a proof assistant, direct successor of LCF. Its name comes from the logic that it implements: a Higher Order Logic *à la Church*. The last version is HOL4 [hol]. However, HOL is also a family of proof assistants that share to some extent the same logic and architecture. Members include also Isabelle/HOL and HOL-light (see below). For an historical account of the HOL system and its relatives we suggest the paper of Gordon *et al* [Gor00]. The kernel of HOL guarantees the soundness of the system by accepting new theorems only if they are introduced with a set of fixed inference rules. Tactics are programs that find the good sequence of inference rules accepted by the kernel to prove a goal. This is different from the design of other proof assistants like Coq or NuPRL that implement logic which is a variant of the Martin-Löf type theory and have a kernel that is basically a typechecker (thanks to the Curry-Howard isomorphism). A form of the axiom of choice is built-in as inference rule in the kernel of HOL.

It is perhaps remarkable that, according to Gordon, one of the main applications in mind, during the design of HOL, was hardware verification. The first order logic of LCF was extended to higher order logic in HOL to this practical end, more than for other general/mathematical/foundational reasons.

Isabelle is one of the main implementations of HOL, although a distinguishing characteristic of Isabelle is its genericity with respect to the logic in the sense that its architecture is based on a kernel for a *meta-logic* in which several *object-logics* can be implemented by declaring their inference rules. Isabelle is distributed with a library of several logics including HOL but also ZF (Set theory), FOL (first order logic), and Cube (Barendregt's *Lambda Cube* [Bar91]). The proof manager supports both goal-directed and, recently, a Mizar-like style.

The HOL logic seems to be the most utilized and developed and the one which is most tightly integrated with the kernel. Isabelle/HOL has a rich specification language that features polymorphism, overloading, an advanced system for syntax and useful libraries for a (typed) set theory. A relatively large and complete library of results for Isabelle/HOL is available from the web site.

Isabelle has an advanced mechanism for notation. The user has the ability to fine tune the syntax used in definitions and proofs. TEX symbols can be used to have a good looking typesetted documents.

Proof General is the preferred interface for Isabelle. The system provide some powerful general tactics (e.g., `simp` for rewriting, `blast` for finishing the proof automatically). The user needs to understand quite in depth how these tactics work internally to be able to give appropriate hints about the intended usage of newly proved results in subsequent proofs.

An ample tutorial [NPW02] is available which not limited to showing the basic commands and examples but also tries to give some general hints on how to prove a theorem by showing several examples of successful and failing strategies in different contexts.

It is unfortunate that the different object logics do not interface each other. This creates a separation between results proved in different logics

(HOL versus ZF). Also, there are no dependent types or other manipulation of the sort "Type" such as in Coq or Nuprl, which in the long run could prove to be a handicap.

The tactical language is SML itself, that is, a powerful typed higher order programming language. However, to write a tactic for Isabelle in plain ML seems to be a quite technical task. Fortunately, this seems rarely needed (unless one wants to program new object logics) thanks to the availability of powerful general tactics.

The only format of all Isabelle documentation is ps/pdf which is not the best for reading on a terminal. An hyper-textual/html version of the same documentation would certainly be more practical for searching and on-line reading.

HOL-LIGHT is a reengineering of HOL written by John Harrison [Har00]. It is not comparable with other HOL-like systems due to its much smaller user community (note however that this is the system of choice for the long term project of Thomas Hales cite [Hal]). Its very clean and concise source code (less than 800 kbytes) gives you the pleasant impression that the system is in your hands. There is no real proof manager other than the read-eval-loop of Caml-Light (the dialect of ML in which the system is written) and the tactical language is nothing more that Caml-Light as well. Its simplicity makes the system a possible candidate for experiments and informal (and, perhaps, formal) considerations about variants of the logic and the architecture of HOL-like systems. Denney has a project of importing in Coq the proof of the fundamental theorem of calculus written by Harrison in HOL-Light [Den00]. There is also a "Mizar Light" mode for HOL-Light [Wie01].

We now come to the assistants which implement a logic with dependent sorts. NUPRL and COQ use a heirarchy of universes $Type_i$ which appear as ordinary objects, in that we can do manipulations such as considering arrows $F : B \to Type_i$ and taking the product of the (Fb) over $b \in B$. The NuPRL manual has a good discussion about the logic they are using, the relationship with set theory and so forth. In NuPRL the tactic language is to write tactics directly in ML. This is an attractive idea since tactics don't need to be rigorously verified (they have for goal just to provide a proof of the theorem in question; the fact that the provided term is a proof is of course to be rigorously checked by the theorem-prover, but you are free to write tactic programs which have bugs in them). Unfortunately the manual dates from 1995 so it isn't clear what type of current support is maintained for NuPRL. There is a more modern variant NuPRL4, and a global project known as MetaPRL, so development of the system might be picking up after a hiatus. The lack of continuity in the development of the system shows up as sparsity of the mathematical subjects in the library [NuPb] [Paua].

The Coq system is the primary one used by the authors. It lies in the class of very-high-order logics, implementing a typed lambda calculus with sorts and dependent types, as well as inductive objects. But it is much more heavily maintained and has a very wide group of users.

The reference manual is complete and it is available both in ps and html form. We find that there is room for improving the tutorial adding more examples and hints about common techniques and coding style. A good, more advanced, tutorial "on recursive types" written by Gimenez is available [Gim98]. A book of Bertot and Castéran, that could be a good introduction and a guide to several advanced topics about Coq, has been recently published [BC04]. Recently with the release of version 8.0, a revised, much more human-readable syntax, together with a powerful system for notation, has been introduced.

Coq has a module system modeled on the module system of OCaml. Modules can be parametrized on other modules (so called *functors*). This still has some obscure points, like the management of metalogical data (hints, coercions) within modules. Also it is not completely clear that the precise choices in syntax and implementation of modules were happy ones for the mathematical user; from afar at least, the modules of Phox and the locales of Isabelle look like they might be more useful.

Proofs are done by tactics which modify the "goal window" telling us what we want to prove and with which hypotheses. This leads to a fast interaction between the user and the computer. One must be aware that proofs by tactics end up yielding proof scripts which may well be incomprehensible at least without following step-by-step the goal window output (although as mentionned regarding Mizar, this might in the end be preferable for understanding global proof strategy). Emphasis on backward logic, and also on decidability of proof algorithms, means that a lot of obvious chances for automation are missed. For example the "auto" tactic isn't able to prove "forall a b c, a=b -¿ b=c -¿ a = c".

Perhaps most problematic is the lack of any good consensus on logical foundations and implementation of set-like objects. There are many different ways to look at a set: it can be a type; or a type with an equivalence relation (or even a partial equivalence relation defined only on the "valid" elements); or an element of a given type said to represent the class of all sets (and this type could either be axiomatized or else constructed as in Aczel and Werner). Different user's contributions use different versions of the notion of a set, as well as inhabiting different logical worlds with differing levels of intuitionism or contrarily classical logic being supposed. This means that the different contributions cannot *a priori* be made to work together, so in principle new contributions tend to start from scratch. It is often difficult even just to download, install and use other contributions. It is possible that this difficulty could be overcome with further work, both on a foundational and a practical level. But it makes for a situation which, to a newcomer, is bound to be bewildering at best.

In the above discussion we have restricted to what could be considered as the major systems at the current time. These will probably evolve or be supplanted in the future, and there are any number of young systems proposing differing points of view: the reader should look up ALF, Phox, Plastic, Metamath, Epigram, ...

3 Available results

The mathematics which has already been formalized in various systems goes way past the standard undergraduate curriculum (although not necessarily subsuming all of it), and makes serious inroads into the graduate curriculum. There is not yet quite enough material available to be able to start research work in a large number of fields, but this is not far away; and in a small number of fields specially those more closely related to computer-science and logic themselves, it is currently possible to do new research mathematics using theorem provers. In a few years this will extend much further, and in some fields it may well be *impossible* to do new research *without* making use of theorem provers.

We now try to give a more detailed list of mathematical topics that have been mechanized, being well conscious that we cannot aim for exhaustiveness. Note that some contributions have a particular emphasis torward intuitionism and constructive mathematics, so they can be used in constructive and non constructive mathematics. We avoid giving an endless list of citations by omitting all references to formalized documents that are accessible by the standard means (i.e., by a simple search in the web site of one of the cited provers ACl2 [ACL], Alpha [Alp], Coq [Coq], ELF [ELF], EQP [EQP]. Ghilbert [Ghi], HOL4 [hol], Lego [LEG], Isabelle [AFP], IMPS [IMP], MetaPRL [Metb], Metamat [Meta], Mizar [Miz] [JFM], NuPRL [NuPb], Otter [Ott], Phox [Raf], PVS [PVS], Theorema [The], TPS [TPS] Yarrow [Yar] Z/Eves [Z/E]).

• **Axiomatic Set Theory** [Far01] [PG96] [Gor] [AG95] [Ale]. Examples: cardinals and ordinals, the Cantor-Schroeder-Bernstein theorem [Hug], relative consistence of the axiom of choice [Pau03], formalization of paradoxes [BCW];

- **Groups:** Zassenhaus theorem, the Jordan-Hölder theorem [KP99];
- **Rings, ideals, fields** [Sai98] (often including specialized decision tactics for these structures [Bou97] [GWZ00a] [Har01]). Sometimes using the approch of universal algebra [Cap];
- **Linear algebra:** vector spaces and modules over rings, the Nakayama lemma, tensor products;
- The **fundamental theorem of algebra**: in particular, see [GHW$^+$] [GWZ00b] for a constructive proof;
- **Buchberger's algorithm** for computing Gröbner bases in polynomial ideals [Thé01], and more generally;
- The **Knuth-Bendix** critical pair theorem [RAHM00] [RAHM00];
- **Homological algebra** [ABR03], [HR03];
- **number theory**: chinese remainder theorem, quadratic reciprocity; [Jer];
- Basic **classical real and complex analysis** with inequalities, sums and limits, continuity and derivatives, integrals [GN02] [Har01] [Jut77] and
- The **fundamental theorem of integral calculus**;
- in fact all of Landau's analysis book was formalized long ago [**?**];
- **Nonstandard real analysis** including the ultrapower construction of the hyperreals [FP00];
- The **Bolzano-Weierstrass** Theorem;
- **Metric** and **normed spaces** [FT91];
- **Newton's *Principia Mathematica*** [Fle01] [FP99];
- Basic **general topology** [Fri04] [FT91];
- basic **euclidian geometry** and non-euclidian geometry [CK86];
- **algebraic topology** [Ken];
- The **Church-Rosser** theorem;
- Unsolvability of **Turing's halting problem**;
- **Gödel's first incompleteness theorem** (ex:[O'C] ???other);
- Basic definitions in **algebraic geometry**: the notions of locally ringed space, the Zariski topology on $Spec(A)$ and the notion of affine scheme [Chi03];
- Basic **category theory** [HS00] [O'K04];
- ...

We don't include any discussion of applications to formal reasoning about computation, which (unfortunately, from our point of view) are far more widespread than the above types of applications to mathematics.

On the other hand there is a fascinating class of applications which are between the two fields, where computer proof is used to make progress on mathematical problems which we might otherwise have remained untouched. These include:

- the proof that **XCB** is a single axiom for equivalential logic [WUF02];
- **uniqueness of the 5-ary Steiner Law** on a cubic curve [PM96]
- **"a cancellative semigroup operation on a cubic curve must be commutative"**;
- **Robinson algebras are boolean** [McC97] [Kol96].
- ...

4 Mathematical theory-sharing projects

One of the basic facts of life in this domain is that the quantity of mathematics which is currently known and which it would be good to formalize in some system, is overwhelming. As a result, it is quite impractical for a single person to do it all—some mode of human interaction is required. The organisation of this interaction, which is first and foremost a problem of *sharing theories* among many users, is almost as difficult a problem as the organisation of the human-computer interaction at the heart of the theorem-proving process. The high sociological component means that choices of mode of interaction have a significant effect on the resulting output.

Perhaps the oldest and most impressive theory-sharing project is the *Mizar mathematical library* recorded in the *JFM*. We suggest that readers scan those web pages (for example the MML identifier index page) to get an idea of the extent of mathematics which has been formalized within this project. The interaction between different users passes through a centralized "library committee" which validates new contributions to the library; and all entries in the MML have equal standing as far as being referenced in future contributions. The uniformity of logic and notation guaranteed by the library committee permits the uniformity of conditions of utilisation.

This system is not very agile on the other hand. Among other things it has prevented significant updating of the specification language to incorporate more modern type-theoretic notions as well as goal-oriented and tactical proof presentation such as appear in other systems we have discussed above.

Isabelle also has good criteria for leading to contributions which can be reutilised, and is just starting a "journal". The Coq user-contributions are a much more freewheeling affair, which is in a certain sense less stiffling but also could become problematic for the notion of building new contributions on top of prior ones from other groups.

One can imagine a whole spectrum of possible modes of mathematical interaction. A lot of interest is currently given to the notion of *web-based* mathematical knowledge databases, with the HELM, MKMnet, MetaPRL and Omega projects. It should be possible in the not-to-distant future to set up a system whereby mathematicians work directly in an interface to a common database, with each lemma going into the database when it is proven, where it could then be used by others. In this way it might be possible to maintain projects with significant numbers of people working on the same thing at the same time. This would represent a big velocity gain with respect to the present situation where people must essentially work independently. Even in projects where different people contribute, it is still necessary for each one to pretty well understand the whole project.

Some readers will have said, "wait a minute, are you saying that it would be better if lots of people worked on the same project but nobody had a global understanding of it?" It is true that this seems a bit shocking. Actually when you think about it, the modern world is organized to a great extent in exactly this way. We are becoming less like a hologram and more like an ant colony. We are not claiming that the world is necessarily going to be any *better* (indeed that would depend on your definition of better), just that it is going to be *different*. Just as often the most charming architectural device to this day remains the dry-laid stone wall, the rugged die-hard individualist of a mathematician who still sticks to pencil and paper in spite of the ambient electronic or even quantum revolution, undoubtably will occupy a valued niche. Nonetheless, other great things will be accomplished by those who push the available tools to their utmost limits.

So if we assume that it is desireable to organize a distributed approach to theory development, there still remains the question of how to do it. This is currently under experimentation, notably in the HELM (Hypertextual Electronic Library of Mathematics) project, but also in somewhat similar projects MathWeb and Open Math Doc, MetaPRL and Omega (most of which can be found within the umbrella Mathematical Knowledge Management Consortium web pages—but also individual systems such as Mizar but also Isabelle and Coq have to deal with this problem just within their own libraries). In all cases the idea is to organize a database with essentially automatic tools (as distinguished from the Mizar library committee which is human-based), such that the fundamental operations of searching for a given result or result relevant to a given topic, importing that result into the user's local theorem-proving environment, and adding a new result to the database, are as easy and natural as possible. Given the enormous diversity of notations, this is a nontrivial task.

The HELM project is a good example. They decompose the proof documents into a large number of small bits that are managed with the aid of XML technology and make the theories available for searching and browsing through an html browser (with a fast Internet connection!).

The search strategy of the Helm server makes several assumptions on the coding style. This makes the system effective for finding a theorem in the standard library but not useful for all users. It should be stressed that one of the main points is to have good search tools: a formalized theory is often composed of an impressive number of small lemmas. Each lemma is stored with a corresponding name that will be used to recall the result in the subsequent proofs. Sometimes is easy to find a good name, e.g., `plus_assoc` for the associativity of the addition, but this is the minority of the cases. More often you have to give name to a statement like $\forall n\ m, n < S\ m \to n \leq m$ (a random example from the Coq library, S is the successor of a natural number) and you end by using names like `lt_n_Sm_le`. An efficient search tool would solve the problem by giving, at each step of a proof a collection of pertinent results already available in the library. (Actually, high-level Coq tactics like `auto`, `omega`, `ring`, may solve certain goals without the need of explicitly specify which lemma are needed in order to build the proof).

Apart from the problem of retrieving results from the library, there is a more difficult problem of knowing how to let users post results into the library. For the moment the HELM library has automatically integrated user contributions from Coq and NuPRL (so users don't have to do anything particular). In the future it will probably be desireable to have a specific interface for placing results in the library. The Omega project seems to be working on this, and originating in a more educational venue the COQWEB project within WIMS provides an example of the type of web interface which might be good.

5 The future

As mathematics expands and becomes more diverse, we are increasingly reliant on the refereeing system to validate techniques and results; but at the same time with the quantity of mathematics now being produced, the refereeing system is becoming incapable of assuming this responsability. The notion of computer-verified proofs and other mathematical documents will play a role in rapid and accurate validation of our work.

It is reasonable to expect that more and more journals for certified mathematics will come into being. The oldest is Mizar's *JFM*, but recently Isabelle system has also started organizing its user-contributions with a journal.

It is interesting to note that the Cornell math preprint "ArXiV" now accepts *Mathematica* notebook files as a contribution format. Regretting that this concerns a system you have to pay for, we can hope that this program could be extended to other file formats—but one can note that the systems we have discussed above don't agree on any type of common format.

This is not to say that the role of refereeing will disappear. It would be good not to pretend that certified proofs say all that much about the actual content of the theorems which are being certified; the refereeing or other peer-review processes will be helped by verification of the basic mathematical details of arguments, and can more serenely consider the fundamental questions of content.

The field of computer proof is under continual development. Reflection, oracles/skeptical systems (i.e. the interface with other types of computer calculation) will play a big role. Each area of mathematics presents special difficulties for construction of the tools which are adapted to it. Once this will be done, mathematicians will have at their disposal a wide array of powerful possibilities for doing math interactively with the computer. We noticed on the Isabelle projects page a particularly interesting project by J. Aransay, C. Ballarin and J. Rubio on formalized algebraic topology. They propose to verify proofs of the computer algebra systems EAT and Kenzo for doing calculations in algebraic topology [HR03].

This type of environment will lead to a big increase in the speed and efficiency of development of large and complicated theories, but also an increase in the ease of access to these theories. It will probably also lead to fascinating cases in which proofs found interactively with a lot of help from the computer, will be so new and strange that we will find it very hard to fully grasp them [**?**] [**?**].

There will probably also be lots more mathematical theories available for people to play around with. It becomes practical to try out a set of axioms for a theory to see what it produces, without needing a large group of people who have to understand the motivation for the axioms (there might not be any), just in order to verify the results. Thus, at the same time as group-projects are enabled by theorem-proving systems, also isolated individual projects are enabled too.

The notion of computer-verified proof also represents a profound change for mathematical education. Mathematics has traditionally been the subject where students come into the closest contact with the notion of *rigour*.

Even if the subject matter has evolved, the basic techniques of definitions, statements and proofs, haven't changed much for several thousand years. It can perhaps be summed up by saying that the educational process was a way of transforming what started out as dialogue between the student and the teacher, into a dialogue between the student and him- or herself. Now a third entity comes into the game: dialogue between the student and the computer.

C. Raffalli reports on a project to use his Phox theorem prover as the focus of study groups in upper undergraduate mathematics [RD02]. In his project the students learn to develop a proof of a relatively advanced theorem within the context of the system. The tactics are reparametrizable along the way (this would be a good thing for other systems to implement) which makes it so that the teacher can prepare a setting whereby the students can concentrate on the problem at hand without having to learn all of the intricacies of the full theorem-verification problem.

G. Xiao is developping a global mathematics educational system over the web known as WIMS. It is already being used in coursework here at the University of Nice, and elsewhere (Orsay ...). Computer verification of the answers plays an important role in having students meet face to face the problem of mathematical rigor, and allows for the programming of intricate and varied exercises. A nascent experimental project by L. Pottier is to integrate verified theorem proving into the WIMS environment by an interface COQWEB which allows students to perform interactive proofs in the Coq system, but guided by a web page where the proof options are chosen by clicks. This might provide a good style of interface to access databases such as in the Helm project.

We close by mentionning—by way of example—a diverse collection of topics which we think it might be interesting and practical to try for.

—look for algebraic cycles which represent some of the known Hodge classes such as Kunneth projectors, the Lefschetz operator, etc.

—formalize the inductive procedures in the "méthode d'Horace" and other enumerative techniques (A. Hirschowitz, L. Chicli), or look for alternative computational methods (F. Schreyer)

—look for non-rigid representations of fundamental groups, or actually any finitely presented groups (A. Magid).

—formalize Quillen's closed model categories and other aspects of higher category theory

—automate the process of doing "diagram-chase" type arguments in category theory (J. Lipman); also try doing this for higher category theory.

—verify semantic interpretations for things like lambda calculus, typed

lambda calculus or other similar systems, in things like the internal logic of a topos.
—provide seamless computer proofs in places where we now have computer-assisted proofs (this is an idea of a number of people; we first heard about it from Benjamin Werner... also McKay suggests that a first problem to look at would be the nonexistence of a projective plane of order 10 ...)
—For the far future: imagine scanning EGA and running a program on the file, which 2 seconds later says "ok: mostly fixable; ... found error lemma ?.?.? page ??, counterexample as follows ... but it was an inessential error: that lemma was used in ? and ? but this can be fixed as follows ...". A shorter term goal would be to try to see how easy it is to formalize something like EGA directly from text.
—We can finish a pretty obvious project that must have crossed many peoples minds: to verify Taylor-Wiles' proof of Fermat's last theorem. If this could be done it would really represent a new state of affairs in mathematics.

Conclusion: For the future we think it is best to view this subject as a cooperative venture. If you had a system which combined ACL2, Otter, Maple or other symbolic computation programs, Isabelle/HOL, Coq, Phox, and Mizar—and which started from a library containing all of the results which have been formalized in any one of these systems—it would be really great! This would cause us to run up against one of the real fundamental problems: how an average user can get the thing installed and running correctly (and not crashing or overheating) on his own machine...

References

[ABR03] Jesús Aransay, Clemens Ballarin, and Julio Rubio, *Towards a higher reasoning level in formalized homological algebra*, 11th Symposium on the Integration of Symbolic Computation and Mechanised Reasoning (Calculemus) (Rome, Italy) (Thérèse Hardin and Renaud Rioboo, eds.), Aracne Editrice, 2003, http://www4.in.tum.de/~ballarin/publications/calculemus2003.pdf, pp. 84–88.

[ACL] *ACL2 system*, http://www.cs.utexas.edu/users/moore/acl2/acl2-doc.html.

[Acz78] Peter Aczel, *The type theoretic interpretation of constructive set theory*, Logic Colloquium '77 (Proc. Conf., Wrocław, 1977), Stud. Logic Foundations Math., vol. 96, North-Holland, Amsterdam, 1978, pp. 55–66. MR 80f:03068

[Acz82] ———, *The type theoretic interpretation of constructive set theory: choice principles*, The L. E. J. Brouwer Centenary Symposium (Noordwijkerhout, 1981), Stud. Logic Found. Math., vol. 110, North-Holland, Amsterdam, 1982, pp. 1–40. MR 85g:03085

[Acz86] ———, *The type theoretic interpretation of constructive set theory: inductive definitions*, Logic, methodology and philosophy of science, VII (Salzburg, 1983), Stud. Logic Found. Math., vol. 114, North-Holland, Amsterdam, 1986, pp. 17–49. MR 88a:03149

[Acz99] ———, *On relating type theories and set theories*, Types for proofs and programs (Irsee, 1998), Lecture Notes in Comput. Sci., vol. 1657, Springer, Berlin, 1999, pp. 1–18. MR 1 853 592

[AFP] *The Archive of Formal Proofs*, http://afp.sourceforge.net/.

[AG95] Sten Agerholm and Mike Gordon, *Experiments with ZF set theory in HOL and Isabelle*, Higher order logic theorem proving and its applications (Aspen Grove, UT, 1995), Lecture Notes in Comput. Sci., vol. 971, Springer, Berlin, 1995, pp. 32–45. MR 97k:03008

[Ale] G. Alexandre, *An axiomatisation of intuitionistic zermelo-fraenkel set theory*, http://coq.inria.fr/contribs-eng.html.

[All87] Stuart Allen, *A non-type-theoretic definition of Martin-Löf's types*, in *Proceedings, Symposium on Logic in Computer Science* [Com87], pp. 215–221.

[Alp] *Alfa system (formerly ALF)*, http://www.math.chalmers.se/~hallgren/Alfa/.

[AP90] M. Abadi and G. D. Plotkin, *A per model of polymorphism and recursive types*, in *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science* [IEE90], pp. 355–365.

[Avi] Jeremy Avigad, *Jeremy Avigad's Home Page*, http://www.andrew.cmu.edu/user/avigad/.

[Bar] Henk Barendregt, *Foundational papers by henk barendregt (and co-authors)*, Home page of the author: http://www-school.cs.kun.nl/~henk/papers.html.

[Bar91] Henk Barendregt, *Introduction to generalized type systems*, Journal of Functional Programming **1** (1991), no. 2, 125–154.

[Bar03a] Henk Barendregt, *Foundations of mathematics from the perspective of computer mathematics*, 2003.

[Bar03b] ———, *Towards an interactive mathematical proof*, Thirty Five Years of Automath (F. Kamareddine, ed.), Kluwer, 2003, pp. 25–36.

[BC04] Yves Bertot and Pierre Castéran, *Coq'Art: The Calculus of Inductive Constructions*, Text in theoretical computer science: an EATCS series, vol. XXV, Springer-Verlag, 2004.

[BCD90] Alexandre Boudet, Evelyne Contejean, and Hervé Devie, *A new AC unification algorithm with an algorithm for solving systems of diophantine equations*, in *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science* [IEE90], pp. 289–299.

[BCW] B. Barras, T. Coquand, and B. Wernerm, *Paradoxes in set theory and type theory*, User contribution for Coq.

[BG01] Henk Barendregt and Herman Geuvers, *Proof-checking using dependent type systems*, Handbook of Automated Reasoning, 2001, pp. 1149–1238.

[BKT94] Yves Bertot, Gilles Kahn, and Laurent Théry, *Proof by pointing*, Theoretical aspects of computer software (Sendai, 1994), Lecture Notes in Comput. Sci., vol. 789, Springer, Berlin, 1994, pp. 141–160. MR 95f:68210

[Bou66] Nicolas Bourbaki, *Éléments de mathématique. Fasc. XVII. Livre I: Théorie des ensembles. Chapitre I: Description de la mathématique formelle. Chapitre II: Théorie des ensembles*, Actualités Scientifiques et Industrielles, No. 1212. Troisième édition revue et corrigée, Hermann, Paris, 1966. MR 34 #7356

[Bou97] S. Boutin, *Using reflection to build efficient and certified decision procedures*, TACS'97 (Martin Abadi and Takahashi Ito, eds.), vol. 1281, LNCS, Springer-Verlag, 1997.

[Bun99] Alan Bundy, *A survey of automated deduction*, Lecture Notes in Computer Science **1600** (1999), 153–??

[BW97] B. Barras and B. Werner, *Coq in Coq*, http://pauillac.inria.fr/~barras/coqincoq.ps.gz, 1997.

[Cap] Venanzio Capretta, *Universal algebra in Coq*, http://www-sop.inria.fr/Venanzio.Capretta/universal_algebra.html.

[CH85a] Thierry Coquand and Gérard Huet, *Constructions: a higher order proof system for mechanizing mathematics*, EUROCAL '85, Vol. 1 (Linz, 1985), Lecture Notes in Comput. Sci., vol. 203, Springer, Berlin, 1985, pp. 151–184. MR 826 550

[CH85b] ______, *A selected bibliography on constructive mathematics, intuitionistic type theory and higher order deduction*, J. Symbolic Comput. **1** (1985), no. 3, 323–328. MR 87j:68001

[CH85c] ______, *A selected bibliography on constructive mathematics, intuitionistic type theory and higher order deduction*, J. Symbolic Comput. **1** (1985), no. 3, 323–328. MR 87j:68001

[CH88] ______, *The calculus of constructions*, Inform. and Comput. **76** (1988), no. 2-3, 95–120. MR 89j:68096

[Cha04] Kenneth Chang, *In math, computers don't lie. or do they?*, New York Times (2004), April 6.

[Chi03] Laurent Chicli, *Sur la formalisation des mathématiques dans le calcul des constructions inductives*, Ph.D. thesis, Université de Nice-Sophia Antipolis, Nice, Nov 2003, http://www-sop.inria.fr/lemme/Laurent.Chicli/these_chicli.ps.

[CK86] Shang-ching Chou and Hai-Ping Ko, *On mechanical theorem proving in Minkowskian plane geometry*, in *Proceedings, Symposium on Logic in Computer Science* [IEE86], pp. 187–192.

[CMT02] Victor A. Carreño, César A. Muñoz, and Sofiène Tahar (eds.), *Theorem proving in higher order logics: 15th international conference, tphols 2002*, Lecture Notes in Computer Science, vol. 2410, Hampton, VA, USA, Springer-Verlag, August 2002.

[Com87] The Computer Society of the IEEE, *Proceedings, symposium on logic in computer science*, Ithaca, New York, 22–25 June 1987.

[Con] Daniel W. Connolly, *A survey of QED and related topics*, http://www.w3.org/Math/QED.html.

[Coq] *The Coq Proof Assistant*, http://coq.inria.fr.

[Coq86] Thierry Coquand, *An analysis of Girard's paradox*, in *Proceedings, Symposium on Logic in Computer Science* [IEE86], pp. 227–236.

[Cou96] Judicaël Courant, *Un calcul de modules pour Coq*, Actes de la journée du Pôle Preuves et Spécifications Algébriques du GDR de programmation (Orléans, France), 1996.

[Cou97a] ———, *An applicative module calculus*, Theory and Practice of Software Development 97 (Lille, France), Lecture Notes in Computer Science, Springer-Verlag, April 1997, pp. 622–636.

[Cou97b] ———, *A Module Calculus for Pure Type Systems*, Typed Lambda Calculi and Applications 97, Lecture Notes in Computer Science, Springer-Verlag, 1997, pp. 112 – 128.

[Cou98] ———, *Un calcul de modules pour les systèmes de types purs*, Thèse de doctorat, Ecole Normale Supérieure de Lyon, 1998.

[Cou99] ———, *$\mathcal{MC}$: A module calculus for Pure Type Systems*, Research Report 1217, LRI, June 1999.

[Cou01] ———, *$\mathcal{MC}_2$: A Module Calculus for Pure Type Systems*, Research Report 1292, LRI, September 2001.

[Cou02] ———, *Explicit universes for the calculus of constructions*, in Carreño et al. [CMT02], pp. 115–130.

[CPS03] Laurent Chicli, Loïc Pottier, and Carlos Simpson, *Mathematical quotients and quotient types in Coq*, Types for Proofs and Programs (H. Geuvers and F. Wiedijk, eds.), LNCS, no. 2646, Springer, 2003, pp. 95–107.

[dB80] N.J. de Bruijn, *A survey of the project Automath*, To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism (J. P. Seldin and J. R. Hindley, eds.), Academic Press, 1980.

[Den00] Ewen Denney, *A prototype proof translator from HOL to Coq*, Theorem Proving in Higher Order Logics, 2000, pp. 108–125.

[DM] David Delahaye and Micaela Mayero, *A Maple mode for Coq*, Coq contribution http://pauillac.inria.fr/coq/contribs/MapleMode.html.

[Dow97] G. Dowek, *Démonstration automatique*, 1997, Notes de cours : Ecole Nationale Supérieure des Techniques Avancées.

[ELF] *The ELF meta-language*, http://www-2.cs.cmu.edu/˜fp/elf.html.

[EQP] *EQP equational prover*, http://www-unix.mcs.anl.gov/AR/eqp/.

[Far01] W. M. Farmer, *STMM: a Set Theory for Mechanized Mathematics*, Journal of Automated Reasoning **26** (2001), 269–289, http://imps.mcmaster.ca/doc/stmm.ps.

[Fel93] Amy Felty, *Encoding the calculus of constructions in a higher-order logic*, in *Proceedings, Eighth Annual IEEE Symposium on Logic in Computer Science* [IEE93], pp. 233–244.

[Fer01] S. Fereman, *Typical ambiguity: trying to have your cake and eat it too*, Proceedingsd of the conference Russell 2001 (Munich), June 2001, http://math.stanford.edu/ feferman/papers/Ambiguity.pdf.

[Fle] Jacques Fleuriot, http://www.dai.ed.ac.uk/homes/jdf/pub.html.

[Fle01] J. D. Fleuriot, *A combination of geometry theorem proving and nonstandard analysis, with application to newton's principia*, Springer-Verlag, 2001.

[FMRS90] P. Freyd, P. Mulry, G. Rosolini, and D. Scott, *Extensional PERs*, in *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science* [IEE90], pp. 346–354.

[FOR] *Formal methods web page at Oxford*, http://www.afm.sbu.ac.uk/.

[FP99] Jacques D. Fleuriot and Lawrence C. Paulson, *Proving Newton's Propositio Kepleriana using geometry and nonstandard analysis in Isabelle*, Automated deduction in geometry (Beijing, 1998), Lecture Notes in Comput. Sci., vol. 1669, Springer, Berlin, 1999, pp. 47–66. MR 1 775 946

[FP00] ———, *Mechanizing nonstandard real analysis*, LMS J. Comput. Math. **3** (2000), 140–190 (electronic). MR 2001j:03022

[FR97] Marcelo P. Fiore and Giuseppe Rosolini, *Two models of synthetic domain theory*, J. Pure Appl. Algebra **116** (1997), no. 1-3, 151–162, Special volume on the occasion of the 60th birthday of Professor Peter J. Freyd. MR 98j:18003

[Fri98] Harvey M. Friedman, *Finite functions and the necessary use of large cardinals*, Ann. of Math. (2) **148** (1998), no. 3, 803–893. MR 2002b:03108

[Fri04] Stefan Friedrich, *Topology*, 2004, http://afp.sourceforge.net/entries/Topology.shtml.

[FT91] W. M. Farmer and F. J. Thayer, *Two computer-supported proofs in metric space topology*, Notices of the American Mathematical Society **38** (1991), 1133–1138, http://imps.mcmaster.ca/doc/two-proofs.ps.

[Geua] Herman Geuvers, *Inconsistency of classical logic in type theory*, http://www.cs.kun.nl/~herman/note.ps.gz.

[Geub] ______, *On-line list of publications of Herman Geuvers*, http://www.cs.kun.nl/~herman/pubs.html.

[GGS99] Georg Gottlob, Etienne Grandjean, and Katrin Seyr (eds.), *Computer science logic, 12th international workshop, csl '98, annual conference of the eacsl, brno, czech republic, august 24-28, 1998, proceedings*, Lecture Notes in Computer Science, vol. 1584, Springer, 1999.

[GH98] David Griffioen and Marieke Huisman, *A comparison of PVS and Isabelle/HOL*, Theorem Proving in Higher Order Logics: 11th International Conference, TPHOLs '98 (Canberra, Australia) (Jim Grundy and Malcolm Newey, eds.), vol. 1479, Springer-Verlag, 1998, pp. 123–142.

[Ghi] *Ghilbert system*, http://www.ghilbert.org/.

[GHW+] Freek Geuvers, Jan Herman, Randy Wiedijk, Henk Zwanenburg, and Barendregt Pollack, *Fta : Fundamental theorem of algebra project*, http://www.cs.kun.nl/~freek/fta/.

[Gim98] E. Gimenez, *A tutorial on recursive types in Coq*, Distribué comme partie de la documentation du système Coq., 1998.

[Gir72] J.-Y. Girard, *Interpretation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieure*, Ph.D. thesis, Université Paris 7, 1972.

[GN02] Herman Geuvers and Milad Niqui, *Constructive reals in Coq: axioms and categoricity*, Types for proofs and programs (Durham, 2000), Lecture Notes in Comput. Sci., vol. 2277, Springer, Berlin, 2002, pp. 79–95. MR 2 044 532

[Gor] Michael J. C. Gordon, *Papers on hol and set theory*, Author's home page.

[Gor94] Michael J. C. Gordon, *Merging hol with set theory: preliminary experiments*, Tech. Report 353, University of Cambridge Computer Laboratory, 1994.

[Gor00] Mike Gordon, *From LCF to HOL: a short history*, Proof, language, and interaction, Found. Comput. Ser., MIT Press, Cambridge, MA, 2000, pp. 169–185. MR 1 773 455

[GPM96] Eduardo Giménex and Christine Pausin-Mohring (eds.), LNCS, vol. 1512, Aussois, France, Springer-Verlag, 1996.

[GPWZ02] Herman Geuvers, Randy Pollack, Freek Wiedijk, and Jan Zwanenburg, *A constructive algebraic hierarchy in Coq*, J. Symbolic Comput. **34** (2002), no. 4, 271–286, Integrated reasoning and algebra systems (Siena, 2001). MR 2003m:68190

[Gra] Krzysztof Grabczewski, *Home page*, http://www.phys.uni.torun.pl/˜kgrabcze/.

[GWZ00a] H. Geuvers, F. Wiedijk, and J. Zwanenburg, *Equational reasoning via partial reflection*, Theorem proving in higher order logics (Portland, OR, 2000), Lecture Notes in Comput. Sci., vol. 1869, Springer, Berlin, 2000, pp. 162–178. MR 1 879 747

[GWZ00b] Herman Geuvers, Freek Wiedijk, and Jan Zwanenburg, *A constructive proof of the fundamental theorem of algebra without using the rationals*, TYPES, 2000, pp. 96–111.

[Hal] Thomas Hales, *The flyspeck project fact sheet*, http://www.math.pitt.edu/˜thales/flyspeck/index.html.

[Hal97a] T. C. Hales, *Sphere packings. I*, Discrete Comput. Geom. **17** (1997), no. 1, 1–51. MR 97k:52025

[Hal97b] ______, *Sphere packings. II*, Discrete Comput. Geom. **18** (1997), no. 2, 135–149. MR 98h:52033

[Hal02] Thomas C. Hales, *A computer verification of the Kepler conjecture*, Proceedings of the International Congress of Mathematicians, Vol. III (Beijing, 2002) (Beijing), Higher Ed. Press, 2002, pp. 795–804. MR 2004a:52036

[Har] John Harrison, *John Harrison's home page*, http://www.cl.cam.ac.uk/users/jrh/.

[Har95] ______, *Metatheory and reflection in theorem proving: A survey and critique*, Technical Report CRC-053, SRI Cambridge, Millers Yard, Cambridge, UK, 1995, Available on the Web as `http://www.cl.cam.ac.uk/users/jrh/papers/reflect.dvi.gz`.

[Har96a] ______, *Formalized mathematics*, Technical Report 36, Turku Centre for Computer Science (TUCS), Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland, 1996, Available on the Web as `http://www.cl.cam.ac.uk/users/jrh/papers/form-math3.html`.

[Har96b] ______, *Proof style*, in Giménex and Pausin-Mohring [GPM96], pp. 154–172.

[Har00] John Harrison, *The HOL-Light manual*, 2000.

[Har01] John Harrison, *Complex quantifier elimination in HOL*, TPHOLs 2001: Supplemental Proceedings (Richard J. Boulton and Paul B. Jackson, eds.), Division of Informatics, University of Edinburgh, 2001, Published as Informatics Report Series EDI-INF-RR-0046. Available on the Web at `http://www.informatics.ed.ac.uk/publications/report/0046.html`, pp. 159–174.

[HEL] *Helm project.*

[Hen02] Dimitri Hendriks, *Proof reflection in Coq*, Journal of Automated Reasoning **29** (2002), no. 3, 277–307.

[Her] Hugo Herbelin, *A program from an A-translated impredicative proof of Higman's Lemma*, http://coq.inria.fr/contribs/higman.html.

[hol] *Hol4*, http://hol4.sourceforge.net.

[How87] Douglas J. Howe, *The computational behaviour of Girard's paradox*, in *Proceedings, Symposium on Logic in Computer Science* [Com87], pp. 205–214.

[How96] D. J. Howe, *Importing mathematics from HOL into Nuprl*, Ninth international Conference on Theorem Proving in Higher Order Logics TPHOL (Turku, Finland) (J. Von Wright, J. Grundy, and J. Harrison, eds.), vol. 1125, Springer Verlag, 1996, pp. 267–282.

[HR03] Thérèse Hardin and Renaud Rioboo, *Towards a higher reasoning level in formalized homological algebra*, 11th Symposium on the Integration of Symbolic Computation and Mechanised Reasoning (Calculemus) (Rome, Italy), Aracne Editrice S.R.L., 2003, pp. 84–88.

[HS00] Gérard Huet and Amokrane Saïbi, *Constructive category theory*, Proof, language, and interaction, Found. Comput. Ser., MIT Press, Cambridge, MA, 2000, pp. 239–275. MR 2001d:03024

[HT98] John Harrison and Laurent Théry, *A skeptic's approach to combining HOL and Maple*, J. Automat. Reason. **21** (1998), no. 3, 279–294. MR 1 656 868

[Hug] Herbelin Hugo, *The Theorem of Schroeder-Bernstein. Coq user-contribution*, http://coq.inria.fr/contribs-eng.html.

[IEE86] IEEE Computer Society, *Proceedings, symposium on logic in computer science*, Cambridge, Massachusetts, 16–18 June 1986.

[IEE88] IEEE Computer Society, *Proceedings, third annual symposium on logic in computer science*, Edinburgh, Scotland, 5–8 July 1988.

[IEE89] IEEE Computer Society Press, *Proceedings, fourth annual symposium on logic in computer science*, Asilomar Conference Center, Pacific Grove, California, 5–8 June 1989.

[IEE90] IEEE Computer Society Press, *Proceedings, fifth annual ieee symposium on logic in computer science*, Philadelphia, Pennsylvania, 4–7 June 1990.

[IEE91] IEEE Computer Society Press, *Proceedings, sixth annual ieee symposium on logic in computer science*, Amsterdam, The Netherlands, 15–18 July 1991.

[IEE92] IEEE Computer Society Press, *Proceedings, seventh annual ieee symposium on logic in computer science*, Santa Cruz, California, 22–25 June 1992.

[IEE93] IEEE Computer Society Press, *Proceedings, eighth annual ieee symposium on logic in computer science*, Montreal, Canada, 19–23 June 1993.

[IEE94] IEEE Computer Society Press, *Proceedings, ninth annual ieee symposium on logic in computer science*, Paris, France, 4–7 July 1994.

[IEE95] IEEE Computer Society Press, *Proceedings, tenth annual ieee symposium on logic in computer science*, San Diego, California, 26–29 June 1995.

[IEE96] IEEE Computer Society Press, *Proceedings, 11*th *annual ieee symposium on logic in computer science*, New Brunswick, New Jersey, 27–30 July 1996.

[IEE97] IEEE Computer Society Press, *Proceedings, twelth annual ieee symposium on logic in computer science*, Warsaw, Poland, 29 June–2 July 1997.

[IMP] *IMPS system*, http://imps.mcmaster.ca/.

[Isa] *Isabelle*, http://www.cl.cam.ac.uk/Research/HVG/Isabelle/.

[Jer] Avigad Jeremy, *Mathematics in isabelle*, http://www.andrew.cmu.edu/user/avigad/isabelle/.

[JFM] *Journal of formalized mathematics*, http://mizar.org/JFM/.

[Jut77] L.S. van Benthem Jutting, *Checking Landau's "Grundlagen" in the AUTOMATH system*, Ph.D. thesis, Eindhoven University of Technology, 1977.

[KC86] Todd B. Knoblock and Robert L. Constable, *Formalized metareasoning in type theory*, in *Proceedings, Symposium on Logic in Computer Science* [IEE86], pp. 237–248.

[KCH04] Hidetsune Kobayashi, L. Chen, and Murao H., *Group-ring-module*, 2004, http://afp.sourceforge.net/entries/Group-Ring-Module.shtml.

[Ken] *The Kenzo program*, http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/.

[KM98] M. Kaufmann and J Moore, *A precise description of the ACL2 logic*, April 1998, http://www.cs.utexas.edu/users/moore/publications/km97a.ps.gz.

[KM01] Matt Kaufmann and J. Strother Moore, *Structured theory development for a mechanized logic*, Journal of Automated Reasoning **26** (2001), no. 2, 161–203.

[Kol96] Gina Kolata, *Computer math proof shows reasoning power*, New York Times (1996), December 10.

[KP99] Florian Kammüller and Lawrence C. Paulson, *A formal proof of Sylow's theorem. An experiment in abstract algebra with Isabelle HOL*, J. Automat. Reason. **23** (1999), no. 3-4, 235–264. MR 2000j:68166

[Lam] Leslie Lamport, *Types considered harmful, or types are not harmless*, This appeared under the first title in a posting by P. Rudnicki on Qed, Volume 2, Aug. 1994. A revised version with the second title appeared as a technical report. A balanced discussion presenting both points of view is in the next reference.

[LEG] *LEGO system*, http://www.dcs.ed.ac.uk/home/lego/.

[Les90] Pierre Lescanne, *Well rewrite orderings*, in *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science* [IEE90], pp. 249–256.

[Li02] Tatsien Li (ed.), *Proceedings of the International Congress of Mathematicians. Vol. II*, Beijing, Higher Education Press, 2002, Invited lectures, Held in Beijing, August 20–28, 2002. MR 2003i:00010a

[LP99] Leslie Lamport and Lawrence C. Paulson, *Should your specification language be typed*, ACM Trans. Program. Lang. Syst. **21** (1999), no. 3, 502–526.

[Luo89] Zhaohui Luo, **ECC**, *an extended calculus of constructions*, in *Proceedings, Fourth Annual Symposium on Logic in Computer Science* [IEE89], pp. 386–395.

[Luo90] ———, *An extended calculus of constructions*, Ph.D. thesis, University of Edinburgh, 1990.

[MA88] Paul F. Mendler and Peter Aczel, *The notion of a Framework and a framework for LTC*, in *Proceedings, Third Annual Symposium on Logic in Computer Science* [IEE88], pp. 392–399.

[McC97] W. McCune, *Solution of the Robbins problem*, Journal of Automated Reasoning (1997), http://www-unix.mcs.anl.gov/ mccune/papers/robbins/.

[Meta] *Metamath system*, http://www.metamath.org/.

[Metb] *MetaPRL*, http://cvs.metaprl.org:12000/metaprl/default.html.

[Mil78] R. Milner, *A theory of type polymorphism in programming*, Journal of Computing System Science **17** (1978), 348–375.

[Miz] *Mizar system*, http://mizar.uw.bialystok.pl/.

[ML75] Per Martin-Löf, *About models for intuitionistic type theories and the notion of definitional equality*, Proceedings of the Third Scandinavian Logic Symposium (Univ. Uppsala, Uppsala, 1973) (Amsterdam), North-Holland, 1975, pp. 81–109. Stud. Logic Found. Math., Vol. 82. MR 52 #10376

[ML84] ———, *Intuitionistic type theory*, Studies in Proof Theory. Lecture Notes, vol. 1, Bibliopolis, Naples, 1984, Notes by Giovanni Sambin. MR 86j:03005

[Mog89] Eugenio Moggi, *Computational lambda-calculus and monads*, in *Proceedings, Fourth Annual Symposium on Logic in Computer Science* [IEE89], pp. 14–23.

[MS00] W. McCune and O. Shumsky, *Ivy: A preprocessor and proof checker for first-order logic*, Computer-Aided Reasoning: ACL2 Case Studies (M. Kaufmann, P. Manolios, , and J Moore, eds.), Kluwer, 2000, Chapter 16.

[MW02] F. Wiedijk M. Wenzel, *A comparison of Mizar and Isar*, J. Automated Reasoning **29** (2002), 389–411.

[Nee02] Amnon Neeman, *A counterexample to a 1961 "theorem" in homological algebra*, Invent. Math. **148** (2002), no. 2, 397–420, With an appendix by P. Deligne. MR 2003d:18021

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel, *Isabelle/HOL — A proof assistant for Higher-Order Logic*, LNCS, vol. 2283, Springer, 2002.

[NuPa] *NuPRL system*, http://www.cs.cornell.edu/Info/Projects/NuPrl/nuprl.html.

[NuPb] *Nuprl library*, http://www.cs.cornell.edu/Info/Projects/NuPrl/Nuprl4.2/Libraries/.

[O'C] Russell O'Connor, *Proof of Gödel's first incompleteness theorem in Coq*, http://math.berkeley.edu/~roconnor/godel.html.

[Oes00] Joseph Oesterlé, *Densité maximale des empilements de sphères en dimension 3 (d'après Thomas C. Hales et Samuel P. Ferguson)*, Astérisque (2000), no. 266, Exp. No. 863, 5, 405–413, Séminaire Bourbaki, Vol. 1998/99. MR 2001k:52030

[O'K04] Greg O'Keefe, *Towards a readable formalisation of category theory*, Electronic Notes in Theoretical Computer Science **91** (2004), 212–228.

[Ott] *Otter: An automated deduction system*, http://www-unix.mcs.anl.gov/AR/otter/.

[Pad96] William Padmanabhan, R.; McCune, *Automated deduction in equational logic and cubic curves*, SPRINGER VERLAG, 1996.

[Paua] Paul Jackson Home Page, http://www.dcs.ed.ac.uk/home/pbj/.

[Paub] Lawrence C. Paulson, *Isabelle projects*, http://www.cl.cam.ac.uk/Research/HVG/Isabelle/projects.html.

[Pau03] ______, *The relative consistency of the axiom of choice mechanized using Isabelle/ZF*, LMS J. Comput. Math. **6** (2003), 198–248 (electronic), Appendix A available electronically at http://www.lms.ac.uk/jcm/6/lms2003-001/appendix-a/. MR 2 051 585

[Pfe] Frank Pfenning, *Bibliography on logical frameworks*, http://www-2.cs.cmu.edu/afs/cs/user/fp/www/lfs-bib.html.

[Pfe01] Frank Pfenning, *Logical frameworks*, Handbook of Automated Reasoning, 2001, pp. 1063–1147.

[PG] *Proof General*, http://proofgeneral.inf.ed.ac.uk/.

[PG96] Lawrence C. Paulson and Krzysztof Grąbczewski, *Mechanizing set theory. Cardinal arithmetic and the axiom of choice*, J. Automat. Reason. **17** (1996), no. 3, 291–323. MR 98b:03019

[PM96] R. Padmanabhan and W. McCune, *An equational characterization of the conic construction on cubic curves*, Lecture Notes in Computer Science (AI subseries), no. 1095, Springer-Verlag, 1996.

[PVS] *PVS (Proof Verification System)*, http://pvs.csl.sri.com/.

[QED94] *The QED manifesto*, Proceedings of the 12th International Conference on Automated Deduction, Springer-Verlag, 1994, pp. 238–251.

[Raf] C. Raffalli, *The PhoX proof assistant*, http://www.lama.univ-savoie.fr/sitelama/Membres/pages_web/RAFFALLI/af2.html.

[RAHM00] J.L. Ruiz Reina, J.A. Alonso, M. J. Hidalgo, and F.J. Martín, *A mechanical proof of Knuth-Bendix critical pair theorem (using ACL2)*, Proceedings of the Third International Workshop on First-Order Theorem Proving (FTP 2000) (St. Andrews, Scotland), 2000, Published as Technical Report 5-2000, Fachberichte Informatik, Universitat Koblenz-Landau.

[RD02] Christophe Raffalli and René David, *Computer assisted teaching in mathematics*, Workshop on 35 years of Automath (Avril 2002, Edingurgh), 2002, to appear.

[RR90] Edmund Robinson and Giuseppe Rosolini, *Polymorphism, set theory, and call-by-value*, in *Proceedings, Fifth Annual IEEE Symposium on Logic in Computer Science* [IEE90], pp. 12–18.

[Sai98] A Saibi, *Algèbre constructive en théorie des types, outils génériques pour la modélisation et la démonstration, application à la théorie des catégories*, Ph.D. thesis, Université Paris VI, 1998.

[Sco82] Dana S. Scott, *Domains for denotational semantics*, Proceedings of the 9th Colloquium on Automata, Languages and Programming, Springer-Verlag, 1982, pp. 577–613.

[SH] T. Streicher and M. Hofmann, *Lifting grothendieck universes and universes in toposes*, Preprint http://www.mathematik.tu-darmstadt.de/~streicher/NOTES/lift.dvi.gz.

[Sima] Carlos T. Simpson, *Computer theorem proving in math.*

[Simb] ______, *Set-theoretical mathematics in Coq.*

[Sim99] Alex Simpson, *Computational adequacy in an elementary topos*, in Gottlob et al. [GGS99], pp. 323–342,.

[Str] T. Streicher, *Lifting grothendieck universes*, Preprint: http://www.mathematik.tu-darmstadt.de/~streicher/NOTES/UniTop.ps.gz.

[The] *Theorema: Computer-supported mathematical theorem proving*, http://www.theorema.org/.

[Thé01] Laurent Théry, *A machine-checked implementation of Buchberger's algorithm*, J. Automat. Reason. **26** (2001), no. 2, 107–137. MR 2001k:68157

[TPS] *TPS (Theorem Proving System)*, http://gtps.math.cmu.edu/tps.html.

[Val03] Silvio Valentini, *A Cartesian closed category in Martin-Löf's intuitionistic type theory*, Theoret. Comput. Sci. **290** (2003), no. 1, 189–219. MR 2004b:03018

[Wer94] B. Werner, *Une théorie des constructions inductives*, Ph.D. thesis, Université Paris VII, Mai. 1994.

[Wer97] ______, *Sets in types, types in sets*, TACS'97 (Martin Abadi and Takahashi Ito, eds.), vol. 1281, LNCS, Springer-Verlag, 1997.

[Wie99] Freek Wierdijk, *Mizar: An impression*, http://www.cs.kun.nl/~freek/mizar/mizarintro.pdf, 1999.

[Wie01] Freek Wiedijk, *Mizar light for HOL light*, Lecture Notes in Computer Science **2152** (2001), 378–393.

[Wie03] Freek Wiedijk, *Comparing mathematical provers*, Mathematical Knowledge Management (Andrea Asperti, Bruno Buchberger, and James Davenport, eds.), Lecture Notes on Computer Science, no. 2594, Springer, 2003, Proceedings of MKM 2003, pp. 188–202.

[WUF02] Larry Wos, Dolph Ulrich, and Branden Fitelson, *Vanquishing the xcb question: The methodological discovery of the last shortest single axiom for the equivalential calculus*, Journal of Automated Reasoning **29** (2002), no. 2, 107–124, http://fitelson.org/xcb.pdf.

[Yar] *Yarrow system*, http://www.cs.kun.nl/˜janz/yarrow/.

[Zam97] Vincent Zammit, *A comparative study of Coq and HOL*, Proceedings of the 10th International Conference on Theorem Proving in Higher Order Logics, TPHOLs'97, Murray Hill, NJ, USA (Elsa L. Gunter and Amy Felty, eds.), Lecture Notes in Computer Science, vol. 1275, Springer, August 1997, http://www.cs.ukc.ac.uk/pubs/1997/394, pp. 323–337.

[Z/E] *Z/EVES system*, http://www.ora.on.ca/z-eves/.