ProofCheck: Writing and checking complete proofs in LaTeX

Bob Neveln and Bob Alps

Abstract

ProofCheck is a system for writing and checking mathematical proofs. Theorems and proofs are contained in a plain TeX or LaTeX document. Parsing and proof checking are accomplished through Python programs which read the source file. A general explanation of the use and structure of the system and programs is provided and a sample proof is shown in detail. The work done by the authors has been based on standard sentence logic, a non-standard predicate logic and set theory with proper classes. Theorems and proofs based on other foundations may be checked if external data files are modified. Four such data files and their possible modifications are described. In addition, the extent to which the formal language can be shaped to accommodate an author's preferences is discussed.

1 Introduction

The purpose of ProofCheck is to enable mathematicians to write readable proofs that are computer checkable. Such readable, computer-checkable proofs could also be of value in the refereeing process. In a previous article [4] the system now called ProofCheck was described. Since then the system has been extended in several ways. The number of inference rules and common notions has been increased. A web site at `www.proofcheck.org` has been developed. Further, the system now works with LaTeX in addition to plain TeX.

2 Other systems

Two well-known systems to which ProofCheck might be compared are HOL [2] and Mizar [5].

HOL (Higher Order Logic) is a computer-assisted theorem proving system operating within the OCAML environment. OCAML (Objective CAML) is an object-oriented programming language which can be run interactively. Mathematical objects are treated as OCAML objects and mathematical theorems are stated in OCAML. Thus both ontology and syntax are subsumed by OCAML type theory. Theorems are proved by entering commands at the OCAML prompt or by running a script. Thus a proof consists of a record of OCAML commands.

Mizar is also a language for stating proofs intended to be human-readable as well as computer-checkable. Proofs are entered in ASCII text in the Mizar language. The language is extensive and based on a particular axiomatization of mathematics, Tarski-Grothendieck set theory. The system can also produce LaTeX output.

These systems also tend to form closed mathematical systems. Proving a theorem using these systems means showing that the given theorem is provable within that system.

3 Features and goals

TeX and LaTeX convert an author's `.tex` source files into DVI or PDF output. ProofCheck works with these same `.tex` source files. When using ProofCheck the work cycle is typically to run the parser and the checker after a successful run of TeX or LaTeX. Errors encountered in each case throw the author back into the text editor.

ProofCheck is intended to be open with respect to mathematical foundations. When ProofCheck checks the proof of a theorem it shows that the theorem follows from definitions and other theorems which have been stated and parsed but not necessarily checked. An author does not need to commit to a particular axiom system.

In their own mathematical work the authors use standard sentence logic, a non-standard predicate logic, and a set theory which admits proper classes — but none of these choices is required. Of course checkable-proofs are easier to write when there is an accumulated body of accepted propositions available for referral. This does constitute an implicit pressure to use the specific development already on the ProofCheck web-site. But nothing prevents an author from creating another one. We will post any such developments we receive.

4 Mathematical language

For almost a century there has been general agreement that there is no obstacle in principle to writing mathematics in a formal language and therefore to checking proofs mechanically. One of the main obstacles in practice to stating definitions and theorems in a formal language is that the required sacrifice of syntactical freedom may be more than a mathematical author is willing to tolerate. In devising a usable proof-checking system therefore it is important to maximize syntactic freedom.

ProofCheck is not built on any specific mathematical language. Instead, a context-free grammar is generated on the basis of whatever definitions have been made. The rules of grammar are based on syntactical ideas of A.P. Morse [3], and include a variation on Morse's handling of second-order variables. Definitions are presumed to take the form $(p \leftrightarrow q)$ for formulas and $(x \equiv y)$ for terms. The set of definitions is taken to include a standard default set of infix operators as well as quantifiers. The default symbols may be easily replaced and with a little additional effort, default forms can be replaced. The resulting language is unambiguous and possesses the property that no term or formula can begin another.

ProofCheck minimizes the loss of syntactical freedom by striving to keep the syntactical restrictions as close as possible to the absolute requirements of logic and consistency.

5 Proof language

The following TeX macros suffice to mark up a proof for checking. We are grateful to Karl Berry for his help in streamlining the first two of these. The general proof structure which these macros implement is very similar to that defined in [1].

`\prop` Any proposition, whether a theorem, a definition or an axiom, must be introduced by this macro. Its use requires ProofCheck to look for a proof. Its syntax is `\prop`, followed by the enumeration, followed by the proposition. For example:

```
\prop 14.7 $(1 < 2)$
```

The `\prop` macro must be at the beginning of a line. The enumeration is of the form $n.m$ where n and m are positive integers with n representing the section or chapter number. The proposition must be enclosed within TeX dollar signs. References in a proof to other theorems use the same $n.m$ style and not LaTeX labels. When re-arrangement of theorems necessitates renumbering, this is handled by a ProofCheck program called `renum`. This program recognizes LaTeX section macros so that unless the section counter is reset manually, consistency with LaTeX sectioning is maintained.

`\note` This macro is used to introduce an assertion within the proof which can be referred to later in the proof. The assertion may be the result of telescoping multiple lines of proof, each having an optional justification. The ProofCheck syntax is `\note`, followed by the note enumeration, followed by the (possibly multi-line) assertion. Here the enumeration consists of a positive integer.

`\By` This macro is used at the end of a line to introduce justification for the step. A note with a single line of assertion may be justified by one of the following:

`\By G` (Given) signals an assertion which is to be accepted locally as a hypothesis. It initiates a block of the proof within which this hypothesis is in effect.

`\By S` (Set) indicates that the note is used to locally define a variable for use in the proof.

`\By .n H .m` (Hence) where n is the number of a note established using note m as a Given, ends the block initiated by note m. Such blocks are called Given-Hence blocks. A Hence justification is typically used to establish an assertion such as $(p \rightarrow q)$ where p is note m and q is note n.

Notes with either a single line or more than one line of assertion may be justified using other notes and a `.m` enumeration or other propositions and `n.m` enumeration with punctuation as shown in the sample proof. More discussion on this syntax may be found in [4].

`\Bye` in addition to introducing a justification, signals the end of a proof and prints "Q.E.D.".

`\linea, \lineb, ...` These begin a new line with increasing degrees of indentation. They are not proof macros per se but are used for any mathematical expressions that need to go beyond one line and need indentation. At present the parser allows only these.

In section 7 the use of these macros in a sample proof is shown.

6 Sample proof: Reader view

The simple proof in figure 1 is taken from a development of the von Neumann model of the natural numbers, ω, in which each natural number is the set of the preceding natural numbers. The theorem asserts that if y is an element of a natural number x then y is a subset of x.

The first line of the proof defines a set A. This note should be easy to read except for possibly the quantifier notation and the classifier notation. Notational changes are discussed in section 9. The second note translates the definition in note 1 into a bi-conditional which is much more useful deductively. We often refer to steps that turn a definition into one or more implications as "unwrapping" steps. Explicit inclusion of such unwrapping steps is often key in getting a proof to check. The stage is set for an induction proof.

The theorem 4.7 referred to is just the standard induction theorem:

$$(\emptyset \in A \wedge \bigwedge x \in A(x \in A \rightarrow \operatorname{scsr} x \in A) \rightarrow \omega \subset A)$$

where "scsr x" denotes the successor of x. Its two hypotheses are the base case which in this proof is established in note 3 and the universalization of the

Theorem
 4.8 $(y \in x \in \omega \to y \subset x)$

Proof: To prove this by induction we begin by letting A be the set of all x such that each element y of x is a subset of x. We set

.1 $(A \equiv ⋿ x \bigwedge y \in x(y \subset x))$ ‡S

It will follow from 4.7 that ω is a subset of A. First we unwrap .1.

.2 $(x \in A \leftrightarrow \bigwedge y \in x(y \subset x) \wedge x \in \mathrm{U})$ ‡08.3;.1

Base Case $(\emptyset \in A)$.

.3 $(\emptyset \in A)$ ‡.2;09.19,09.12

Induction Step $(x \in A \to \mathrm{scsr}\, x \in A)$.

Given

.4 $(x \in A)$ ‡G

We note first that

.5 $(x \in \mathrm{U})$ ‡09.20;.4

.6 $\bigwedge y \in x(y \subset x)$ ‡.2;.4

Then we have

.7 $(y \in \mathrm{scsr}\, x \to y \in x \vee y = x$ ‡3.7
$\to y \subset x \vee y = x$ ‡.6
$\to y \subset x$ ‡011.14
$\to y \subset \mathrm{scsr}\, x)$ ‡011.10;(
3.5;(09.20;.4))

So we can conclude that

.8 $(\mathrm{scsr}\, x \in A)$ ‡.2;(3.3;.5),(.7 U)

Hence

.9 $(x \in A \to \mathrm{scsr}\, x \in A)$ ‡.8 H .4

This completes the proof that

.10 $(\omega \subset A)$ ‡4.7;.3,(.9 U)

The conclusion now follows quickly.

.11 $(y \in x \wedge x \in \omega \to y \in x \wedge x \in A$ ‡011.7;.10
$\to y \in x \wedge \bigwedge y \in x(y \subset x)$ ‡.2
$\to y \subset x)$

Q.E.D. .11

Figure 1: Sample proof: DVI output

induction step established in note 9. This theorem is invoked in the justification of note 10. In the proof of the induction step, note 7 shows that any member y of scsr x is a subset of scsr x. In note 8, we conclude that scsr x is in A. In note 9 we join the hypothesis from note 4 to the conclusion obtained in note 8.

Note 11 details the step from note 10 to the theorem which is short but cannot be skipped. The "Q.E.D. .11" at the end asserts that the theorem itself follows from note 11.

7 Sample proof: Author view

In Figure 2 we have the LaTeX source code for the sample proof.

The first couple of lines of the sample proof begin explaining the proof. Since they are not noted they do not contribute to the check. But neither do they get in the way of the check. Unchecked text of any sort is admissible so long as it does not interrupt mathematical expressions or interfere with proof specification.

The reader will note that all the macros and note justifications described in section 5 are used in this sample proof. There is a Set statement in note 1, a Given-Hence block in notes 4 through 9, and the proof terminates with a `\Bye` macro.

Note 4 opens a Given-Hence block and establishes $(x \in A)$ as a working hypothesis. Note 4 may be referred to only within this Given-Hence block. A Hence justification may close more than one Given note, but each Given note must be explicitly closed by a Hence justification. This Given-Hence block is closed by note 9.

Note 7 is a multi-line note each line of which has a justification. A reference to note 7 accesses the telescoped result of the note which is

$$(y \in \mathrm{scsr}\, x \to y \subset \mathrm{scsr}\, x)$$

The `\Bye` line is justified by note 11. The telescoped note 11 and the statement of the theorem differ only notationally on the left side of the implication. A supplemental parser produces a canonical version of each of the two left sides. These two versions turn out to be identical. Consequently the rule of inference used here merely allows one to infer p from p.

8 Proof checking

Each assertion within a note is checked individually using end-of-line references to theorems and/or notes. Each of these checks is done by submitting a formula and the formulas referred to in its justification to a rule matcher which conducts a simple linear search of the list of inference rules. The search must find a rule which unifies with the submitted formulas in order for the check of the assertion to succeed.

In Figure 3 the rectangular boxes represent TeX files whereas the unifier and the rule matcher are Python scripts.

9 Working with ProofCheck files

The files `rules.tex` and `common.tex` contain the rules of inference and common notions used as the defaults for optional command line parameters of the checking script. In adapting the common notions to suit a particular mathematical interest, the file `common.tex` may be modified or another file altogether may be written. The same applies to the file `rules.tex` should the use of another logic be required. Both of these files require the inclusion of many redundant forms of whatever principles are

```
\noindent{}Theorem

\prop 4.8 $(y \in x \in \omega \c y \subset x)$
    \lineb Proof:  To prove this by induction we begin by
letting $A$ be the set of all $x$ such that each element $y$ of $x$
is a subset of $x$.  We set
\note 1 $(A \ident \setof x \Each y\in x(y\subset x))$ \By S
\linea It will follow from 4.7 that $\omega$ is a subset of $A$. First we unwrap .1.
\note 2 $(x \in A \Iff \Each y \in x (y \subset x) \And x \in \U)$ \By 08.3;.1
    \lineb Base Case $(\e \in A )$~.
\note 3 $(\e \in A)$ \By .2;09.19,09.12
    \lineb Induction Step $(x \in A \c \scsr x \in A)$~.
\linea Given
\note 4  $(x \in A )$  \By G
\linea We note first that
\note 5 $(x \in \U)$  \By 09.20;.4
\note 6 $\Each y \in x(y \subset  x)$\By .2;.4
\linea Then we have
\note 7 $(y \in \scsr x \c y \in x \Or y = x$  \By 3.7
    \lined $\c y \subset x \Or y = x$  \By .6
    \lined $\c y \subset x $  \By 011.14
    \lined $\c y \subset \scsr x )$ \By 011.10;( 3.5;(09.20;.4))
\linea So we can conclude that
\note 8 $(\scsr x \in A)$ \By .2;(3.3;.5),(.7 U)
\linea Hence
\note 9 $(x \in A \c \scsr x \in A)$ \By .8 H .4
\linea This completes the proof that
\Note 10 $(\omega \subset A)$ \By 4.7;.3,(.9 U)
\linea The conclusion now follows quickly.
\Note 11 $(y \in x \And x \in \omega\c y \in x \And x \in A$ \By 011.7;.10
    \lined $\c y \in x \And \Each y \in x(y \subset x)$ \By .2
    \lined $\c y \subset x)$
    \lineb \Bye .11
```

Figure 2: Sample proof: LaTeX input

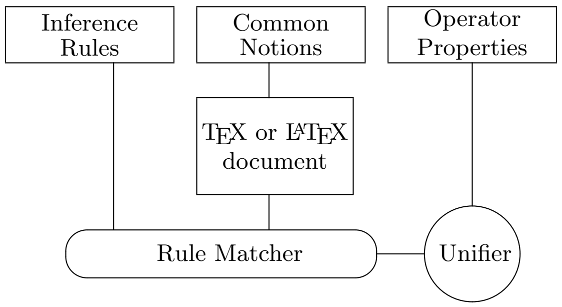

Figure 3: Main proof checking files

included, because in application many difference variations present themselves for unification. Additions contrived with a single application in mind should of course be avoided — making this distinction sometimes requires a non-trivial judgment. Building up these files is time consuming.

On the other hand, changes consisting simply of substituting one symbol for another are easily accommodated.

Common notions Common notions comprise all the definitions and theorems outside of the current working document that are needed to prove the theorems in the document. Two files `common.tex` and `common.ldf` are used to store information about common notions. The file `common.ldf` stores TeX macros to represent various constant symbols used to state the common notions, whereas `common.tex` stores the definitions and theorems themselves. If for example an author wished to use the quantifier '$\forall$' instead of '$\bigwedge$' the definition of `\Each` in this file could be changed as follows:

```
\def\Each{\mathop{\forall}}
```

This would change the output but still require the use of the `\Each` macro in the source file.

Rules of inference Rules of inference include basic rules such as modus ponens and universalization. The role played by rules of substitution is subsumed by the unifier. The present authors have supplemented the basic rules with over 1000 additional rules. All the additional rules are consequences of the logic we use. In all but the most elementary settings, such an expansion of the rule set is essential to keeping proofs under a reasonable length. This file may be populated according to the author's accepted logic. Each entry of the rules of inference file begins with the formula to be proved, followed by `<=`, followed by the formulas needed to prove it. The entry for modus ponens has the following form:

$$q <= (p \rightarrow q)\ ;\ p$$

Math and Logic Symbols The file `equivmacros.trf` consists of a list of macro replacements made prior to sending a term or formula from the author's document to the parser. For example an author who wanted to use `\forall` as the universal quantifier in the source file could include a line in `equivmacros.trf` such as

```
forall Each
```

We will be happy to post any modifications of files as described above on the ProofCheck website.

10 Conclusion

ProofCheck is a very simple system. As shown in figure 3 it consists mainly of rules of inference, a store of assumed elementary propositions, a slightly enhanced unifier and a rule matching script. The size of a download of the complete system from `www.proofcheck.org` is less than one megabyte. The discussion of the proof language in section 5 approaches a complete tutorial. We believe that the fact that proofs can be checked with such a simple system confirms the basic ideas on which it is based.

At the conclusion of [4] we asserted that complete proofs done using ProofCheck required approximately one order of magnitude more time to write than a conventional proof, and about two or three times as much space. Experience since then does not lead us to revise these estimates significantly, although the length of proofs has diminished slightly due to the growth in the number of rules of inference. We anticipate further progress.

References

[1] W. W. Bledsoe and E.J. Gilbert. Automatic theorem proof-checking in set theory. Technical Report SC-RR-67-525, Sandia Laboratories, July 1967.

[2] M. J. C. Gordon and T. F. Melham. *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic.* Cambridge University Press, Cambridge, 1993.

[3] A. P. Morse. *A Theory of Sets.* Academic Press, second edition, 1986.

[4] Bob Neveln and Bob Alps. Writing and checking complete proofs in TEX. *TUGboat* 28(1), 80–83, 2007.

[5] Piotr Rudnicki and Andrzej Trybulec. A collection of TEXed mizar abstracts. Technical Report TR 89-18, University of Alberta, June 1989. `www.mizar.org/project/bibliography.html`.