CARNEGIE MELLON

ETPS: A System to Help Students Write Formal Proofs

by

Peter B. Andrews
Carnegie Mellon University

Matthew Bishop
Azlan Ltd.

Chad E. Brown
Carnegie Mellon University

Sunil Issar

Frank Pfenning
Carnegie Mellon University

Hongwei Xi
Boston University

Research Report No. 03-002
April 2003

Department of Mathematical Sciences
Carnegie Mellon University

ETPS: A System to Help Students Write Formal Proofs

Peter B. Andrews*
Carnegie Mellon University

Matthew Bishop†
Azlan Ltd

Chad E. Brown‡
Carnegie Mellon University

Sunil Issar

Frank Pfenning§
Carnegie Mellon University

Hongwei Xi¶
Boston University

Abstract

ETPS (Educational Theorem Proving System) is a program which logic students can use to write formal proofs in first-order logic or higher-order logic. It enables students to concentrate on the essential logical problems involved in proving theorems, and automatically checks the proofs.

1 Introduction

Students in a variety of disciplines, especially those with a mathematical flavor, often need to learn how to write correct and rigorous proofs. This often creates substantial challenges for both the students and their teachers. A student may easily commit a fundamental logical error in a proof which makes the rest of the proof useless, so it is important that a student receive feedback about errors as promptly as possible. However, reading or grading proofs for a large class of students can be a very demanding and tedious task.

The ETPS (Educational Theorem Proving System) program can be used to make the processes of writing and checking formal proofs much easier and more enjoyable for logic students and their teachers.

The student using ETPS issues commands to apply rules of inference in specified ways, and the computer handles the details of writing the appropriate lines of the proof and checking that the rules can be used in this way. The program thus allows students to concentrate

*andrews@cmu.edu
†matt.bishop@azlan.co.uk
‡cebrown@andrew.cmu.edu
§fp@cs.cmu.edu
¶hwxi@cs.bu.edu

[0]The development of ETPS was supported by NSF grants MCS81-02870, DCR-8402532, CCR-8702699, CCR-9002546, CCR-9201893, CCR-9502878, CCR-9624683, CCR-9732312, CCR-0097179, and a grant from the Center for Design of Educational Computing of Carnegie Mellon University.

on the essential logical problems underlying the proofs, and it gives them immediate feedback for both correct and incorrect actions.

ETPS was developed in conjunction with the automated Theorem Proving System TPS [2], which has facilities for proving theorems automatically as well as the purely interactive facilities of ETPS. The two systems are distributed together, and are available from the web [5]. Details are discussed below.

After reviewing eight programs that support the teaching of logic, the authors of [4] (which was partially reprinted in [3]) concluded 'For elementary and advanced courses in mathematical logic for students with a formal background, we choose ETPS, a powerful tool that is also easy to learn.'

2 Environments for Running ETPS

ETPS can be run in an unspecialized environment, in an X-window on a Unix-based computer, or using a Java interface. There are two versions of the Java interface. One version provides a command line interface, while the other provides a point-and-click style interface with popup windows. The Java interfaces are still being improved.

3 Displaying Formulas and Proofs

ETPS offers the student a choice of styles for displaying logical formulas (wffs). The simplest style, which is called GENERIC, displays exercise X2110 (for example) on the screen as

```
          EXISTS x R x
        AND FORALL y [R y IMPLIES EXISTS z Q y z]
        AND FORALL x FORALL y [Q x y IMPLIES Q x x]
IMPLIES EXISTS x EXISTS y .Q x y AND R y .
```

If the student is using the X-window interface with the style XTERM or one of the Java interfaces with the style ISTYLE, ETPS displays exercise X2110 on the screen as

$$\exists x\, R\, x \wedge \forall y[\, R\, y \supset \exists z\, Q\, y\, z] \wedge \forall x \forall y[\, Q\, x\, y \supset\ Q\, x\, x]$$
$$\supset \exists x \exists y.\, Q\, x\, y \wedge\ R\, y.$$

Proofs created using ETPS are presented in *natural deduction* style. An example of such a proof is in Figure 1. The proof consists of a sequence of lines. Each line of the proof consists of a number in parentheses which serves as a label for that line and for the wff asserted in it, a list (possibly empty) of numbers of lines whose wffs are assumed as hypotheses for that line, the assertion sign $\vdash$, the wff asserted in that line, and a justification.

The TABLEAU command can be used to display the structure of the proof as a tree.

ETPS can handle higher-order logic (type theory) as well as first-order logic, and it has facilities for using definitions. These features enable a wide variety of theorems of mathematics and other disciplines to be expressed and proved in ETPS in very natural ways.

Each definition can have a *face*, which causes it to be displayed and printed using special characters. A definition of a binary operator can declare that operator to be an *infix* operator,

(1)	1	$\vdash$	$\exists x\, R\, x$	
			$\wedge \forall y[\, R\, y \supset \exists z\, Q\, y\, z]$	
			$\wedge \forall x \forall y.\, Q\, x\, y \supset\ Q\, x\, x$	Hyp
(10)	1	$\vdash$	$\exists x\, R\, x$	RuleP: 1
(11)	1,11	$\vdash$	$R\, a$	Choose: a 10
(20)	1	$\vdash$	$\forall y.\, R\, y \supset \exists z\, Q\, y\, z$	RuleP: 1
(21)	1	$\vdash$	$R\, a \supset \exists z\, Q\, a\, z$	UI: a 20
(22)	11,1	$\vdash$	$\exists z\, Q\, a\, z$	MP: 11 21
(23)	1,11,23	$\vdash$	$Q\, a\, b$	Choose: b 22
(30)	1	$\vdash$	$\forall x \forall y.\, Q\, x\, y \supset\ Q\, x\, x$	RuleP: 1
(31)	1	$\vdash$	$\forall y.\, Q\, a\, y \supset\ Q\, a\, a$	UI: a 30
(32)	1	$\vdash$	$Q\, a\, b \supset\ Q\, a\, a$	UI: b 31
(33)	1,11,23	$\vdash$	$Q\, a\, a$	MP: 23 32
(34)	1,11	$\vdash$	$Q\, a\, a$	RuleC: 22 33
(96)	1,11	$\vdash$	$Q\, a\, a \wedge\ R\, a$	Conj: 34 11
(97)	1,11	$\vdash$	$\exists y.\, Q\, a\, y \wedge\ R\, y$	EGen: a 96
(98)	1,11	$\vdash$	$\exists x \exists y.\, Q\, x\, y \wedge\ R\, y$	EGen: a 97
(99)	1	$\vdash$	$\exists x \exists y.\, Q\, x\, y \wedge\ R\, y$	RuleC: 10 98
(100)		$\vdash$	$\exists x\, R\, x$	
			$\wedge \forall y[\, R\, y \supset \exists z\, Q\, y\, z]$	
			$\wedge \forall x \forall y[\, Q\, x\, y \supset\ Q\, x\, x]$	
			$\supset \exists x \exists y.\, Q\, x\, y \wedge\ R\, y$	Deduct: 99

Figure 1: An ETPS proof of X2110

so that it is printed between its arguments. Thus, the assertion that the intersection of sets A and B is a subset of A is displayed as

$$A \cap B \subseteq A.$$

The commands SHOWTYPES and SHOWNOTYPES can be used to specify that wffs of higher-order logic will be displayed with or without type symbols. Thus exercise X5305 (a formulation of Cantor's Theorem that there is no function which maps a set onto its power set) can be displayed as

$$\forall s_{o\alpha}. \sim \exists g_{o\alpha\alpha} \forall f_{o\alpha}. f \subseteq s \supset \exists j_{\alpha}. s\, j \wedge g\, j = f$$

or as

$$\forall s. \sim \exists g \forall f. f \subseteq s \supset \exists j. s\, j \wedge g\, j = f.$$

The versatility of the display on the screen when running in an X-window is limited somewhat by the number of special characters available.

ETPS has commands for creating files which can be printed to display complete or incomplete proofs on paper. The command PRINTPROOF creates such a file in GENERIC style, while the command TEXPROOF creates a file which can be run through TEX to display the proof using the symbols of logic as illustrated above.

4 Proofwindows

The student using the Java or X-window interfaces can execute the command BEGIN-PRFW to create special *proofwindows* which display and automatically update the proof when the student executes commands to apply rules of inference. One of these proofwindows contains the entire current proof. Another proofwindow displays only the current subproof, which consists of the line of the proof which the student is currently trying to derive and the lines from which 'e[1] is trying to derive it. These lines are called *active*. This helps the student to focus on the immediate problem. A third proofwindow displays the active lines as well as the numbers of all other lines of the proof, so that the student can readily see where the active lines fit into the entire proof and where new lines can be inserted.

ETPS automatically updates the information about which lines of the proof are active. For example, when a student starts to prove X2110, the proof just contains the wff to be proved, as displayed in Figure 2. (The pseudo-justification PLAN1 simply indicates that this line has not yet been justified. Such lines are called *planned* lines.) If the student applies the DEDUCT rule to apply the Deduction Theorem (backwards) to prove line (100), the proof is modified to that displayed in Figure 3. At the same time, ETPS deletes (100) from the list of active lines, since it is now justified and need not be considered again; at this stage the current subproof window simply displays the lines shown in Figure 4.

Of course, the student will sometimes wish to change the set of active lines, and there are commands for doing this.

[1]Since there has long been a need for a gender-neutral personal pronoun in English, we shall write *'e* as an abbreviation for *he or she*, and *'e's* as an abbreviation for *his or her*.

	.....			
(100)		$\vdash$	$\exists x\, R\, x$	
			$\wedge \forall y[\, R\, y \supset \exists z\, Q\, y\, z]$	
			$\wedge \forall x \forall y[\, Q\, x\, y \supset\ Q\, x\, x]$	
			$\supset \exists x \exists y.\, Q\, x\, y \wedge\ R\, y$	PLAN1

Figure 2: Initial step in proving X2110

(1)	1	$\vdash$	$\exists x\, R\, x$	
			$\wedge \forall y[\, R\, y \supset \exists z\, Q\, y\, z]$	
			$\wedge \forall x \forall y.\, Q\, x\, y \supset\ Q\, x\, x$	Hyp
	.....			
(99)	1	$\vdash$	$\exists x \exists y.\, Q\, x\, y \wedge\ R\, y$	PLAN2
(100)		$\vdash$	$\exists x\, R\, x$	
			$\wedge \forall y[\, R\, y \supset \exists z\, Q\, y\, z]$	
			$\wedge \forall x \forall y[\, Q\, x\, y \supset\ Q\, x\, x]$	
			$\supset \exists x \exists y.\, Q\, x\, y \wedge\ R\, y$	Deduct: 99

Figure 3: Second step in proving X2110

(1)	1	$\vdash$	$\exists x\, R\, x$	
			$\wedge \forall y[\, R\, y \supset \exists z\, Q\, y\, z]$	
			$\wedge \forall x \forall y.\, Q\, x\, y \supset\ Q\, x\, x$	Hyp
	.....			
(99)	1	$\vdash$	$\exists x \exists y.\, Q\, x\, y \wedge\ R\, y$	PLAN2

Figure 4: Active lines for second step in proving X2110

	. . . . .			
(23)	1	$\vdash$	$Q\,a\,b$	PLAN7
(32)	1	$\vdash$	$Q\,a\,b \supset Q\,a\,a$	UI: b 31
(33)	1	$\vdash$	$Q\,a\,a$	MP: 23 32

Figure 5: An application of Modus Ponens

	. . . . .			
(23)	1	$\vdash$	$Q\,a\,b$	PLAN7
	. . . . .			
(33)	1	$\vdash$	$Q\,a\,a$	PLAN8

Figure 6: Modified portion of proof from Figure 5 after deletion of line (32)

---

5 Interacting with ETPS

Rules of inference can be applied very flexibly. Any or none of the lines referred to in the description of a rule of inference may already be present in the proof when the student executes that rule. ETPS will simply ask where the various lines are to be placed, and (if necessary) what wffs are to be asserted in them.

For example, consider the application of Modus Ponens in Figure 5. If MP is called when line (32) is present but lines (23) and (33) have not yet been put into the proof, ETPS will ask for the number of the Line with Implication (to which the student responds "32"), the number of the Line with Succedent of Implication (to which the student responds "33"), and the number of the Line with Antecedent of Implication (to which the student responds "23"), and ETPS adds lines (23) and (33) to the proof.

As a consequence of this flexibility in applying rules of inference, students can construct proofs working forwards, backwards, or in a combination of these modes.

The student can rearrange or modify a proof by renumbering lines, deleting lines, and adding new lines. If a deleted line was formerly used in the justification of another line, the formerly justified line automatically becomes a planned line. For example, the proof lines in Figure 5 are replaced by those in Figure 6 if line (32) is deleted.

Once the proof is complete, the student can execute the CLEANUP command to simultaneously delete all lines which are not actually needed in the proof. The SQUEEZE command can be used to renumber the lines consecutively, transforming the proof in Figure 1 to that in Figure 7.

When ETPS prompts the student for an argument to a command, it often provides a default suggestion, which is specified in brackets. If the student wishes to use this default, 'e simply hits the <Return> key. This speeds up interaction with ETPS considerably.

Online help is available for all ETPS commands. In particular, descriptions of the rules of inference are available online. If the student needs more information when prompted for an argument to a command, 'e can respond with ? or ??. If the command involves applying a rule of inference, in response to ? ETPS might specify that a line number or wff is needed, whereas in response to ?? it would state the rule of inference.

Sometimes students who are just starting to learn how to prove theorems have no idea

(1)	1	$\vdash$	$\exists x\, R\, x$	
			$\wedge \forall y[\, R\, y \supset \exists z\, Q\, y\, z]$	
			$\wedge \forall x \forall y.\, Q\, x\, y \supset\ Q\, x\, x$	Hyp
(2)	1	$\vdash$	$\exists x\, R\, x$	RuleP: 1
(3)	1,3	$\vdash$	$R\, a$	Choose: a 2
(4)	1	$\vdash$	$\forall y.\, R\, y \supset \exists z\, Q\, y\, z$	RuleP: 1
(5)	1	$\vdash$	$R\, a \supset \exists z\, Q\, a\, z$	UI: a 4
(6)	3,1	$\vdash$	$\exists z\, Q\, a\, z$	MP: 3 5
(7)	1,3,7	$\vdash$	$Q\, a\, b$	Choose: b 6
(8)	1	$\vdash$	$\forall x \forall y.\, Q\, x\, y \supset\ Q\, x\, x$	RuleP: 1
(9)	1	$\vdash$	$\forall y.\, Q\, a\, y \supset\ Q\, a\, a$	UI: a 8
(10)	1	$\vdash$	$Q\, a\, b \supset\ Q\, a\, a$	UI: b 9
(11)	1,3,7	$\vdash$	$Q\, a\, a$	MP: 7 10
(12)	1,3	$\vdash$	$Q\, a\, a$	RuleC: 6 11
(13)	1,3	$\vdash$	$Q\, a\, a \wedge\ R\, a$	Conj: 12 3
(14)	1,3	$\vdash$	$\exists y.\, Q\, a\, y \wedge\ R\, y$	EGen: a 13
(15)	1,3	$\vdash$	$\exists x \exists y.\, Q\, x\, y \wedge\ R\, y$	EGen: a 14
(16)	1	$\vdash$	$\exists x \exists y.\, Q\, x\, y \wedge\ R\, y$	RuleC: 2 15
(17)		$\vdash$	$\exists x\, R\, x$	
			$\wedge \forall y[\, R\, y \supset \exists z\, Q\, y\, z]$	
			$\wedge \forall x \forall y[\, Q\, x\, y \supset\ Q\, x\, x]$	
			$\supset \exists x \exists y.\, Q\, x\, y \wedge\ R\, y$	Deduct: 16

Figure 7: Proof of X2110 with lines renumbered

what to do next. In such cases the ADVICE command in ETPS may be helpful. (The teacher determines whether ADVICE may be used on any particular problem.) When this command is executed, ETPS provides naive advice based on the form of the active planned line. If the student does not find this advice sufficiently helpful, 'e can execute the ADVICE command again and obtain more specific or different advice. For example, if the main connective of the active planned line is an implication as in Figure 2, repeated execution of the ADVICE command would yield the following dialogue:

<2>ADVICE
Some planned line is an implication.
<3>ADVICE
DEDUCT 100
<4>ADVICE
I can't see much beyond what I have already told you. Of course, one could always try an indirect proof.
<5>ADVICE
INDIRECT 100
<6>ADVICE
Using a lemma is usually unnecessarily complicated for simple problems. But if you have a good idea what that lemma should be, go ahead.
<7>ADVICE
LEMMA 100
<8>ADVICE
One can always introduce a new hypothesis. But it rarely turns out to be useful, because it's hard to get rid of it.
<9>ADVICE
HYP 100
<10>ADVICE
I don't have any other suggestions for planned line 100. If you want advice for another planned line use SUBPROOF. I will start over with my initial suggestions, if you call ADVICE again.

ETPS has a convenient formula editor which permits the student to extract formulas from anywhere in the proof, and build new formulas from them. Special editor windows (similar to proofwindows) display and update the wff being edited and the subformula of it which is the current focus of activity.

When wffs of higher-order logic are being typed in, it is not necessary to specify the types of all the variables. The student must simply specify enough type information to avoid ambiguity, and ETPS applies a type inference mechanism based on work by Robin Milner and Daniel Leivant to determine the types of the other variables.

ETPS puts the commands the student issues while working on an exercise in a file (called a *work* file), and these commands can be re-executed if the student is interrupted or for some other reason needs to restore 'e's work. Complete or incomplete proofs can also be saved in files, and these can be reloaded into ETPS whenever they are needed.

6 Experience with ETPS

ETPS has been used by students in certain logic courses at Carnegie Mellon University to construct natural deduction proofs since 1983. It has also been used at The University of Birmingham in England, and perhaps elsewhere. Students with a technical background generally learn to use ETPS fairly quickly just by reading parts of the ETPS Manual (which contains some complete examples of how to construct proofs in ETPS) and starting work on quite easy exercises.

It can be enlightening to watch how students use ETPS after they have done a number of exercises and developed habits for using the system. Sometimes students show remarkable creativity in developing surprisingly awkward ways of doing things. Even if the final proof of a theorem is very clean and well structured, it may have been developed in a very tortuous manner. Therefore, it is sometimes useful to give a demonstration of how to prove a theorem using ETPS after students have already proved this theorem, so that they can gain deeper insights into the process.

Our experience has shown that students generally perform better when attempting to prove difficult theorems with ETPS than they perform without ETPS. For example, one of the more challenging first-order exercises in ETPS is X2138:

$$\forall x \exists y\, F\, x\, y$$

$$\wedge \exists x \forall e \exists n \forall w [\, S\, n\, w \supset\ D\, w\, x\, e]$$
$$\wedge \forall e \exists d \forall a \forall b [\, D\, a\, b\, d \supset \forall y \forall z.\, F\, a\, y \wedge\ F\, b\, z \supset\ D\, y\, z\, e]$$
$$\supset \exists y \forall e \exists m \forall w.\, S\, m\, w \supset \forall z.\, F\, w\, z \supset\ D\, z\, y\, e$$

Typically, when this exercise is assigned to a class which must write out proofs by hand, only a few students manage to prove it correctly, but in a class using ETPS most of the students can prove it.

7 Customizing ETPS

The predefined exercises and most of the rules of inference in ETPS are taken from the textbook [1]. However, a teacher who wishes to set up a variant of ETPS with different exercises, rules of inference, or names for the rules of inference can do so. Adding exercises is trivial. To define new rules of inference, one uses the RULES module of TPS. One simply describes the new rules in a simple lisp meta-language (using the existing rules as examples), and uses commands in TPS to create the lisp code for executing these commands.

ETPS has a powerful rule of inference called Rule P which allows one to derive $\mathbf{B}$ from $\mathbf{A}_1$, ..., and $\mathbf{A}_n$ if $[\mathbf{A}_1\ \wedge \ldots \wedge\ \mathbf{A}_n\ \supset\ \mathbf{B}]$ is tautologous (a substitution instance of a tautology). Many rules of inference involving propositional connectives (such as Modus Ponens, Modus Tollens, the Transitive Law of Implication, Conjunction Elimination, and Conjunction Introduction) are special cases of Rule P. The teacher can decide whether or not Rule P may be used on any particular exercise.

8 Recording and Processing Grades

ETPS has facilities for automatically recording which exercises are completed by each student, and putting this information in a record of grades. This record can be processed by the GRADER program which is also part of ETPS. The GRADER program has a variety of features which make it useful for processing grades for any course, whether or not ETPS is used in that course.

9 Checking Prenex Normal Form Exercises

An additional utility which comes with ETPS is a facility for automaticaly checking exercises which ask students to put wffs into prenex normal form. Students need some practice at such manipulations, but such exercises are notoriously tedious to check. When this facility is used, the students submit their answers (in the form they have learned to use in writing wffs for ETPS) to the teacher by email. Each answer has a compound name which indicates which exercise is being done and which student submitted it. The teacher puts all these answers into one file, loads them into TPS, and executes a command called PRENEX-ITERATE. TPS then checks each answer to see whether it is indeed in prenex normal form, and uses its automatic proof procedure to try to prove that the answer is equivalent to the wff given in the problem. Finally, TPS composes a report on the results it obtains for all the submitted answers. Of course, the search procedure involves search bounds, so TPS may fail to prove a valid wff, but as a practical matter the student's answer is almost always incorrect if TPS fails to prove that it is equivalent to the given wff in the allotted time.

10 Obtaining ETPS

ETPS is available from the web site

`http://gtps.math.cmu.edu/tps.html`

From the same web site one can download (as postscript or pdf files) the ETPS User's Manual (which is intended for students and other users of ETPS), the GRADER Manual, and the TPS User's Manual (which contains information about setting up ETPS). Online documentation for ETPS is also available at this web site. ETPS has been used extensively under Unix and Linux systems, and to some extent under Windows.

ETPS is a Common Lisp program, so a person who wishes to have ETPS running on 'e's computer must have (or obtain) a Common Lisp compiler, and compile ETPS after downloading the source files from the web. Similarly, one must have a Java compiler on the machine if one wishes to use one of the Java interfaces for ETPS.

11 Conclusion

Experience has shown that ETPS is an effective teaching tool which makes it practical and fun for students to get significant practice at writing formal proofs. It helps both students and teachers to concentrate on the essentials of learning and teaching how to prove theorems.

12 Acknowledgements

We are grateful for contributions to the development of ETPS which resulted from work done on the TPS project by Dale Miller and Dan Nesmith. We thank Manfred Kerber and Christoph Benzmüller for helpful comments.

References

[1] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Kluwer Academic Publishers, second edition, 2002.

[2] Peter B. Andrews, Matthew Bishop, Sunil Issar, Dan Nesmith, Frank Pfenning, and Hongwei Xi. TPS: A Theorem Proving System for Classical Type Theory. *Journal of Automated Reasoning*, 16:321–353, 1996.

[3] Doug Goldson and Steve Reeves. Using Programs to Teach Logic to Computer Scientists. *Notices of the American Mathematical Society*, 40:143–148, 1993.

[4] Douglas Goldson, Steve Reeves, and Richard Bornat. A Review of Several Programs for the Teaching of Logic. *The Computer Journal*, 36:373–386, 1993.

[5] TPS and ETPS Homepage. http://gtps.math.cmu.edu/tps.html.