The AProS Project: Strategic Thinking & Computational Logic

WILFRIED SIEG, *Carnegie Mellon University, Department of Philosophy, Laboratory for Symbolic & Educational Computing.*
E-mail: sieg@cmu.edu

Abstract

The paper discusses tools for teaching logic used in *Logic & Proofs*, a web-based introduction to modern logic that has been taken by more than 1,300 students since the fall of 2003. The tools include a wide array of interactive learning environments or cognitive mini-tutors; most important among them is the *Carnegie Proof Lab*. The *Proof Lab* is a sophisticated interface for constructing natural deduction proofs and is central, as strategically guided discovery of proofs is the distinctive focus of the course. My discussion makes explicit the broader intellectual context, but also the pursuit of pedagogical goals and their experimental examination. The intellectual context includes i) the theoretical background for the proof search algorithm *AProS* and its use for a dynamic *Proof Tutor*, and ii) the programmatic expansion of the course to *Computational Logic*. (I recommend that the reader enter the virtual classroom of *Logic & Proofs*: the interactive components just cannot be properly reflected in a narrative. It is also very easy to download *AProS* and observe its ways of searching for proofs.)[1]

Keywords: introduction to logic, strategic thinking, automated proof search, dynamic proof tutor, Carnegie Proof Lab, Open Learning Initiative (OLI), intercalation calculus, natural deduction proof.

Introduction

The mathematics needed in the 21st century is more than the calculus that evolved in fruitful symbiosis with physics, beginning in the 17th century. Leibniz, who was one of the inventors of this mathematical calculus, also conceived of a logical calculus at the time. He put great emphasis on logical rules to *guide thinking*, on a universal language to *organize concepts*, and on algorithms to *solve problems*. The idea of the logical calculus came to theoretical fruition only in the first half of the 20th century, but it has become absolutely vital during the second half of that century in the context of the computing revolution. There is wide agreement that the crucial components of this new calculus are proofs, functions, and computations; Thurston [27] and Wing [29] discuss complementary and supportive views. A broad perspective of this new logical calculus has shaped the intellectual direction for the AProS Project: its fundamental components are to be presented in a fully computer-based course called *Computational Logic*.

In collaboration with a number of colleagues and students I have pursued the AProS Project for roughly twenty years.[2] (AProS stands for **A**utomated **Pro**of **S**earch.) The core of the project has been the development of the automated search method *AProS* for finding

[1] The respective URLs are: http://www.cmu.edu/oli/ and http://www.phil.cmu.edu/projects/apros/.

[2] More information about the AProS Project, in particular concerning collaborators and background, can be found at www.phil.cmu.edu/projects/apros/. The project's central directions, restricted to pure logic, were initially pursued as the Carnegie Mellon Proof Tutor Project. — Thanks to all participants and special thanks, on this occasion, to Douglas Perkins who helped me to greatly improve my presentation.

Vol. 15 No. 4, © The Author, 2007. Published by Oxford University Press. All rights reserved.
For Permissions, please email: journals.permissions@oxfordjournals.org
doi:10.1093/jigpal/jzm026

natural deduction proofs in classical, intuitionist, and minimal logic, as well as in some modal logics. The methods were extended to finding proofs in elementary metamathematics (see [26]) and are being adapted now for basic set theory. The motivation was at first purely pedagogical, as it was to serve as the basis for intelligently tutoring students in their efforts to construct arguments. A report was given in [23]. However, some theoretical logical challenges had to be overcome and it took a long time to develop and implement proof search methods that are strategically motivated as well as efficient. It took an even longer time before we began designing an instructional setting for our pedagogical objectives; in the summer of 2003 we started to develop a computer-based introduction to modern logic, *Logic & Proofs*, which is focused on the strategic discovery of proofs. *Logic & Proofs* contains the core of material that will be systematically extended over the next two years to a novel course, *Computational Logic*: the treatment of proofs is to be deeply integrated with that of functions (in 2007) and computations (in 2008).[3]

The development of the new material for *Computational Logic* provides one obvious and important context for our work; but there is a second one, namely, Carnegie Mellon's Open Learning Initiative, OLI. This initiative is supported by a group of faculty with strong pedagogical interests. Its goal is to develop web-based courses in a variety of subjects. OLI courses should be excellent in their respective fields, but they should also be based on pedagogical principles that have been discovered through research in cognitive psychology, in particular through research on computer-based teaching and tutoring; see, for example, [1] and also [21]. A number of cognitive psychologists, Koedinger and van Lehn among them, are members of the OLI group. (OLI and its goals are described at www.cmu.edu/oli/.) Our course addresses not only a real educational need, but serves also as a Learning Laboratory to investigate, for example, i) the effectiveness of using a dynamic tutor for strategic thinking, and ii) the empirical adequacy of the model for strategic thinking that underlies the tutor.[4]

1 (Proofs in) Logic

Logic is involved in the systematic development of any subject, once basic principles (axioms) have been formulated and inferences are used to obtain other statements (theorems) as logical consequences. On a large scale, such a development was achieved in mathematics for the first time through Euclid's *Elements*. In that remarkable work inferential principles were not explicitly given, however; it took more than 2000 years before an adequate logic was formulated in the second half of the 19th century. Building on Hilbert and Bernays's work in the 1920s as represented in [13]—work that aimed at providing the means for a direct formalization of mathematical arguments—Gentzen formulated natural deduction systems in his thesis ''Untersuchungen über das logische Schließen'', translated in [10]. His systems use exclusively elimination and introduction rules for the logical connectives and have the further distinctive feature of allowing assumptions in arguments.

[3] This project together with its empirical evaluation is being supported by National Science Foundation grant 0618806 (Mathematical Science Division, CCLI - Expansion Projects).

[4] In this way our work joins the two purposes of research in mathematics education articulated by Schoenfeld in [22], p. 641, namely, to understand on the one hand the nature of mathematical thinking, teaching and learning, and to use on the other hand such understanding to improve instruction.

1.1 Strategic moves

We use a version of Gentzen's calculus for classical logic; the appropriate versions for intuitionistic and minimal logic as well as for modal logics are easily formulated; see [19]. The rules for sentential connectives and quantifiers are standard; only the formulation of the rules for negation deviates from Gentzen's and Prawitz's.

$$\neg\text{I}: \quad \dfrac{\begin{matrix}[\varphi]\\ \vdots \\ \bot\end{matrix}}{\neg\varphi} \qquad\qquad \neg\text{E}: \quad \dfrac{\begin{matrix}[\neg\varphi]\\ \vdots \\ \bot\end{matrix}}{\varphi}$$

These two rules are supplemented by the $\bot$-introduction rule:

$$\bot\,\text{I}: \quad \dfrac{\psi \quad \neg\psi}{\bot}$$

The rules reflect the meaning of the connectives and capture local tactics for their use, but do not directly yield strategies for argumentation. Focusing on this goal, we distinguish three basic strategic moves: forward (use of elimination rules, except for ¬E, to infer formulae from assumptions or already obtained formulae), backward (use of inverted introduction rules to set up new subgoals), and indirect (use of ¬E); the strategic moves are to be applied in this order. Notice that we do not apply introduction rules in a forward direction.

There remain, however, at least two strategic problems. The first is this: the restricted forward moves can be pointless, unless they are "goal directed", i.e., the (current) goal is a positive subformula of an assumption and can be reached potentially by a sequence of elimination inferences. The form of the statement in which the goal is embedded determines this sequence uniquely, and elimination rules should be applied only in such contexts. We call the sequence an "extraction sequence" and say that we "extract" the goal from the appropriate premise. The second problem concerns the choice of a contradictory pair ψ and $\neg\psi$ when, in an indirect argument, the falsum $\bot$ has to be inferred: attention can be restricted to formulae ψ whose negation $\neg\psi$ occurs as a positive subformula of an assumption.

This strategic approach to proof construction is at the intellectual center of *Logic & Proofs*. Students practice it in the *Carnegie Proof Lab* or *CPL*. In the Proof Lab, Fitch Diagrams are used for the presentation of natural deduction proofs. The Lab is the critical place for actively engaging individual students. It is a sophisticated application for constructing proofs that allows students to make logical moves in the most intuitive way. In particular, they can argue forward and backward; in [21] it was shown that this is crucially important for students to find difficult arguments. The Lab also diagnoses errors in employing rules or manipulating proofs. The error-reports provide links to instructional material related to the mistakes. Students can preview moves before applying them in their argument, and they can also save partial work. The interface is quite intuitive; the reader can get a good impression, by opening, for example, the exercise section of Chapter 7 and seeing how students can work in the Lab. (The URL is in note 1.) Within the very near future we shall take another important step and provide dynamically generated tutorial support: the Lab will be connected to the

AProS proof generator yielding the equation Proof Lab + Proof Generator = Proof Tutor; that is discussed in section 2. All of this is to address, in the context of logic, the core challenge for teaching any formal material well, namely, to make up for the dramatically varied background of students by highly individualized instruction.

1.2 Pedagogical structures

The instruction in *Logic & Proofs* goes beyond teaching symbolic languages and formal calculi, as it helps students gain a reflective approach to proof construction. This reflective approach is supported by the strategic component I just sketched, but also by emphasizing the interpretation of symbolic languages, i.e., by an important semantic component. How are these considerations embedded in the course content? To answer this question, I give an overview of *Logic & Proofs*. However, it is best to enter the virtual classroom for the course and explore it in some detail.

A brief ''Preface'' surveys the course and articulates its main goals. It is followed by an ''Introduction'' that connects arguments with the logical form of statements. Sentential logic is then systematically explored in six chapters. The first chapter, ''Syntax and Symbolization'', discusses the syntax of sentential logic and the use of formulae to represent the logical form of statements. The truth-functional meaning of the connectives is presented in Chapter 2, ''Semantics'', and exploited to motivate the construction of truth-trees (or analytic tableaux). Such a construction is presented as an efficient way of obtaining counterexamples. This semantic approach is contrasted with systematically finding proofs in the next three chapters, ''Derivations'', ''Indirect Rules'', and ''Strategies and Derived Rules''. The sixth and final chapter, ''Elementary Metamathematics'', discusses truth-functional completeness of various fragments of the language, normal forms, and the semantic completeness of the calculus. This basic structure is mirrored in the following six chapters for the treatment of predicate logic (with identity).

The overall design of the course and the detailed presentation of material are to invite active engagement and critical reflection. The organization of the individual chapters serves the very same purpose. Every chapter is introduced by a brief lecture describing its content and formulating its learning objectives. That is followed by the presentation of the core material divided into topically coherent sections. Finally, there is a chapter review in which students look back at the learning objectives and then have ample opportunities to explore problems. Crucially, every major concept or technique is accompanied and supported by engaging ''Interactive Learning Environments'' (ILEs) or ''dynamic cognitive mini-tutors''. If you are already web-connected to *Logic & Proofs*, listen to the brief lecture on formal syntax at the beginning of Chapter 2. Then look at a paradigmatic ILE concerning parse trees that discusses how one decides whether or not a particular string of symbols is a grammatically correct formula of the language.

As mentioned above, the reflective approach to proof construction has two components, the strategic one and a complementary semantic one: students have to gain insights into semantic relations like logical consequence (and its connection to counterexamples). That requires a detailed discussion of the truth definition for sentential and predicate logic. Again, if you are web-connected to *Logic & Proofs*, listen to the brief lecture on ''chasing truth up the tree of grammar'' at the beginning of section 1 in Chapter 3; then look at the ILEs that follow immediately. It is our goal here to go dramatically beyond the current thorough presentation and build a sophisticated *Semantics Lab* that will do for semantic modeling what the *Proof Lab* does for syntactic proof construction. That is a project I am pursuing jointly with a colleague at Stanford, Grigori Mints.[5]

We have offered *Logic & Proofs* at Carnegie Mellon and at IUPUI (Indiana University and Purdue University at Indianapolis), beginning in the fall of 2003. It has been taught successfully to about 300 students each year; in the academic year 2006/07 we had more than 400 students. We solicited feedback from students and used their responses to iteratively improve the course; as to the effect of introducing ILEs see [4]. Among the students who have taken *Logic & Proofs* at Carnegie Mellon is a surprising number of graduate students—roughly 10%—mostly from Engineering departments. Undergraduates taking the course are from many different parts of the University, from Computer Science and Engineering, English and Philosophy, Biology and Physics, and the College of Fine Arts (mostly the School of Drama).

2 Automated Search & Tutoring

The systems of natural deduction are indeed natural for human proof construction. For the longest time they were considered as ''unsuitable for automated theorem proving''; that's how Fitting expressed it in the first edition of his book [9] on theorem proving. The preferred theorem proving methods were, and to a large extent still are, based on either resolution or on tableaux systems, which are notational variants of Gentzen's sequent calculi. To overcome this unsuitability, I introduced intercalation calculi for sentential logic in the late 1980s and for full predicate logic in the early 1990s. (See papers [23]-[25] and the dissertation [5].)

2.1 Exploring questions

Intercalation calculi provide the proof theoretic underpinnings for automated proof search. Indeed, their role in the search for natural deduction proofs corresponds roughly to that of cut-free calculi in the search for sequent derivations. They allow building up suitable search spaces that contain either a normal proof or a counterexample. Using this observation one can establish a *sharpened completeness* theorem: If φ is a logical consequence of Γ, then there is a normal proof of φ from Γ. That allows one to obtain in turn a *normal form theorem* by semantic means: If there is a proof of φ from Γ, then there is a normal proof of φ from Γ.

To obtain these metamathematical results most directly, we use a formulation of intercalation calculi that locates them properly between sequent and natural deduction calculi. In his 1936 paper on the consistency of arithmetic, Gentzen gave a sequent formulation of natural deduction; see [10]. Let me indicate his rules for conjunction as an example. &I: if there is a proof of φ_1 from Γ and a proof of φ_2 from Γ, then there is a proof of φ_1 & φ_1 from Γ; &E: if there is a proof of φ_1 & φ_1 from Γ, then there is a proof of φ_1 from Γ and a proof of φ_2 from Γ. The basic configurations in proofs are sequents of the form $\Gamma \supset \varphi$, where Γ is a finite sequence of formulae, the open assumptions on which φ depends. All the rules, except for the structural ones Gentzen uses, operate essentially on the right-hand side of the horseshoe.

[5] This collaborative work builds on the project ''Define/Check'' Mints pursued in 2002/03.

By contrast, the basic configurations of intercalation calculi are questions of the form $\Gamma\ ;\ \Delta\ ?\ \varphi$. Both Γ and Δ are sequences of formulae, Γ indicating the premises from which φ may be inferred and Δ a sequence of formulae obtained by elimination rules applied to an element of Γ containing φ as a positive subformula. (Structural rules are not used at all.) The crucial point to notice is that elimination rules are only applied on the left-hand side of the question mark, whereas introduction rules are used only on the right-hand side. That corresponds to the tactical moves I discussed above and, in this metamathematical context, avoids detours of steps leading to non-normal proofs.

If we start out with the question $\Gamma\ ?\ \varphi$, i.e., is there a (normal) proof leading from assumptions in Γ to φ, then we can build up a search space using the rules of the intercalation calculus. The proof generator *AProS* is a logically complete procedure for traversing this space: if there is a proof of φ from the given premises Γ, *AProS* finds a normal proof. The generator is guided strategically in the same way as our students, namely to pursue in a refined way the three basic moves: forward, backward and the classical indirect argument via $\neg$E. You can download *AProS* at http://www.phil.cmu.edu/projects/apros/ and follow its search stepwise for proofs of Tertium non datur, Contraposition, de Morgan's Law or any problem of your choice.

2.2 Finding answers

The Proof Tutor will link *AProS* and the Proof Lab. It is intended to help students as follows: if a student requests help in constructing a proof, the Tutor asks *AProS* to complete the argument from (essentially) the point the student left it. The Tutor then analyzes the strategic moves in *AProS's* proof and provides hints, based on its analysis. The first hint will be a general strategic one; subsequent hints will provide more concrete advice on how to proceed. The last hint in the sequence will recommend that the student take a particular step in the proof construction.

As the first step of building the Tutor, we have begun to automatically analyze completed proofs given by *AProS*, i.e., to walk through and describe *AProS* proofs. Since *AProS's* procedure is strategically guided, one can trace its search path and obtain information on which to base a broad description as well as successive refinements. In the second step of building the Proof Tutor, we are considering partial proofs given by a student and provide advice on how to complete the argument. That may require backtracking in the proof search and choosing a different path in the search space. This proof theoretic problem of pruning a given partial proof is being explored now by my student Douglas Perkins as part of his M.S. Thesis ''Strategic Proof Tutoring in Logic''.

Our implementation will be informed by good solutions of the logical problems, but also by interface considerations and metacognitive features of effective computer-based tutors.[6] In particular, the structure of the interaction between Tutor and students will take into account our broad goal of emphasizing the connection between formal and informal reasoning together with the crucial need of strategic reflection. Ultimately, the implementation has to be measured against the effective support the Tutor provides to students. Here detailed empirical investigations come in; we are currently analyzing the effect of ILE supported instruction on students' strategic proof attempts.

[6] Cf. the investigations of Aleven and Koedinger in [1].

3 Functions & Computations

The expansion of the logical material to elementary set theory and to the basic theory of computation has two distinct purposes: the first and primary one is, naturally, to convey fundamental knowledge to students; the second is to exploit the strategic approach to giving proofs in logic for presenting the sophisticated course material in a rigorous way. In the first expansion, proofs are no longer purely logical, but use definitions, theorems and a thorough understanding of core concepts. We will expand *AProS* and the Proof Tutor to provide automated tutoring that emphasizes key heuristics in addition to logical strategies. For the basic theory of computation we are going to proceed differently; that is discussed below.

3.1 Elements of set theory

Here is a brief outline of the topics that are to be discussed in six chapters, the first three of which present material that is canonical for an elementary introduction to set theory; cf. the very thoughtful and polished presentations by Devlin [8] and Velleman [28]. The first chapter begins with an informal characterization of sets and formulates two basic principles, extensionality and separation. Functional relations are discussed, and the standard Boolean operations, but also the powerset operation, are introduced as set generating processes. Finally, natural numbers are set theoretically given and the principle of proof by induction is justified. The second chapter introduces ordered pairs; relations and functions between sets can then be seen as set theoretic objects. We discuss and establish the basic properties of injections, surjections, and bijections. Finally, the principle of (primitive) recursive definition is justified. The third chapter is devoted to diagonalization, the proof of Cantor's Theorem, and a detailed argument for the Cantor Bernstein Theorem.

The center of attention in these chapters is the concept of function, which reflects, as Dedekind put it in his classical essay [7], ''the ability of the mind to relate things to things, to let a thing correspond to a thing, or to represent a thing by a thing, an ability without which no thinking is possible.'' The topics in the remaining chapters are deeply related to this conceptual core, except for Chapter 4, which presents the Zermelo Fraenkel axioms for set theory (ZF) and discusses segments of the cumulative hierarchy as the intended model for ZF. The point of this chapter is for students to gain a thorough understanding of the axiomatic method and see that the informal, but rigorous developments in the three preceding chapters are directly formalizable in ZF. The fifth chapter introduces graphs and explores colorings as well as matchings. In the sixth chapter we transition to the second expansion and discuss, in detail, Gödel's definition of general recursive functions as those number theoretic functions that satisfy recursion equations and whose values can be determined, for each set of arguments, via elementary rules for equational reasoning.

3.2 Basics of computability theory

Having moved from purely logical arguments to proofs in set theory and having shifted from general set theoretic functions to the calculable number theoretic ones, we develop elementary computability theory guided by the classical texts of Davis [6] and Kleene [15]; our perspective is that of [17]. The concept of a ''Turing machine computation'' is at the center of this expansion, and the crucial point is for students to gain an appreciation of the power and limitations of computational methods.[7]

Here is a brief description of the mathematical topics. Chapter 1 discusses the decidability of monadic logic, finite state machines, and the general decision problem. The second chapter covers Turing machine computations and looks at some problems that can be effectively decided by Turing machines but not by finite state machines. In the third chapter the Turing computability of all general recursive functions is shown, as is Kleene's normal form theorem. The next chapter establishes the unsolvability of the halting problem, which is used in Chapter 5 to prove the undecidability of predicate logic. In Chapter 6 Gödel's Incompleteness Theorems [12] will be established for set theory, at an abstract level. The phrase ''at an abstract level'' means that we take for granted as axioms the crucial representability and derivability conditions, as well as the construction of self-referential statements; that was presented in [26].

3.3 Extension of AProS

The core strategies of the generator can be expanded to efficiently prove, at an abstract level, Gödel's Incompleteness Theorems and the Cantor Bernstein Theorem. If the underlying approach can be further generalized and results in adequate procedures, then we have the beginnings of a deep characterization of the cognitive operations involved in finding proofs in mathematics. In any event, for the elementary part of set theory to be developed in *Computational Logic*, it is very plausible that *AProS* will prove all the relevant theorems. At issue is in what form strategic thinking can be extended most effectively from the formal context of pure logic to the semi-formal one of set theory and, then, to the informal context of computability theory. This is crucial for the pedagogical and cognitive goals.

In order to provide tutorial support in set theory, not only the proof search procedure of *AProS* has to be extended, but also the functionality of the Proof Lab. Gibson in [11] has already extended *AProS* so that identities are naturally incorporated into the search procedure. For elementary set theory we have to achieve two goals: first, obtain a fully automated version of the argument in [14] and, second, to establish the intermediate lemmata used in this proof from the axioms for set theory. Appropriate heuristics, complementing the basic strategic search, can be found; this is in the classical tradition of Polya [18], but is also informed by contemporary work on proof planning, e.g., by Bundy [3] and Melis [16]. The crucial task is to isolate the ''leading ideas'' underlying a part of mathematics and to formulate them procedurally. That was done quite successfully for the proof of the Incompleteness Theorems in [26] and will provide the basis for tutorial support on the Gödel material that is presented in the last chapter of Computational Logic.

4 Concluding Remarks

Implementing the course amounts to building a Learning Laboratory for investigating the effectiveness, extendibility, and transfer of strategic thinking. The Proof Lab is the crucial instrument for this research, as it allows us to collect data sufficient to reconstruct, at every stage, the state of a student's proof search: *it allows us to look into the ''proving mind''.*

[7] For this part we are, with the permission of John Etchemendy, going to transform Turing's World into a Computing Lab, in which students can program and run Turing machines.

Not only can we evaluate the effectiveness of dynamic tutoring and other pedagogical matters, but we can also investigate the adequacy of *AProS* as a cognitive model of strategic thinking in logic and beyond, i.e., analyze in the most refined way rational argument construction.

Such analyses are connected with work in cognitive psychology. Rips formulated in [20] a computational theory of mental proofs given by PSYCOP, an automated theorem prover that has structural similarities with *AProS*. In addition, *AProS's* search procedure falls under Anderson's ACT-R theory, as it is production rule based; its operations can be integrated into the cognitive architecture as directly as the restricted algebraic operations Anderson considers in [2]. Above and beyond the logical and empirical support for the claim that *AProS* reflects deep features of human argumentation, there is thus a theoretical integration suggesting psychological naturalness.

AProS and its extensions provide the intellectual basis of our sophisticated Proof Tutor for logic and elementary set theory. The Tutor will assist students to practice the ways of strategic thinking in formal and semi-formal contexts and transfer them to informal, but logically structured ones; it will help us to teach effectively the basic components of *Computational Logic*, the calculus for the 21st century, and transcend the boundaries of standard educational settings.

References

[1] V. Aleven and K. Koedinger: An effective metacognitive strategy: learning by doing and explaining with a computer-based Cognitive Tutor; Cognitive Science 26, 2002, 147–179.

[2] J. Anderson: Human symbol manipulation within an integrated cognitive architecture; Cognitive Science 29 (2005), 313–341.

[3] A. Bundy: Planning and patching proof; in: B. Buchberger and J.A. Campbell (eds.), *Artificial Intelligence and Symbolic Computation*, Springer-Verlag, 2004, 26–37.

[4] M. Burke: Logic and Proofs (Web-based course); Teaching Philosophy 29 (2006), 255–260.

[5] J. Byrnes: *Proof Search and Normal Forms in Natural Deduction;* Ph.D. Dissertation, Carnegie Mellon, 1999.

[6] M. Davis: *Computability and Unsolvability*; New York, 1958.

[7] R. Dedekind: *Was sind und was sollen die Zahlen?* (1888); its translation is found in: W. Ewald (ed.), *From Kant to Hilbert*, Oxford University Press, 1996, 790-833.

[8] K. Devlin: *Sets, functions, and logic — an introduction to abstract mathematics*; Chapman & Hall, 2004.

[9] M. Fitting: *First-order logic and automated theorem proving*; Springer-Verlag, 1990.

[10] G. Gentzen: *The collected papers of Gerhard Gentzen*, edited and translated by M.E. Szabo; North-Holland, 1969.

[11] T. Gibson: *Proof search in first-order logic with equality; MS Thesis*, Carnegie Mellon, 2006.

[12] K. Gödel: On undecidable propositions of formal mathematical systems (1934); in: *Collected Works*, volume I, Oxford University Press, 1986, 346–371.

[13] D. Hilbert: Die Grundlagen der Mathematik (1927); translated in: J. van Heijenoort (ed.), *From Frege to Gödel — A source book in mathematical logic*, 1879–1931; Harvard University Press, 1967, 464–479.

[14] I. Kash: A partially automated proof of the Cantor-Bernstein Theorem; Senior Honors Thesis, Adv. W. Sieg, Carnegie Mellon, Computer Science Department, 2004.
[15] S.C. Kleene: *Introduction to Metamathematics*; North-Holland, 1952.
[16] E. Melis: Why proof planning for Maths education and how?; in: D. Hutter and W. Stephan (eds.), *Mechanizing Mathematical reasoning*, Springer-Verlag, 2005, 364–378.
[17] D. Mundici and W. Sieg: Computability, in: *Routledge Encyclopedia of Philosophy*, 1998.
[18] G. Polya: *Mathematical Discovery — On understanding, learning, and teaching problem solving*; Wiley & Sons, 1981.
[19] D. Prawitz: *Natural Deduction — A proof-theoretic study*; Stockholm, 1965.
[20] L.J. Rips: *The Psychology of Proof*; MIT Press, 1994.
[21] R. Scheines and W. Sieg: Computer environments for proof construction; Interactive Learning Environments 4(2), 1994, 159–169.
[22] A.H. Schoenfeld: Purposes and methods of research in mathematics education; Notices Amer. Math. Soc. 47(6), 2000, 641–649.
[23] W. Sieg and R. Scheines: Searching for proofs (in sentential logic); in: L. Burkholder (ed.), *Philosophy and the Computer*, Westview Press, 1992, 137–159.
[24] W. Sieg and J. Byrnes: Normal natural deduction proofs (in classical logic); Studia Logica 60 (1998), 67–106.
[25] W. Sieg and S. Cittadini: Normal natural deduction proofs (in non-classical logics); in: D. Hutter and W. Stephan (eds.), *Mechanizing mathematical reasoning*, Springer-Verlag 2005, 169–191.
[26] W. Sieg and C. Field: Automated search for Gödel's proofs; Annals of Pure and Applied Logic 133, 2005, 319–338.
[27] W. Thurston: On proof and progress in mathematics; Bull. Amer. Math. Soc. 30 (1994), 161–177.
[28] D.J. Velleman: *How to prove it — a structured approach*; Cambridge University Press, 1994.
[29] J. Wing: Computational thinking; Communications ACM 49 (2006), 33–35.

Received 1 June 2007