A New Approach to Teaching Discrete Mathematics

David Gries[*] and Fred B. Schneider[†]
Computer Science, Cornell University

June 20, 2001

Abstract

We advocate teaching introductory discrete mathematics by first teaching equational propositional and predicate logic and then using it as *tool* (and not simply viewing it as an object of study) throughout the study of later topics —e.g. set theory, induction, relations, theory of integers, combinatorics, solving recurrence relations, and modern algebra.

The in-depth (6–7 weeks) treatment of logic emphasizes rigorous proofs, strategies for developing proofs, and much practice, so that students develop a skill in formal manipulation. Care is taken to explain all concepts clearly and rigorously. Later topics are dealt with by viewing them as theories —extensions of the predicate calculus.

The course should motivate the use of logic as a pervasive tool. It must enlighten, and not stifle and oppress. Our experience shows that this is possible. Anecdotal evidence shows that students come away with less fear of mathematics, proof, and notation, more confidence in their mathematical abilities, an appreciation for rigor, and the urge to try out their new skills in other courses.

Keywords

Teaching mathematics, equational logic, discrete mathematics.

[*]Based on work supported in part by the NSF under grant CDA-9214957 and Darpa under ONR grant N00014-91-J-4123.

[†]Based on work supported in part by the ONR under contract N00014-91-J-1219, the NSF under grant CCR-9003440, DARPA/NSF grant CCR-9014363, NASA/DARPA grant NAG-2-893, and AFOSR grant F49620-94-1-0198. The opinions expressed herein are those of the authors and do not reflect the views of these agencies.

1 Introduction

Generally speaking, mathematicians and computer scientists are not satisfied with the level at which college students understand math. Students have difficulty constructing proofs, their reasoning abilities are inadequate, and they don't know what constitutes "rigor". Moreover, their fear of formalism and rigor impedes learning in science and engineering courses, as well as in math courses.

This paper describes a new approach to teaching introductory discrete math at the freshman/sophomore level. Our experience with the approach leads us to believe that it engenders a fresh and positive outlook on mathematics and proof, addressing the problems mentioned above.

A key ingredient of our approach is an equational treatment of propositional and predicate logic —instead of the prevailing natural-deduction and Hilbert styles. The equational treatment has several advantages.

1. It is easily learned, since it builds on students' familiarity with equational reasoning from high-school algebra.
2. After just three days of teaching logic, problems can be solved that would be difficult without the logic, thus providing motivation for further study and for developing a skill.
3. The logic can be applied rigorously, without complexity and detail overwhelming (a criticism of other logics), so it becomes a useful tool.
4. The accompanying rigid proof format allows the introduction and discussion of proof principles and strategies, thereby giving students direction in developing proofs.
5. The logic can be extended to other areas, including those typically taught in discrete math courses.
6. A problem-solving paradigm that students see in high school is reinforced: (i) formalize, then (ii) manipulate the formalization according to rigorous rules, and finally (iii) interpret the results.

Our course in discrete math starts with six weeks of propositional and predicate logic. The emphasis is on giving students a skill in formal manipulation. Students see many rigorous proofs and develop many themselves, in a rigid proof style (not in English). They practice applying proof principles and strategies made possible by the proof style. Along the way, they learn that attention to rigor can be a simplifying force —instead of an onerous burden.

The unit on logic does include a discussion of informal presentations of proofs typically found in math (e.g. proof by contradiction) and how they are based in logic (e.g. proof by contradiction is based on the theorem $p \equiv \neg p \Rightarrow \mathit{false}$). Thus, students see the connection between the informal proof sketches they see in other courses and rigorous proofs.

The rest of our course covers a variety of other topics of discrete math, e.g. set theory, a theory of integers, induction, relations and functions, combinatorics, and solving recurrence relations. Each topic is presented in the same rigorous style in which logic was presented, by giving the necessary axioms for the theory and building up a library of theorems. For example, set theory is introduced by adding to pure predicate logic the axioms that characterize set comprehension and the set operators.[1] Proofs about these set operators are then presented in the same style as before. Thus, the notions of proof and proof style become the unifying force in the course.

Many believe that stressing rigor turns students away from math. Our experience is just the opposite. Students take comfort in the rigor, because it gives them a basis for knowing when a proof is correct. And, we find our students applying what they have learned to other courses —even before our discrete math course ends. However, success in teaching logic and rigor does require that students have time to digest the new notations and to become skilled in formula manipulation.

The paper proceeds as follows. In Section 2 we compare our equational proofs with more conventional approaches. We then turn (in Section 3) to our equational propositional logic and discuss how we teach it. We follow this up with a discussion of quantification and predicate logic. In Section 5, we show how to use our logic as the basis for other topics in discrete math. Our approach to logic meets resistance from those who feel that meaning and understanding is sacrificed when formal manipulation is emphasized. We address this criticsm in Section 6. We conclude in Section 7 with some comments about experiences in using the new approach.

2 A comparison of proof methods

Consider proving, from set theory, that union distributes over intersection:

(1) $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$.

[1]Of course, informal English descriptions, Venn diagrams, examples, applications, and the like are used in discussing set theory. We don't mean to imply that we simply write formulas on the board without any attempt at discussing and motivating them. But the formal definitions assume far more importance in our approach than usual.

Most mathematicians favor the following kind of proof —at least their texts contain such proofs.[2]

Proof. We first show that $A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C)$. If $x \in A \cup (B \cap C)$, then either $x \in A$ or $x \in B \cap C$. If $x \in A$, then certainly $x \in A \cup B$ and $x \in A \cup C$, so $x \in (A \cup B) \cap (A \cup C)$. On the other hand, if $x \in B \cap C$, then $x \in B$ and $x \in C$, so $x \in A \cup B$ and $x \in A \cup C$, so $x \in (A \cup B) \cap (A \cup C)$. Hence, $A \cup (B \cap C) \subseteq (A \cup B) \cap (A \cup C)$.

Conversely, if $y \in (A \cup B) \cap (A \cup C)$, then $y \in A \cup B$ and $y \in A \cup C$. We consider two cases: $y \in A$ and $y \notin A$. If $y \in A$, then $y \in A \cup (B \cap C)$, and this part is done. If $y \notin A$, then, since $y \in A \cup B$ we must have $y \in B$. Similarly, since $y \in A \cup C$ and $y \notin A$, we have $y \in C$. Thus, $y \in B \cap C$, and this implies $y \in A \cup (B \cap C)$. Hence $(A \cup B) \cap (A \cup C) \subseteq A \cup (B \cap C)$. theorem follows.

This proof has several shortcomings:

- It is, unnecessarily, a ping-pong argument —equality is shown by proving two set inclusions.
- The two case analyses complicate the argument.
- The use of English obscures the structure of the proof.
- Some expressions, for example $x \in A$ and $x \in A \cup B$, are repeated several times, thus lengthening the proof unnecessarily. (Replacing some of the expressions with pronouns would only make matters worse by introducing ambiguity)
- The justifications for inferences within the proof are not given; too much is left for the reader to fill in. For example, the proof says, "If $x \notin A$, then, since $x \in A \cup B$ we must have $x \in B$", but no reference is given to the theorem being used to substantiate this inference.
- The proof style is too undisciplined to invite useful discussion of proof strategies and principles. After seeing one proof like this, students still have difficulty constructing similar proofs.

Some mathematicians and computer scientists favor teaching Gerhard Gentzen's natural-deduction system [3] or something similar. They argue (correctly) that Gentzen developed natural deduction in order to formalize how mathematicians argue in English. However, formalizing contorted and

[2]Lay [7, p. 36] gives this proof, with a few formulas replaced by blanks for the reader to fill in.

difficult English arguments is of dubious utility —the contortions and difficulties remain. We favor proof methods that allow us to bypass problems introduced by the use of imprecise and unwieldy natural language.

Note that we do not advocate altogether avoiding English, informality, ping-pong arguments, or case analysis. They should be used when they lead to proofs that are more accessible to the reader. Too often, however, they are used to ease the task of the writer at the expense of the reader.

Having criticized the traditional proof style, we now illustrate a better one, by giving our own proof that union distributes over intersection. Our treatment of set theory extends an equational logic. For this discussion, then, we assume that theorems of propositional logic and pure predicate logic, like the following one, have already been proved.

(2) $p \vee (q \wedge r) \;\equiv\; (p \vee q) \wedge (p \vee r)$

Equality of sets is defined using the axiom of Extensionality (we describe our notation for quantification in Section 4).

(3) **Extensionality:** $S = T \;\equiv\; (\forall v \mid : v{\in}S \;\equiv\; v{\in}T)$

Further, union and intersection of sets are defined by membership tests.

(4) $v{\in}B \cup C \;\equiv\; v{\in}B \vee v{\in}C$

(5) $v{\in}B \cap C \;\equiv\; v{\in}B \wedge v{\in}C$

Note that set membership, set equality, and other set operators are defined axiomatically, and in a way that is useful in formal manipulation —not, as is traditional, by an informal model of evaluation.

We now prove theorem (1): union distributes over intersection.

Proof. By Extensionality (3), we can prove (1) by showing that an arbitrary element v is in the LHS of (1) exactly when it is in the RHS:

$$\begin{array}{ll}
 & v \in A \cup (B \cap C) \\
= & \quad \langle \text{Definition of } \cup \text{ (4)} \rangle \\
 & v \in A \;\vee\; v \in B \cap C \\
= & \quad \langle \text{Definition of } \cap \text{ (5)} \rangle \\
 & v \in A \;\vee\; (v \in B \wedge v \in C) \\
= & \quad \langle \text{Distributivity of } \vee \text{ over } \wedge \text{ (2)} \rangle \\
 & (v \in A \vee v \in B) \;\wedge\; (v \in A \vee v \in C) \\
= & \quad \langle \text{Definition of } \cup \text{ (4), twice} \rangle \\
 & (v \in A \cup B) \;\wedge\; (v \in A \cup C)
\end{array}$$

$$\begin{array}{ll} = & \langle \text{Definition of } \cap \ (5) \rangle \\ & v \in (A \cup B) \cap (A \cup C) \end{array}$$

This proof uses substitution of equals for equals (or "Leibniz", as we call it) as its main inference rule, as well as transitivity of equality to link steps. College freshmen are familiar with these rules from high-school algebra and therefore become comfortable with the proof style rather quickly.

The proof has the following nice properties:

- All arguments are explicit and rigorous: With each step, a hint between the two equal terms gives the theorem that indicates why the two terms are equal.
- The proof is simple and direct. Its simplicity accentuates, rather than hides, the relation between disjunction and union and between conjunction and intersection.
- The strategy used in the proof is easy to identify and teach: To prove something about operators (here, $\cup$ and $\cap$), use their definitions to eliminate them, perform some manipulations, and then use their definitions to reintroduce them.

Now consider a theorem whose proof uses quantification (even though the statement of the theorem does not). At the recent AMS meeting in Cincinnati, a mathematician demonstrated a computer program for helping in the development of proofs. The program's interface was elegant and easy to use, but the underlying proof system was not. The logic was natural deduction, augmented with definitions from set theory. The mathematician demonstrated his system with a proof of

(6) $A \subseteq C \land B \subseteq C \;\Rightarrow\; A \cup B \subseteq C$.

Because the definition of $\subseteq$ is in terms of universal quantification,

(7) $B \subseteq C \;\equiv\; (\forall x \mid x \in B : x \in C)$,

his formal proof required several levels of nesting (there were proofs within proofs within proofs), as well as case analysis. The proof was not easy to follow or develop. Below is our simpler, equational proof. It happens to use the same proof strategy as the previous equational proof: eliminate an operator, manipulate, and reintroduce the operator.

$$
\begin{array}{ll}
 & A \subseteq C \;\wedge\; B \subseteq C \\
= & \quad \langle \text{Definition of } \subseteq \text{ (7), twice} \rangle \\
 & (\forall x \mid x \in A : x \in C) \;\wedge\; (\forall x \mid x \in B : x \in C) \\
= & \quad \langle \text{Split range} \rangle \\
 & (\forall x \mid x \in A \vee x \in B : x \in C) \\
= & \quad \langle \text{Definition of } \cup \text{ (4)} \rangle \\
 & (\forall x \mid x \in A \cup B : x \in C) \\
= & \quad \langle \text{Definition of } \subseteq \text{ (7)} \rangle \\
 & A \cup B \subseteq C
\end{array}
$$

An equivalence has been proved, instead of implication (6)! Extending the speaker's proof to justify this equivalence would double the length of his proof, because natural deduction is being used.

In the two examples presented, the equational approach is far superior to more conventional approaches —to the informal model-theoretic proof in the first example and to the logic-based proof alluded to in the second example. In our equational proofs, there is much less writing, the proofs are absolutely rigorous yet simple, and the proofs are easy to digest and repeat to others. All one has to know is an equational version of the pure predicate calculus and the few definitions of set theory.

Were these isolated instances of the superiority of the equational approach, there would little reason to discuss it. However, in our experience, the equational approach exhibits these advantages in all topics that are traditionally included in a first-semester discrete math course. This should not be surprising, since logic is the glue that binds together arguments in all domains. Moreover, bringing this glue to the fore provides the unifying theme that has been missing for so long in discrete math courses.

3 Equational propositional logic

We now outline our equational propositional logic and discuss teaching it. The logic has the same theorems as conventional propositional logic; the difference is in the inference rules, the axioms (and the order in which they are introduced), and the format of proofs.

Equivalence (i.e. equality over the boolean domain) plays a prominent role in our logic; in comparison, it is a second-class citizen in most other propositional logics, where implication dominates. And, we profit from using two different symbols for equality, $=$ and $\equiv$. The expression $b = c$ is defined as long as b and c have the same type —e.g. both booleans, both integers, both set of integers, both graphs. Further, equality $=$ is treated conjunctionally: $b = c = d$ is an abbreviation for $b = c \wedge c = d$.

Symbol $\equiv$ is used only for equality over booleans: $b \equiv c$ is the same as $b = c$ for b, c boolean. Symbol $\equiv$ is assigned a lower precedence than $=$; this allows us to eliminate some parentheses. For example, we can write

$$x = y \;\vee\; x < y \quad\equiv\quad x \leq y \quad .$$

(Note how spacing is used to help the reader with precedences.) Formal manipulation will be used often, and we need ways to keep formulas simple.[3]

A more important benefit arises from having two symbols for equality: we can make use of associativity of equality over the booleans. We write $b \;\equiv\; c \;\equiv\; d$ for either $b \;\equiv\; (c \;\equiv\; d)$ or $(b \;\equiv\; c) \;\equiv\; d$, since they are equivalent. Had we used only $=$ for equality, we could not have benefited from associativity because $a = b = c$ already means $a = b \wedge b = c$. As with arithmetic manipulations, we often use symmetry (commutativity) and associativity of operators without explicit mention. Logicians have not made use of the associativity of $\equiv$. For example, in [8], Rosser uses $\equiv$ conjunctionally instead of associatively. Perhaps this is because logicians have been more interested in *studying* rather than *using* logic.

The four inference rules of our equational logic are given below, using the notation $E[v := P]$ to denote textual substitution of expression P for free occurrences of variable v in expression E:

Leibniz: $\dfrac{P = Q}{E[v := P] = E[v := Q]}$

Transitivity: $\dfrac{P = Q,\; Q = R}{P = R}$

Substitution: $\dfrac{P}{P[v := Q]}$

Equanimity: $\dfrac{P,\; P \;\equiv\; Q}{Q}$

A *theorem* of our logic is either (i) an axiom or (ii) the conclusion of (an instance of) an inference rule whose premises are theorems.

Our proofs generally follow the format of the equational proofs given earlier (although there are some extensions). With this format, uses of inference rules can be left implicit —the rules need not be mentioned. Contrast

[3]As another simplification, we write the application of a function f to a simple argument b as $f.b$.

this with natural deduction and Hilbert-style logics, where the plethora of inference rules dictates the explicit mention of each use of a rule.

Inference rule Leibniz is used to infer an equality; its use is indicated in proofs as:

$$\begin{array}{ll} & \ldots \\ & E[v := P] \\ = & \quad \langle\, P = Q \,\rangle \\ & E[v := Q] \\ & \ldots \end{array}$$

Inference rule Transitivity is used without mention in the usual fashion; for example, it is transitivity that allows us to conclude $v \in A \cup (B \cap C) \equiv v \in (A \cup B) \cap (A \cup C)$ in the proof on page 5. And, inference rule Substitution is used to generate an instance of a theorem that is to be the premise of an application of Leibniz. For example, in the step

$$\begin{array}{ll} & p \wedge q \wedge r \\ = & \quad \langle\, p \wedge q \;\equiv\; q \wedge p \,, \text{with } q := q \wedge r \,\rangle \\ & q \wedge r \wedge p \end{array}$$

the premise of the instance of Leibniz that is being used is $p \wedge q \;\equiv\; q \wedge p$ with $q := q \wedge r$, i.e. $p \wedge (q \wedge r) \;\equiv\; (q \wedge r) \wedge p$. Often in our proofs, Substitution is not mentioned when its use is obvious.

Finally, if the last expression in an equational proof is a theorem, then, by inference rule Equanimity, the first expression is also a theorem. A use of this rule is indicated by the last expression being the theorem *true* or by a comment to the right of the expression, indicating which theorem it is.

The axioms of our equational logic are given in Table 1, ordered and grouped as we teach them. Equivalence is introduced first. And, because the first axiom says that equivalence is associative, thereafter we eliminate parentheses from sequences of equivalences. For example, Symmetry of $\equiv$ (9) is given as $p \equiv q \equiv q \equiv p$. Associativity of $\equiv$ allows us to parse (9) in five ways, thus reducing the number of axioms and theorems that have to be listed: $((p \equiv q) \equiv q) \equiv p$, $(p \equiv q) \equiv (q \equiv p)$, $(p \equiv (q \equiv q)) \equiv p$, $p \equiv ((q \equiv q) \equiv p)$, and $p \equiv (q \equiv (q \equiv p))$.

Our definition of conjunction is called the Golden rule (19). To see that (19) is valid, check its truth table or else use associativity and symmetry to rewrite it as

$$p \;\equiv\; q \quad\equiv\quad p \wedge q \;\equiv\; p \vee q \quad .$$

Table 1: Axioms of Equational Logic

(8) **Associativity of $\equiv$:** $((p \equiv q) \equiv r) \ \equiv \ (p \equiv (q \equiv r))$

(9) **Symmetry of $\equiv$:** $p \equiv q \equiv q \equiv p$

(10) **Identity of $\equiv$:** $\mathit{true} \equiv q \equiv q$

(11) **Definition of *false*:** $\mathit{false} \equiv \neg \mathit{true}$

(12) **Distributivity of $\neg$ over $\equiv$:** $\neg(p \equiv q) \ \equiv \ \neg p \equiv q$

(13) **Definition of $\not\equiv$:** $(p \not\equiv q) \ \equiv \ \neg(p \equiv q)$

(14) **Symmetry of $\vee$:** $p \vee q \ \equiv \ q \vee p$

(15) **Associativity of $\vee$:** $(p \vee q) \vee r \ \equiv \ p \vee (q \vee r)$

(16) **Idempotency of $\vee$:** $p \vee p \ \equiv \ p$

(17) **Distributivity of $\vee$ over $\equiv$:** $p \vee (q \ \equiv \ r) \ \equiv \ p \vee q \ \equiv \ p \vee r$

(18) **Excluded Middle:** $p \vee \neg p$

(19) **Golden rule:** $p \wedge q \ \equiv \ p \ \equiv \ q \ \equiv \ p \vee q$

(20) **Definition of Implication:** $p \Rightarrow q \ \equiv \ p \vee q \ \equiv \ q$

(21) **Consequence:** $p \Leftarrow q \ \equiv \ q \Rightarrow p$

Now, it may be recognized as the law that says that two booleans are equal iff their conjunction and disjunction are equal.

Operator $\Leftarrow$ (see (21)) is included because some proofs are more readily developed or understood using it rather than $\Rightarrow$. As an example, we prove that any integer divides itself, i.e. $c \mid c$ holds, where $\mid$ is defined by

$$b \mid c \ \equiv \ (\exists d \,|\!: b \cdot d = c) \quad .$$

Note that the proof format below has been extended to allow $\Rightarrow$ or $\Leftarrow$ to appear in place of $=$ and that part of the proof is given in English.

Proof. We calculate:

$$\begin{array}{ll} & c \mid c \\ = & \quad \langle \text{Definition of } \mid \rangle \\ & (\exists d \,|\!: c \cdot d = c) \end{array}$$

$$
\begin{array}{ll}
\Leftarrow & \quad \langle\, \exists\text{-Introduction} — P[x := E] \;\Rightarrow\; (\exists x \,|\!: P) \,\rangle \\
 & c \cdot 1 = c \\
= & \quad \langle \text{Multiplicative identity} \rangle \\
 & \mathit{true}
\end{array}
$$

Hence, $\mathit{true} \;\Rightarrow\; c\,|\,c$. By Left Identity of $\Rightarrow$ (i.e. $P \;\equiv\; \mathit{true} \Rightarrow P$), $c\,|\,c$ is a theorem.

Were $\Leftarrow$ not available, this proof would have to be presented in reverse, as shown below. But this second proof is difficult to follow. The first few formulas are rabbits pulled out of a hat. Why start with true? Where did the insight come from to replace true by $c \cdot 1 = c$? A goal in writing a proof should be to avoid rabbits, to have each step guided or even forced by the structure of the formulas, so that an experienced reader would say, “Oh, I would have done that too.” A proof strategy that helps in this regard is: Start with the more complicated side of an equivalence or implication and use its structure to help guide the proof.

$$
\begin{array}{ll}
 & \mathit{true} \\
= & \quad \langle \text{Multiplicative identity} \rangle \\
 & c \cdot 1 = c \\
\Rightarrow & \quad \langle\, \exists\text{-Introduction} \rangle \\
 & (\exists d \,|\!: c \cdot d = c) \\
= & \quad \langle \text{Definition of } \,|\, \rangle \\
 & c\,|\,c
\end{array}
$$

Teaching preliminary concepts (e.g. textual substitution) and the propositional calculus takes us at least eleven 50-minute lectures. Students are shown various proof principles and strategies, are given many proofs, and are asked to prove many theorems themselves. The emphasis is on achieving familiarity with the notation and theorems and on gaining a skill in formal manipulation.

Here is a list of proof principles and strategies that can be taught with our approach. They are not obvious to mathematical novices, and discussing them can be enlightening.

- **Principle.** Structure proofs to avoid repeating the same subexpression on many lines.
- **Principle.** Structure proofs to minimize the number of “rabbits pulled out of a hat” —make each step obvious, based on the structure of the expression and the goal of the manipulation.

Table 2: Portia's Suitor's Dilemma

This story is a take-off on a scene in Shakespeare's *Merchant of Venice.* Portia has a suitor, who wants to marry her. She does not know whether she wants to marry him. So, she gives him a puzzle to solve. She gets two boxes, one gold and the other silver, and puts her portrait into one of them. On the gold box she writes the inscription "The portrait is not here." On the silver box, she writes, "Exactly one of these two inscriptions is true." She tells the suitor that the inscriptions may be true or false, she won't say which, but if he can determine which box contains the portrait, she will marry him.

- **Principle.** Lemmas may provide structure, bring to light interesting facts, and ultimately shorten a proof.
- **Strategy.** To prove something about an operator, eliminate it, manipulate the result, and then, if necessary, reintroduce the operator.
- **Strategy.** The shape of an expression can focus the choice of theorems to be used in manipulating it. Thus, identify applicable theorems by matching the structure of the expression and its subexpressions with the theorems.
- **Strategy.** To prove $P \equiv Q$, transform the term with the most structure (either P or Q) into the other. The idea here is that the one with the most structure provides the most insight into the next Leibniz transformation.

Almost 80 theorems of the propositional calculus can be used as examples and exercises in illustrating proof principles and strategies. Students are not asked to memorize all 80-odd theorems —since mathematicians don't. Rather, students always have on hand a list of the theorems in the order in which they can be proved. Through continual use, students begin to know many of these theorems as their formal friends. On a test, the list of theorems is distributed and students are asked to prove one or two.

We intersperse the study of propositional calculus with interesting applications. Many seemingly difficult word problems can be solved easily through formalization and manipulation, so here is where students begin to see that they are learning a new, powerful, mental tool. One interesting example is Portia's Suitor's Dilemma (see Table 2). Its solution is amazingly simple using our equational logic when one formalizes, manipulates, and then interprets. Its solution in natural deduction is awkward and much longer (see [9, 6] for a discussion).

Table 3: Informal Proof Techniques

Informal proof technique	Basis for the informal technique
Case analysis	$(p \vee q) \wedge (p \Rightarrow r) \wedge (q \Rightarrow r) \;\Rightarrow\; r$
Mutual implication	$p = q \;\equiv\; (p \Rightarrow q) \wedge (q \Rightarrow p)$
Contradiction	$\neg p \Rightarrow \mathit{false} \;\equiv\; p$
Contrapositive	$p \Rightarrow q \;\equiv\; \neg q \Rightarrow \neg p$

Another interesting problem is to make sense of the following sentence: "For every value of array section $b[1..9]$, if that value is in array section $b[21..25]$, then it is not in $b[21..25]$." This sentence may seem contradictory, but formalizing it, simplifying the formal statement, and then interpreting the result yields a surprising answer.

Our study of equational logic is completed with a look at how conventional, seemingly informal, proof techniques in mathematics are based on theorems of propositional calculus (see Table 3). Several proofs are given in both an informal and the equational style and compared.

4 Quantification and the predicate calculus

The treatment of quantification in our course unifies what, until now, has been a rather chaotic topic. Quantification in mathematics assumes many forms, for example:

$$\begin{array}{rcl} \Sigma_{i=1}^{3} i^2 & = & 1^2 + 2^2 + 3^2 \\ (\forall x). 1 \le x \le 3 \Rightarrow b[x] = 0 & \equiv & b[1] = 0 \wedge b[2] = 0 \wedge b[3] = 0 \\ (\exists x). 1 \le x \le 3 \wedge b[x] = 0 & \equiv & b[1] = 0 \vee b[2] = 0 \vee b[3] = 0 \end{array}$$

There appears to be no consistency of concept or notation here. Compounding the problem is that students are not taught rules for manipulating specific quantifiers —much less general rules that hold for all.

We use a single notation for all quantifications. Let $\star$ be any binary, associative, and symmetric operator that has an identity.[4] The notation

(22) $(\star i{:}T \mid R.i : P.i)$

[4] If $\star$ is associative and symmetric but has no identity, then instances of the axioms of Table 4 that have a *false* range do not hold.

Table 4: Axioms for Quantification

(23) **Empty range:** $(\star x \mid false : P) \;=\; (\text{the identity of } \star)$

(24) **One-point rule:** $(\star x \mid x = E : P) \;=\; P[x := E]$

(25) **Distributivity:** $(\star x \mid R : P) \star (\star x \mid R : Q) \;=\; (\star x \mid R : P \star Q)$

(26) **Range split:** Provided $\neg(R \wedge S)$ holds or $\star$ is idempotent,
$(\star x \mid R \vee S : P) \;=\; (\star x \mid R : P) \star (\star x \mid S : P)$

(27) **Range split:** $(\star x \mid R \vee S : P) \star (\star x \mid R \wedge S : P) \;=$
$(\star x \mid R : P) \star (\star x \mid S : P)$

(28) **Interchange of dummies:** $(\star x \mid R : (\star y \mid Q : P)) =$
$(\star y \mid Q : (\star x \mid R : P))$

(29) **Nesting:** $(\star x, y \mid R \wedge Q : P) = (\star x \mid R : (\star y \mid Q : P))$

(30) **Dummy renaming:** $(\star x \mid R : P) = (\star y \mid R[x := y] : P[x := y])$

Note: The usual caveats concerning the absence of free occurrences of dummies in some expressions are needed to avoid capture of variables. Further, the axioms require that quantifications be convergent —e.g. $(\Sigma i \mid 0 \le i : i)$ and $(\equiv i \mid 0 \le i : false)$ are not. All quantifications with finite ranges and quantifications using $\wedge$ and $\vee$ are convergent.

denotes the accumulation of values $P.i$, using operator $\star$, over all values i for which range predicate $R.i$ holds. T is the type of dummy i; it is often omitted in contexts in which the type is obvious.[5] If range $R.i$ is *true*, we may write the quantification as $(\star i \mid: P.i)$.

In the examples below, the type is omitted; also, **gcd** is the greatest common divisor operator.

$$\begin{array}{rcl}
(+i \mid 1 \le i \le 3 : i^2) & = & 1^2 + 2^2 + 3^2 \\
(\wedge x \mid 3 \le x < 7 \wedge prime.x : b[x] = 0) & \equiv & b[3] = 0 \wedge b[5] = 0 \\
(\,\mathbf{gcd}\; i \mid 2 \le i \le 4 : i^2) & = & 2^2 \;\mathbf{gcd}\; 3^2 \;\mathbf{gcd}\; 4^2
\end{array}$$

With the single notation (22) for quantification, we can discuss bound variables, scope of variables, free variables, and textual substitution for all quantifications, once and for all. Further, we can give axioms that hold for all such quantifications (see Table 4).

Having discussed quantification in detail, we then turn to pure predicate logic itself. This calls for just a few more axioms that deal specifically with

[5]Our logic deals with types, but space limitations preclude a discussion of types here.

Table 5: Additional Axioms for Predicate Calculus

(31) **Trading:** $(\forall x \mid R : P) \equiv (\forall x \mid : R \Rightarrow P)$

(32) **Distributivity of $\vee$ over $\forall$:**

(33) $P \vee (\forall x \mid R : Q) \equiv (\forall x \mid R : P \vee Q)$

(34) **(Generalized) De Morgan:** $(\exists x \mid R : P) \equiv \neg(\forall x \mid R : \neg P)$

universal and existential quantification —see Table 5. Note that we do bow to convention and use $\forall$ and $\exists$ instead of $\wedge$ and $\vee$.

Range $R.i$ in notation (22) for quantification can be any predicate. For universal and existential quantification, however, such a range is not necessary. Nevertheless, consistency of notation encourages us to use a single notation, even for universal and existential quantification. Furthermore, in many manipulations, range $R.i$ is left unchanged while term $P.i$ is modified, and the separation of the range and term makes this easier to see. Here, the desire for ease of manipulation has dictated the choice of notations.

Issues of scope, bound variables, etc., make quantification and predicate calculus far more complicated than propositional calculus. Some may even feel that quantification is too complicated for freshmen and sophomores. However, many courses in math, computer science, physics, and engineering require quantification in one form or another, so not teaching quantification means that students are unprepared for those classes. In fact, the lack of knowledge of basic tools like quantification may explain partly why students are apprehensive about mathematics.

Thus, we advocate teaching quantification carefully, completely, and rigorously, but in a manner that instills confidence. We have found that this can be done.

5 Examples of the use of predicate calculus

In our course, equational logic serves as a springboard to the study of other topics. After all, many topics in discrete math can be viewed as extensions of pure predicate calculus —new types of values are introduced and operations on them are defined by adding axioms to the logic. We provide some brief glimpses of how this works.

Set theory

In this article, we use conventional notation $\{x \mid P.x\}$ for set comprehension, but we restrict x to be a bound (dummy) variable (i.e. not an expression). We define membership in such a set comprehension as follows.

(35) **Set membership:** Provided x does not occur free in E,

$$E \in \{x \mid P\} \;\equiv\; (\exists x \mid P : x = E) \;.$$

Recall from (3) that set equality is defined by $S = T \;\equiv\; (\forall x \mid : x \in S \;\equiv\; x \in T)$.

Section 2 already compared equational proofs in set theory to informal English, so we won't do that again here. Instead, we discuss Russell's paradox, which is usually covered in elementary set theory. We define Russell's paradoxical set S (if it is allowed) by its membership test —recall that set union and intersection were also defined by membership tests.

(36) $x \in S \;\equiv\; x \notin x$ for all sets x.

S is a set, and instantiating x with S in (36) yields

$$S \in S \;\equiv\; S \notin S$$

which is *false* . An inconsistency arose by introducing set S . We conclude that S is not well defined and refuse to allow (36). That's all there is to it! There is no need for the confusing, English, ping-pong argument that one finds in many discrete-math texts.

Theory of integers

The integers can be explored by extending predicate logic with the new type $\mathbb{Z}$, giving the axioms for operations on its elements, and then proving various theorems. The same idea was used in introducing sets, so the students see a pattern emerging. Once the pattern is seen, one need not dwell on the proofs of all the theorems concerning addition, multiplication, etc., since the students are already familiar with most of the theorems.

One can spend time on new integer functions and operations, for example the greatest common divisor. We use infix operator **gcd** for this and define it as follows. (We write the maximum of two integers b and c as $b \uparrow c$ and assume that theorems for $\uparrow$ and *abs* have already been taught.)

(37) $b \;\mathbf{gcd}\; c = (\uparrow d \mid d \,|\, b \wedge d \,|\, c : d)$ (for b, c not both 0)

$0 \;\mathbf{gcd}\; 0 = 0$

The first line of (37) does not define $0 \textbf{ gcd } 0$; since all integers divide 0, 0 has no maximum divisor. We define $0 \textbf{ gcd } 0$ to be 0, so that **gcd** is total over $\mathbb{Z} \times \mathbb{Z}$.

Note that $\uparrow$ is associative and symmetric, so we can use it in quantifications. Having spent a great deal of time on quantification, little has to be said about using $\uparrow$ in this manner. However, since $\uparrow$ over the integers does not have an identity, we must avoid using quantification theorems that rely on an identity.

The next step in exploring **gcd** is to state (and prove some of) its properties. This step has two goals: to familiarize the student with the new operation and to provide a basis for later manipulations using **gcd**. Function **gcd** is symmetric, is associative, satisfies $b \textbf{ gcd } b = abs.b$, has 1 as its zero (i.e. $1 \textbf{ gcd } b = 1$), has 0 as its identity (i.e. $0 \textbf{ gcd } b = b$), and so on.

Let us prove $b \textbf{ gcd } b = abs.b$. A few points about the proof given below are worth mentioning. It contains a case analysis, because the definition of **gcd** does. We try to avoid case analysis, but it is not always possible. Second, the proof is written partly in English, since that is the easiest way to see the case analysis. Thus, we are not completely rigid in our proof style. However, the proof does contain two equational subproofs.

Proof. There are two cases to consider: $b = 0$ and $b \neq 0$.

Case $b = 0$.

$$\begin{array}{ll}
 & 0 \textbf{ gcd } 0 = abs.0 \\
= & \quad \langle \text{Def. of } \textbf{gcd} \text{ (37); Def. of } abs \rangle \\
 & 0 = 0 \text{ —Reflexivity of } =
\end{array}$$

Case $b \neq 0$.

$$\begin{array}{ll}
 & b \textbf{ gcd } b \\
= & \quad \langle \text{Def. of } \textbf{gcd} \text{ (37) —} b \neq 0 \text{ by case assumption} \rangle \\
 & (\uparrow d \mid d \,|\, b \wedge d \,|\, b : d) \\
= & \quad \langle \text{Idempotency of } \wedge,\ p \wedge p \equiv p \rangle \\
 & (\uparrow d \mid d \,|\, b : d) \\
= & \quad \langle \text{The maximum divisor of } b \text{ is } abs.b \\
 & \quad\quad \text{—} b \neq 0 \text{ by case assumption} \rangle \\
 & abs.b
\end{array}$$

Mathematical induction

Strong induction over the integers can be defined by the following axiom, where the dummies are of type *natural number*.

Mathematical Induction:

$$(\forall n \mid: P.n) \;\equiv\; P.0 \wedge (\forall n \mid: (\forall i \mid 0 \le i \le n : P.i) \;\Rightarrow\; P(n+1))$$

Our earlier study of quantification has given students the manipulative skills they need to prove formulas by induction. For example, proving $n^2 = (\Sigma i \mid 1 \le i \le n : 2 \cdot i - 1)$ by induction requires knowledge of axioms and theorems for manipulating summations.

The rigorous formulation of induction and proofs by induction also helps clarify certain points that are usually confusing. Here is an example. When proving statements by induction, we always put them in the form

$$(\forall n \mid: 0 \le n : P.n) \quad \text{where} \quad P.n : \;\; \ldots \quad .$$

Thus, we formalize the theorem to be proved and name the induction hypothesis. For example, suppose we want to prove $b^{m+n} = b^m \cdot b^n$ for n, m natural numbers. Writing this formula as

$$(\forall n \mid 0 \le n : P.n) \;\text{ where }\; P.n : \;\; (\forall m \mid 0 \le m : b^{m+n} = b^m \cdot b^n)$$

clarifies what the "induction variable" is, by making it an explicit argument of P. Here, our style imposes a measure of precision that is almost impossible to obtain with an English argument. If one does not know how to write quantifications, it is difficult to explain the different roles of m and n in this proof by induction.

Finally, students' predicate-logic background allows us to go into mathematical induction more deeply than is usually possible in a first course. Consider a pair $(U, \prec)$, where U is a nonempty set and $\prec$ a binary relation over U. When can one use induction on $(U, \prec)$? Answering this question rigorously gives the students a much better understanding of induction than has hitherto been achieved.

The pair $(U, \prec)$ is *well founded* iff every nonempty subset of U has a minimal element (with respect to $\prec$). In Table 6, we prove that $(U, \prec)$ is well founded exactly when it admits induction (the second formula in the proof is the formal definition of well-foundedness, while the penultimate formula is the definition of induction[6]). This proof is so simple that students can be asked to reconstruct it on a test —and our experience is that 95% of them do so. Such performance is impossible using a proof in English, even though the two proofs rely on the same idea of defining a predicate P in terms of a set S (and vice versa). Students don't memorize the proof character for character; instead, they understand the basic idea and develop the proof afresh each time they want to present it.

[6]The formal definitions of well foundedness and induction require second-order predicate calculus.

Table 6: Equivalence of Mathematical Induction and Well-foundedness

$$\begin{array}{ll}
& (U,\prec) \text{ is well founded} \\
= & \quad \langle \text{Definition of well-foundedness} \rangle \\
& (\forall S \mid: S \neq \emptyset \ \equiv\ (\exists x \mid: x \in S \wedge (\forall y \mid y \prec x : y \notin S))) \\
= & \quad \langle\, X \equiv Y \equiv \neg X \equiv \neg Y \text{ ; Double negation} \rangle \\
& (\forall S \mid: S = \emptyset \ \equiv\ \neg(\exists x \mid: x \in S \wedge (\forall y \mid y \prec x : y \notin S))) \\
= & \quad \langle \text{De Morgan, twice} \rangle \\
& (\forall S \mid: S = \emptyset \ \equiv\ (\forall x \mid: x \notin S \vee \neg(\forall y \mid y \prec x : y \notin S))) \\
= & \quad \langle \text{Change dummy, using } P.z \ \equiv\ z \notin S \,\rangle \\
& (\forall P \mid: (\forall x \mid: P.x) \ \equiv\ (\forall x \mid: P.x \vee \neg(\forall y \mid y \prec x : P.y))) \\
= & \quad \langle \text{Law of implication, } p \Rightarrow q \ \equiv\ \neg p \vee q \,\rangle \\
& (\forall P \mid: (\forall x \mid: P.x) \ \equiv\ (\forall x \mid: (\forall y \mid y \prec x : P.y) \ \Rightarrow\ P.x)) \\
= & \quad \langle \text{Definition of induction over } (U,\prec) \,\rangle \\
& (U,\prec) \text{ admits induction}
\end{array}$$

Generating functions

Solving recurrence relations (homogeneous difference equations) using generating functions is sometimes taught in introductory discrete-math courses. The generating function $G(z)$ for a sequence x_0 , x_1 , x_2 , $\ldots$ is the infinite polynomial $x_0 \cdot z^0 + x_1 \cdot z^1 + x_2 \cdot x^2 + \ldots$, or

$$(\Sigma\, i \mid 0 \leq i : x_0 \cdot z^i) \quad .$$

The generating function for a sequence is just a different representation of the sequence.

Many useful generating functions can be written in a closed form, but for the student with little knowledge of the axioms for manipulating quantifications, the derivation of these closed forms is a black art. However, students with skill in manipulating quantifications can do more than follow the proofs; they can discover the closed forms themselves, once they have been shown the basic idea.

To see this, we now calculate the closed form of the generating function $G(z)$ for the sequence $c^0, c^1, c^2, \ldots$ for some nonzero c. The calculation starts with the definition of the generating function. Thereafter, the calculation unfolds in an opportunistic or forced fashion: at each step, the shape of the formula guides the development in an almost unique way. We have extended the hints of the proof with comments that motivate each step.

$$
\begin{array}{ll}
 & G(z) = (\Sigma i \mid 0 \le i : c^i \cdot z^i) \\
= & \quad \langle \text{Split off term —this is the simplest and almost the only} \\
 & \quad\quad \text{change possible.} \rangle \\
 & G(z) = c^0 \cdot z^0 + (\Sigma i \mid 1 \le i : c^i \cdot z^i) \\
= & \quad \langle \text{Arithmetic —isolate the complicated term} \rangle \\
 & G(z) - 1 = (\Sigma i \mid 1 \le i : c^i \cdot z^i) \\
= & \quad \langle \text{Change dummy —the obvious way to remove the summa-} \\
 & \quad\quad \text{tion is to use the definition of } G(z) \text{; change the range of} \\
 & \quad\quad \text{the summation to make it the same as that of } G(z) \text{.} \rangle \\
 & G(z) - 1 = (\Sigma i \mid 0 \le i : c^{i+1} \cdot z^{i+1}) \\
= & \quad \langle \text{Distributivity —expose the RHS of definition of } G(z) \rangle \\
 & G(z) - 1 = c \cdot z \cdot (\Sigma i \mid 0 \le i : c^i \cdot z^i) \\
= & \quad \langle \text{Definition of } G(z) \rangle \\
 & G(z) - 1 = c \cdot z \cdot G(z) \\
= & \quad \langle \text{Arithmetic} \rangle \\
 & G(z) = 1/(1 - c \cdot z)
\end{array}
$$

6 Intuition and understanding

The most frequent criticism we hear of our approach is that stressing formal manipulation will impede the development of intuition and understanding. Also, our critics fear that students will lose track of “meaning”, or semantics. We now address these issues.

On intuition and discovery

By intuition, one usually means direct perception of truth or fact, independent of any reasoning process; keen and quick direct insight; or pure, untaught, noninferential knowledge (*Webster's Encyclopedic Unabridged Dictionary*, 1989). There is simply no hope of teaching this —how can one teach something that is untaught, noninferential, and independent of any reasoning process? Of course, one can hope that students will develop an ability to intuit by watching instructors in math courses over the years. Intuition, after all, can be developed through experience. But this hit-or-miss prospect cannot be called *teaching* intuition.

On the other hand, a good part of mathematics is concerned with the opposite of intuition: with reasoning processes that complement our ability to reason in English. This part of mathematics can be taught, and our approach to logic is an excellent vehicle for that task.

The question may then arise whether students can be taught something about discovery that does not hinge on intuition. Here, our syntactic method of proof outshines the more conventional proof methods of reasoning in English. We are able to teach aids to discovery: with our disciplined, syntactic, proof style, we can teach principles and strategies whose application can indeed lead to proofs in many (but not all) cases. We have yet to see comparable principles and strategies for conventional English proofs.

Finally, in many cases, formalization and syntactic manipulation can set the stage for discovery. For example, recall the earlier discussion of closed forms of generating functions. The chance of intuition or English reasoning helping to discover closed forms is small, but freshmen and sophomores can be taught to discover them through syntactic manipulation.

We are *not* saying that intuition has no value, but only that we do not know how to teach it. We believe that in our own course, students gain as much intuition as in other courses. In our course, however, they also gain an understanding of where intuition is necessary in problem solving and where syntactic manipulation can help.

On meaning or semantics

It has also been said that semantics or "meaning" is lost in our approach to proofs. The English proof of distribution of union over intersection on page 4 is full of meaning, we hear, while our syntactic proof suppresses it.

However, any (sound) English argument can be formalized in a natural-deduction logic. The resulting natural-deduction proof is just as much a syntactic argument as an equational proof, so its English counterpart is just an informal version of a syntactic proof (in which inference rules are not mentioned). There is nothing "semantic" about it.

Thus, comparing proofs based on meaning or semantics is a red herring.

Let us discuss meaning or semantics in relation to the introduction of an operator like set union. The operator can be explained in several ways. One can illustrate set union with a Venn diagram, define in English what it means, show how to evaluate a set union, give examples, and give a formal definition. Having different views provides redundancy, which is helpful in promoting full comprehension, and our formalistic approach to teaching mathematics does not preclude meaning-ful discussions of alternative views. However, the student should be made to realize that for purposes of reasoning —constructing proofs— it is the axiomatic definition that is important. In fact, the student should view the axiomatic definition as encoding all the meaning of the object being defined.

On understanding

What really is at issue is the question of understanding: which approach, the equational or the conventional one, provides more understanding?

A proof should convey to a reader belief in a theorem. A proof provides evidence for belief, where the evidence consists of the facts (e.g. previously proved theorems) on which it rests and on how these facts interact to convince. We understand the proof when we understand which facts are used and how they interact. Full understanding also implies the ability to explain the proof to others and perhaps to prove other theorems with similar proofs. (Note that one may have an intuition about a theorem, based on some brief discussion of it, but that is not the same as a proof.)

Now look at the English proof of distribution of union over intersection on page 4. That proof does not state the facts on which the theorem rests. For example, it says, "If $y \notin A$, then, since $y \in A \cup B$ we must have $y \in B$", but there is no reference to a theorem that explains why this inference holds. This is typical of most proofs in English: the underlying facts are left to be inferred and found by the reader.

Second, in that English proof, it is difficult to see precisely how the facts interact. The proof cannot be easily learned and then explained to others. And, having seen the proof, students still have difficulty proving similar theorems. In fact, showing the proof to students gives little insight into proofs in general.

Thus, from the view of providing "understanding", the English proof is deficient.

Now turn to the equational proof on page 5. Every fact used in the proof is stated. The uses of the three inference rules are given implicitly by the structure of the proof, and the premises of uses of Leibniz's rule are stated explicitly. Thus, "understanding" this proof, in terms of the facts it uses and their interaction, is almost trivial. Further, the proof uses a simple proof strategy: eliminate operators using their definitions, perform manipulations, and reintroduce the operators. Since the strategy is one that is used often, it can be taught and then applied by students in many different cases. In fact, after studying this proof, a student should have little difficulty developing it afresh when it is to be explained to someone else. The student has full understanding.

In short the conventional proof method has trouble promoting full understanding, while the equational proof method makes understanding easy.

As another example, look at the proof of equivalence of well-foundedness of $(U, \prec)$ and mathematical induction over $(U, \prec)$. Rarely do texts give an English proof of this equivalence, because it would be too difficult. The equational proof, however, is fully understood by most of our students. Further, the equational proof stunningly brings to light the important fact upon which the proof is based, namely the ability to put predicates and sets in correspondence using the relation $P.z \equiv z \notin S$. The corresponding English proof, because its structure is so complex, hides this fact.

7 Experience with our approach

We have written a text, *A Logical Approach to Discrete Math* [5], that embodies our approach to teaching discrete math. The text covers the conventional topics of discrete math. In Spring 1993, we taught a discrete-math course to 70 students (mainly freshmen and sophomores) in the Computer Science Department at Cornell, using a draft of the text. The students in the course were mainly computer science majors, but there were also majors from mathematics, operations research, and engineering fields.

We had previously taught this course using a standard discrete-math text. The students, were much much happier with the new approach. Generally, students have not liked discrete math courses, seeing them as a hodge-podge of unrelated topics with no unifying theme and with little motivation. That changed. Taught with the new approach, students saw logic and proof as a unifying and interesting theme underlying all the topics being taught. Further, their comments indicated that they:

- Lost their fear of mathematics and mathematical notation.
- Gained a good understanding of proofs and their development.
- Acquired skill in formal manipulation and begin almost immediately to apply that skill in other courses.
- Gained an appreciation for rigor (so much so that they asked for it in later courses).

Further, we found that part of the six weeks spent on logic could be made up in the latter part of the course, because students understood the later material more rapidly. We also feel that material in subsequent courses can be taught more effectively and efficiently because students are better prepared, although we have no firm evidence on this yet.

Near the end of our discrete-math course, we asked the students to write a short essay explaining what they had learned. Five wrote negative comments, but the other 65 were overwhelmingly positive. This is the first time we experienced or heard of an overwhelmingly enthusiastic response to a discrete math course. All their comments are summarized in the first chapter of the Instructor's Manual [6], where the gestalt of the course is discussed. We repeat a few comments here.

> This course was really groovy, perhaps my favorite one . . . I am a math major who never thought I would enjoy doing proofs, until taking this course.
>
> Stressing logic has helped my methods of thinking. I definitely have a better understanding of proof.
>
> In my linear algebra class, I understand proofs with much less difficulty, and proving theorems about matrices has become easier as well.
>
> My math homework from last week ended with two proofs, and I was *happy* about it!
>
> There are real applications out there for predicate and propositional calculus and an observable benefit to formalism.
>
> I really enjoyed . . . [this course], because of the rigor and techniques of proof I learned.
>
> The class dissolved my fear of proving things, and I find proofs easier to tackle now.
>
> To be perfectly honest, I enjoyed it, which is surprising considering I do not enjoy math.
>
> This course taught me the most about how to think logically—in all the many courses I have taken in my Cornell career.

A draft of text [5] was also used in Fall 1993 by Sam Warford at Pepperdine, who considered his course "a big success".

We hope that others will try out this new approach to teaching math. We think that the approach produces students with far better math skills and that it would help to integrate the approach into all math curricula. In fact, a course for non-majors based on this approach could serve as a replacement for calculus.

Acknowledgements

The approach discussed in this paper was developed over the past 20 years by researchers in the formal development of algorithms. Researchers in that

field continually needed to manipulate formulas of the predicate calculus, but conventional logics, like natural deduction, required too much formal detail. The equational approach was used informally in the late 1970's, and the first text on the formal development of programs [4] used a rudimentary equational logic. Dijkstra and Scholten [2] developed the equational logic on which ours is based. The unification of quantification was also developed in the early 1980's; the first text we know of that talks about this topic is by Backhouse [1].

References

[1] Backhouse, R. *Program Construction and Verification.* Prentice-Hall International, Englewood Cliffs, N.J., 1986.

[2] Dijkstra, E.W., and C. Scholten. *Predicate Calculus and Program Semantics.* Springer-Verlag, New York 1990.

[3] Gentzen, G. Untersuchungen über das logische Schliessen. *Mathematische Zeitchrift 39* (1935), 176-210, 405-432. (An English translation appears in M.E. Szabo (ed.). *Collected Papers of Gerhard Gentzen.* North Holland, Amsterdam, 1969).

[4] Gries, D. *The Science of Programming.* Springer-Verlag, New York, 1981.

[5] Gries, D., and F.B. Schneider. *A Logical Approach to Discrete Math.* Springer-Verlag, New York, 1993.

[6] Gries, D., and F.B. Schneider. *Instructor's Manual for "A Logical Approach to Discrete Math".* Gries and Schneider, Ithaca, 1993. (To discuss obtaining a copy, email gries@cs.cornell.edu.)

[7] Lay, S.R. *Analysis with an Introduction to Proof.* Second Edition. Prentice Hall, Englewood Cliffs, NJ. 1990.

[8] Rosser, B. *Logic for Mathematicians.* McGraw-Hill, New York, 1953.

[9] Wiltink. A deficiency of natural deduction. *Information Processing Letters 25* (1987), 233-234.

David Gries, Short biographical sketch

Gries received his B.A. from Queens College in 1960, M.S. from Illinois in 1963, and Dr. rer. nat. from the Munich Institute of Technology in 1966 (all in math). After three years as assistant professor at Stanford, he left (because Stanford had no weather) for Cornell (where he has experienced weather for 25 years). He served as Chair of Computer Science in 1982–87.

Gries is known for his earlier texts *Compiler Construction for Digital Computers* (1971), *An Introduction to Programming: a Structured Approach* (1973, with Dick Conway), and *The Science of Programming* (1981). He is co-managing editor of Springer-Verlag's Texts and Monographs in Computer Science series, a main editor of *Acta Informatica*, and a managing editor of *Information Processing Letters*.

He was a Guggenheim Fellow in 1984–85 and is a Fellow of the AAAS. For his contributions to education, he received the ACM SIGCSE award in 1991 and the American Federation of Information Processing Societies' (AFIPS) education award in 1985. He was co-recipient of the 1976 ACM Programming Languages and Systems Best Paper award. He served as Chair of the Computing Research Association during its formative years 1987–89 and received its Service Award in 1991.

Fred B. Schneider, Short biographical sketch

Schneider received an M.S. (1977) and Ph.D. (1978) from SUNY Stony Brook in Computer Science. When he received his B.S. in CS and EE from Cornell in 1975, he resolved never to come back and never to work in academia. Three years later, he returned as assistant professor of Computer Science and has been there ever since. He served as associate chair and director of the department's undergraduate programs from 1991–1993.

Schneider's research concerns concurrent programming, particularly for fault-tolerant, real-time, and distributed systems. He has developed logics for reasoning about system behavior as well as algorithms and systems for this setting.

Schneider is a Fellow of the AAAS. From 1983–1988, he served on the College Board Committee for Advanced Placement in Computer Science. He is the managing editor of *Distributed Computing*, co-managing editor of Springer-Verlag's Texts and Monographs in Computer Science series, and a member of the editorial boards of *Annals of Software Engineering*, *High Integrity Systems*, *Information Processing Letters*, and *IEEE Transactions on Software Engineering*.